The Transport Layer Revisited

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 such as TCP perform poorly in mobile environments, primarily due to their inability to cope with the dynamics incurred by node mobility. We re-consider the design decisions that lead to the end-to-end design of the transport layer. To this end, we present a framework for reliable hop-by-hop transport protocols. Based on this framework, we design and evaluate such a protocol and compare its performance to TCP. Overall, our hop-by-hop protocol achieves up to 3 times faster delivery of messages in our experiments with mobile networks. We conclude that hop-by-hop protocols might be more suitable for reliable communication in mobile networks than end-to-end protocols.**

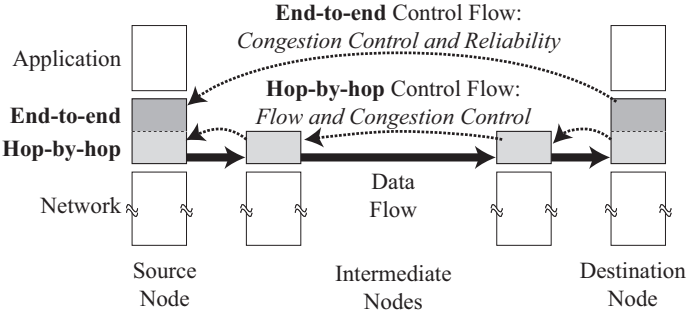


Fig. 1. The framework

I. Introduction

Wireless networking has become a common technology and is used for instance in wireless mesh networks with hundreds of nodes or in ad hoc networks comprising just a few nodes. In these scenarios, the majority of wireless devices are mobile. However, simulation studies [1]–[4] show that the standard reliable transport protocol of the Internet, TCP [5], [6], 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 are based 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. [7]) or they let intermediate nodes notify the source about link failures (e.g. [8]). 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 [9] and intermediate nodes should be stateless [10].

However, the characteristics of mobile networks are radically different from those of classical fixed networks because node mobility leads to frequent changes of the network topology. The real-world measurements in [11] 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 helpful to include the intermediate hosts in the data transfer. Protocols where intermediate nodes play an active role in controlling the forwarded data are called *hop-by-hop protocols* in this paper. Even though the assistance of the intermediate nodes allows for effective data transfer even when end-to-end routes are not always available, only few such transport protocols have been proposed to date. One example is Split TCP [12], which splits long connections at certain intermediate nodes. A more recent example, the Delay Tolerant Networking group [13] also considers hop-by-hop transport mechanisms (Bundling), but the responsibility for reliability is delegated to intermediate nodes (custodial transfer).

Due to the lack of research on hop-by-hop transport protocols, we have developed a framework for hop-by-hop transport protocols. This framework provides flow control, congestion control, and end-to-end reliability. In simulation studies, we show that a hop-by-hop transport protocol performs much better in mobile networks than the standard end-to-end protocol, TCP NewReno.

We then analyze in simple, static scenarios, what the reason for this considerable performance increase is. Our results show that an end-to-end transport protocol is fundamentally limited in its performance under circumstances where packet loss is high and route lifetimes are low.

In brief, the main contributions of this paper are:

- We analyze the fundamental advantages of hop-by-hop transport protocols in mobile networks as follows:
- We design and implementat a framework for hop-by-hop transport protocols; and
- We evaluate the performance of a protocol provided by this framework and compare it with TCP NewReno in both mobile and static scenarios.

The rest of this paper is organized as follows. In the next section, we describe our hop-by-hop framework. We discuss its key features, i.e., reliable data transfer in networks with high packet loss and frequent route failures, in Section III. In Section IV, we present a simulation study of hop-by-hop and end-to-end transport in a mobile network and analyze, why the hop-by-hop approach performs so much better. In Section V, we discuss related work. We conclude and discuss future steps in Section VI.

II. The Framework

As discussed in Section V, 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. Note that our framework allows multiple connections between a pair of hosts by providing a demultiplexing identifier, similar to the port number of TCP.

The 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. On this layer, 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. This layer thus provides 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.

