SaFT: Reliable Transport in Mobile Networks

Simon Heimlicher, Rainer Baumann, Martin May and Bernhard Plattner
Computer Engineering and Networks Laboratory, ETH Zurich
8092 Zurich, Switzerland
{heimlicher,baumann,may,plattner}@tik.ee.ethz.ch

***Abstract*— End-to-end transport protocols perform poorly in mobile environments, primarily due to the frequent route breaks induced by node mobility. We present a framework for hop-by-hop transport protocols and propose a new reliable transport protocol: SaFT (Store-and-Forward Transport). We evaluate its performance in mobile ad hoc networks vis-a-vis TCP. Overall, SaFT achieves up to 3 times faster delivery of messages in such networks. We conclude that store-and-forward protocols are suited well to provide reliable communication in mobile ad hoc networks.**

I. Introduction

Simulation studies [1] show that the standard reliable transport protocol of the Internet, TCP [2], performs poorly in mobile environments because it fails to respond appropriately to problems at lower layers such as route failures and link errors. Nevertheless, a large number of new transport protocols have been proposed that still use the same principles as TCP, i.e., they perform flow and congestion control as well as retransmissions from the end points of the connection. These proposals either employ a heuristic method to distinguish link failures from congestion (e.g. [5]) or they let intermediate nodes notify the source about link failures (e.g. [6]). The similarity of these proposals to TCP is in part due to the popularity of this protocol and in part related to the view that functions should be implemented at the highest possible layer (End-to-end Arguments [3]) and intermediate nodes should be stateless (Design Philosophy of the DARPA Internet Protocols [4]).

However, the characteristics of mobile networks are radically different from those of classical fixed networks because node mobility leads to frequent changes of the link topology. The real-world measurements in [7] support this view by showing, how short link lifetimes are in a mobile network in an office environment. We conclude from these works that the end-to-end paradigm of existing transport protocols should be reconsidered for mobile networks.

In mobile networks, it seems appropriate to include the intermediate hosts in the forwarding task. Such *hop-by-hop protocols* are able to operate effectively even when end-to-end routes are intermittent. However, not many hop-by-hop transport protocols have been proposed to date. One example is Split TCP [8], which splits long connections at certain intermediate nodes. A more recent example, the Delay Tolerant Networking group [9] also considers hop-by-hop transport mechanisms (Bundling), but intermediate nodes are assumed to be reliable.

Due to the lack of research on hop-by-hop transport protocols, we have developed a framework for hop-by-hop transport protocols following the store-and-forward principle. This framework provides flow control, congestion control, and end-to-end reliability. With this framework, we have developed SaFT, a reliable transport protocol for mobile networks. In simulation studies, we show that this protocol performs much better in mobile networks than TCP NewReno. In brief, the main contributions of this paper are:

- Design and implementation of a framework for transport protocols following the store-and-forward principle;
- Development of SaFT, a reliable transport protocol based on this framework; and
- Validation and performance evaluation of SaFT in a mobile network vis-a-vis TCP NewReno.

II. The Framework

As discussed in the literature (cf. [1]), primary causes of poor TCP performance in mobile networks are packet loss, route changes, and route failures. Consequently, the design goal of our framework is to overcome the limitations of traditional end-to-end transport protocols such as TCP and to (i) provide resilience to highly dynamic network conditions; (ii) enable transmission over intermittent end-to-end routes; and (iii) minimize end-to-end control traffic.

From an application perspective, the framework provides a reliable byte stream over a highly dynamic and unreliable network topology. The only requirement on the routing protocol is, that it provides the next hop to the destination if a route is available. The

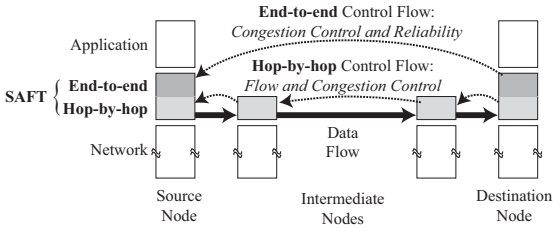


Fig. 1. The framework

framework is implemented on two sublayers, as shown in Figure 1. The *hop-by-hop layer* runs on every node and provides per-link flow control and congestion control. Data is managed in units of *fragments*, which only comprise a few IP packets to allow for fine-grained control over the hop-by-hop data transmission. The *end-to-end layer* operates at the end points of the connection. It performs global congestion control and guarantees reliable data delivery and can thus provide a reliable byte-stream interface to the application-layer, just like TCP. Data is managed as *segments*, which are data units comprising a few fragments. The framework accomplishes the three fundamental tasks of a reliable transport protocol—end-to-end reliability, flow control, and congestion avoidance—independently, as discussed below.

End-to-end Reliability: In a network of unreliable nodes, end-to-end reliability can only be ensured by the end points of the connection; therefore, an end-to-end retransmission mechanism is provided at the end-to-end layer of the framework.

Flow Control: Hop-by-hop protocols are able to control every link along the route separately, which is a clear advantage over end-to-end protocols, especially in mobile networks. Our framework offers a rate-based flow control algorithm that runs at the sending side of the link. The receiver informs the sender about the fill state of its buffer in every acknowledgment packet to avoid unnecessary data transmission.