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 waits until the routing protocol provides it with the address of the next hop to the destination and then immediately begins to transfer. The end-to-end congestion control algorithm ensures, that if the route becomes unavailable, the transmission is interrupted when the end-to-end congestion control window is exhausted (see below).

Our framework accomplishes the three fundamental tasks of a reliable transport protocol—end-to-end reliability, flow control, and congestion avoidance—independently. In the next sections, we will briefly discuss the mechanisms that provide these functions.

A. 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. Should no acknowledgement be received for transmitted data, un-acknowledged data is retransmitted by the source, respecting an exponentially increasing back-off interval.

B. Flow Control

Protocols following the hop-by-hop principle are able to control every link along the route separately. This is a clear advantage over end-to-end protocols in mobile networks. Our framework offers a rate-based flow control algorithm that runs at the sending side of each link. The receiver informs the sender about the fill state of its buffer in every acknowledgment packet to avoid unnecessary data transmission. The sender estimates an accurate sending rate based on the measuremed transfer time of each fragment with the same algorithm that TCP uses to calculate its roundtrip timeout. This estimated rate is shared among all connections that use the same next hop, leading to higher fairness and utility of links, as discussed later.

C. Congestion Avoidance

Our framework provides both a hop-by-hop and an end-to-end congestion control mechanism. These mechanisms counter two different problems: 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 in order to avoid transmissions to nodes that have moved away. Since not all fragments have to travel the same route, the hop-by-hop congestion control mechanism merely counts, how many fragments it has sent to a particular node and ignores their sequence numbers. In order to avoid deadlocks, the mechanism operates on a per-node basis, i.e., when the next hop changes, the sender is allowed to transmit a full fragment window's worth of data to the new next hop. Furthermore, retransmissios of failed fragments are prioritized.

The end-to-end congestion control mechanism has a job similar to the congestion control algorithm of TCP: it limits the number of un-acknowledged segments on a per-connection basis. It only runs at the source and enforces a hard limit on the total amount of unacknowledged data allowed into the network on behalf of each connection. Moreover, if no acknowledgement is received for a segment for a certain period, the source needs to retransmit the segment. Since a segment retransmission incurs a heavy load on the network, the source adopts an exponential back-off. As shown in [14], an exponential back-off guarantees that the system remains stable. Furthermore, exponential back-off allows fair co-existence of our protocol with other transport protocols such as TCP.

III. Discussion

Compared to classical packet forwarding, hop-by-hop protocols require additional processing power and memory for cross connections at every intermediate node because

fragments that could not be delivered to the next hop are stored on intermediate nodes and retransmitted a certain number of times. In order to limit these requirements, the amount of buffer space is tightly bounded by the congestion control mechanisms discussed above. For instance, with the settings we use in our simulation experiments (cf. Section IV), the buffer space used on an intermediate node is less than 100kB per connection. If less space is available, the performance degrades gracefully; our framework can run with as little as one packet size of buffer space, which is typically around 1kB.

As a performance improvement, intermediate nodes with free buffer space may 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 upon receipt of the first packet of the fragment.

The performance of the hop-by-hop transfer of fragments is largely determined by the transmission rate and the retransmission timeout (RTO). Only accurate and up-to-date measurements of the fragment transfer time allow a node to transmit at the correct rate and to avoid premature retransmission. With our framework, a node shares this data among all connections that use the same next hop, allowing all these connections to always operate with the most recent information.

In order to provide better service to connections that involve multiple hops, our framework provides a tunable interface scheduling algorithm that manages, which connection is allowed to transmit. In our experience, it is a good compromise to allocate the same share of time to every connection, thus allowing connections that send at high data rates to transmit more data than slower ones. This scheduling provides much better fairness and higher throughput for multi-hop connections than the preference for short connections of TCP, as will be shown in Section IV.

In the remainder of the section, we discuss how the framework provides resilience against packet loss, route changes, and route failures.