Congestion Avoidance: Our framework provides both a hop-by-hop and an end-to-end congestion control mechanism based on a sliding-window algorithm. The hop-by-hop congestion control algorithm limits the number of fragments that the source and any intermediate host is allowed to send to its next hop without waiting for an acknowledgment. At the source node, the end-to-end congestion control algorithm limits the number of unacknowledged segments on a per-connection basis, which enforces a hard limit on the total amount of unacknowledged data allowed into the network. Furthermore, all segment retransmissions are scheduled at the source with an exponential back-off to guarantee that the system remains stable and to allow for fair co-existence with other transport protocols such as TCP. Since the framework is designed for networks with intermittent end-to-end routes, it does not have a connection establishment phase. When a node has data to send, it immediately begins the transfer and continues until the congestion control algorithm detects a problem with the receiver and stops the transmission.

A. Discussion

Compared to classical packet forwarding, store-and-forward protocols require additional processing power and memory for cross connections at every intermediate hop because fragments that could not be delivered to the next hop are stored on intermediate nodes and retransmitted a certain number of times. However, the amount of required buffer space is limited by the congestion control mechanisms discussed above, allowing the framework to run with as little as one packet's worth of buffer space.

As a performance improvement, intermediate nodes with free buffer space cache the most recently received fragments for a limited period of time. This allows to save bandwidth in the case of retransmissions, as follows. If a node receives a packet that belongs to a fragment that it has cached already, the receiver acknowledges the fragment to the sender and begins retransmission of this fragment towards the destination.

The performance of the hop-by-hop transfer of fragments is largely determined by the transmission rate and the retransmission timeout. Only accurate and up-to-date round-trip time (RTT) measurements allow a node to transmit at the correct rate and to avoid premature retransmission. With our framework, a node shares transmission rate and RTT data among all connections that use the same next hop, allowing these connections to operate with the most recent information. In the remainder of the section, we discuss how the framework provides resilience

against packet loss, route changes, and route failures.

Packet Loss: Since the framework provides 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 fragment and the last hop will retransmit the fragment. If a fragment acknowledgment packet is lost and a fragment is retransmitted even though the receiver has it in its cache already, the receiver immediately sends an ACK to the sender. In addition, every fragment ACK contains the sequence numbers of the last received fragments. Thus, in general, a fragment can be acknowledged by multiple independent ACK packets. The same mechanism is employed for segment ACKs.

It might be argued, that local retransmission is already provided by most MAC layer protocols and should not be duplicated at a higher layer. Since the transport layer protocol has knowledge about the final destination of every data packet, it is able to retransmit the packet to a different next hop when recovering from a link failure; note that this cannot be accomplished by the MAC layer protocol.

Route Changes: In order to avoid stale packets from congesting the network, every data and ACK packet contains a so-called *final acknowledgment number*, i.e, the sequence number of the last fragment received in sequence at the destination. Whenever a node receives a higher final acknowledgment number, it updates its status and flushes all fragments with lower sequence numbers. This effectively controls the cache sizes at intermediate hops and provides a redundant acknowledgment channel from the destination back to the source.

Route Failures: Current ad hoc routing protocols, such as AODV [10] and DSR [11], strive to provide either an end-to-end route or no route at all. As soon as a packet is lost, all routes using the unreliable link are considered to be down. Since hop-by-hop protocols do not depend on end-to-end routes, this functionality is counterproductive. The framework provides a route caching mechanism, allowing it to continue to use invalid routes for a specifiable period if the link to the next hop is up.

III. Evaluation

We compare the performance of SaFT with TCP NewReno [2] in a messaging application running over a mobile ad hoc network. We run the simulation in "ns-2"; all nodes use the IEEE 802.11 MAC layer running DCF with RTS/CTS and the AODV [10] routing protocol. In an area of 1000m × 3000m, nodes move according to the random-waypoint model [11] at a speed of 1–10m/sec. The connection pattern is as follows: 10 pairs of nodes are chosen randomly. One node of every pair sends 10 messages of 100kB to its peer at uniformly distributed points in time during the first 100 seconds (see [12] for a complete description of the simulation environment).

We run experiments for network sizes of 20, 30, 40, and 50 nodes and measure the time until all messages are transmitted. The average values of 50 seeds are plotted in Figure 2. With SaFT, the average arrival

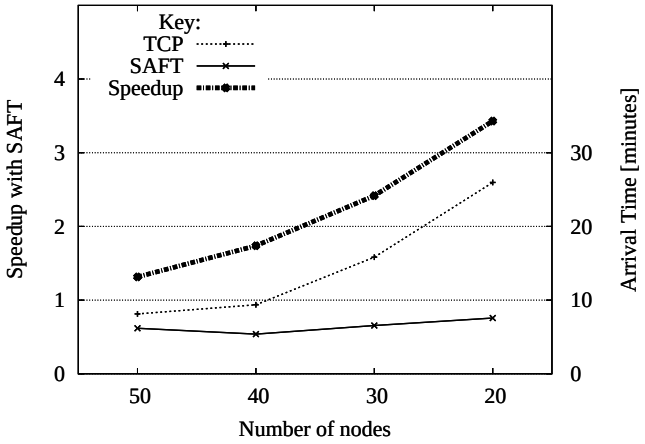
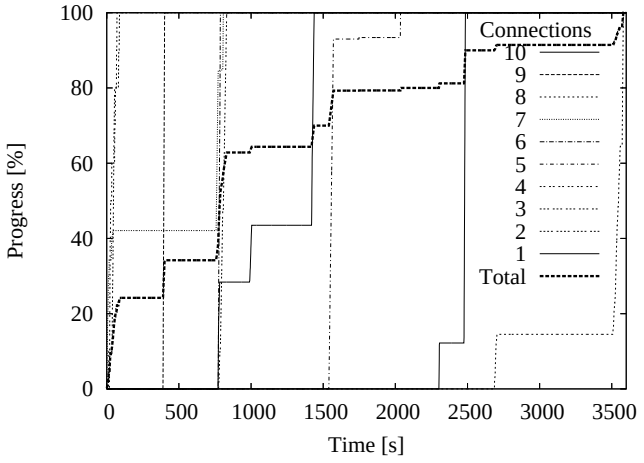


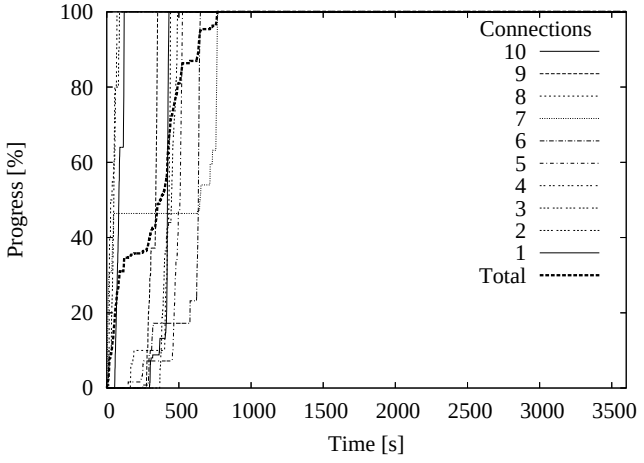
Fig. 2. Average message arrival time

time is almost independent of the number of nodes. In contrast, TCP performs poorly with less than 40 nodes. For instance in the example network with 30 nodes, it takes 17 minutes for all messages to arrive at the destination nodes with TCP. With SaFT, all connections finish already after 7 minutes.

In Figure 3, we show in one instance of the simulation experiment with 30 nodes, how the connections progress with TCP and SaFT, respectively. In Figure 3(a), the stop-and-go behavior of TCP in this experiment becomes obvious. SaFT in contrast transmits data more smoothly and much faster, as shown in Figure 3(b). Our analysis of the trace files has revealed that TCP sends data primarily on single-hop connections, and that typically only one connection is transmitting at a time. In contrast, SaFT starts to transmit data much earlier than TCP and frequently uses multiple single- and multi-hop routes. From this experiment, we conclude that SaFT is particularly well suited for networks with low node density.



(a) TCP



(b) SaFT

Fig. 3. Progress of ten connections

IV. Conclusion

In this paper, we have addressed the problem of reliable data transfer in mobile networks. Due to the limited suitability of classical end-to-end protocols, we have considered alternative transport protocols. Specifically, we have developed a framework that allows to design, analyze and evaluate transport protocols following the store-and-forward principle. With this framework, we have developed a store-and-forward transport protocol that we call SaFT. This protocol provides end-to-end reliability, flow control and congestion control at a very low cost in terms of memory and bandwidth overhead.

In order to evaluate the performance of SaFT, we have compared it with a traditional end-to-end transport protocol in a mobile ad hoc network. We have found that store-and-forward protocols lead to up to three times faster delivery of data while at the same time sharing communication resources much more fairly among single and multi-hop connections.

We believe that the performance of SaFT justifies the newly introduced overhead in terms of computing and memory efforts at intermediate nodes. Moreover, a store-and-forward transport protocol offers opportunities to develop new applications for mobile ad hoc networks. We are implementing SaFT on PDAs to perform real-world measurement studies.

References

[1] A. A. Hanbali, E. Altman, and P. Nain, "A Survey Of TCP Over Mobile Ad-hoc Networks," Institut National De Recherche En Informatique Et En Automatique (INRIA), Tech. Rep., May 2004.

[2] 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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] V. Lenders, J. Wagner, and M. May, "Analyzing the Impact of Mobility in Ad Hoc Networks," in *ACM/Sigmobile Workshop on Multi-hop Ad Hoc Networks: from Theory to Reality (REALMAN)*, Florence, Italy, May 2006.

[8] 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.

[9] 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.

[10] 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

[11] D. B. Johnson, D. A. Maltz, Y.-C. Hu, and J. G. Jetcheva, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," Internet Draft, IETF, February 2002.

[12] 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. 239, July 2005.