A. 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 upon receiving the first packet of the fragment. In addition, every fragment ACK contains the sequence numbers of the last few received fragments. Thus, in general, a fragment can be acknowledged by multiple independent ACK packets. As a result, the resilience of the acknowledgment mechanism against packet loss is dramatically increased at a very low cost in terms of transmission overhead.

TCP uses cumulative acknowledgment to achieve the same goal. However, with our protocol, consecutive fragments may travel different paths and thus cumulative acknowledgement is not applicable for fragments.

It might be argued, that local retransmission is already provided by most MAC layer protocols and should not be duplicated at a higher layer. However, as only 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.

B. 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 nodes and provides a redundant acknowledgment channel from the destination back to the source.

C. Route Failures

Current ad hoc routing protocols, such as AODV [15] and DSR [16], 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.

IV. Evaluation

In this section, we evaluate the performance of a specific instance of a hop-by-hop transport protocol that we developed with our framework. We compare our hop-by-hop transport protocol with TCP NewReno [6], which is the latest version of the most popular end-to-end transport protocol. First, we study the performance of a messaging application in a network of mobile nodes. Subsequently, we analyze both protocols in a static multi-hop scenario in order to determine, which characteristics of the hop-by-hop protocol are responsible for its much higher performance compared to the end-to-end protocol.

A. Simulation Environment

We use the network simulator "ns-2" [17] for all experiments. The distributed coordination function (DCF) of IEEE 802.11 [18] is used at the MAC layer. The 802.11

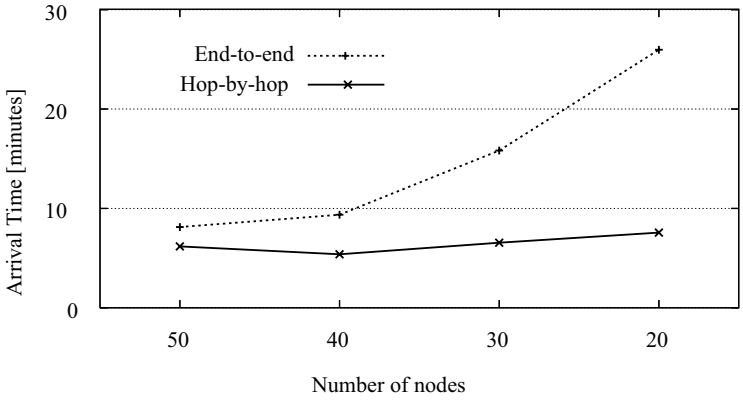


Fig. 2. **Mobile network.** Average message arrival time vs. number of nodes.

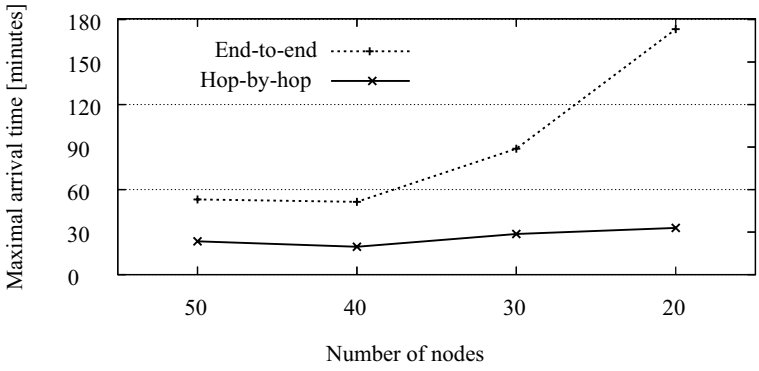


Fig. 3. **Mobile network.** Maximal message arrival time vs. number of nodes.

DCF uses an RTS/CTS handshake for unicast transmissions to neighboring nodes for packets larger than a given threshold (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 [15] routing protocol, specifically the "AODV-UU" [19] implementation for ns-2. We compare our hop-by-hop protocol with TCP NewReno [6], as implemented in the ns-2 "TCP/FullTcp/Newreno" agent. We use an FTP application with a packet size of 1000 bytes.

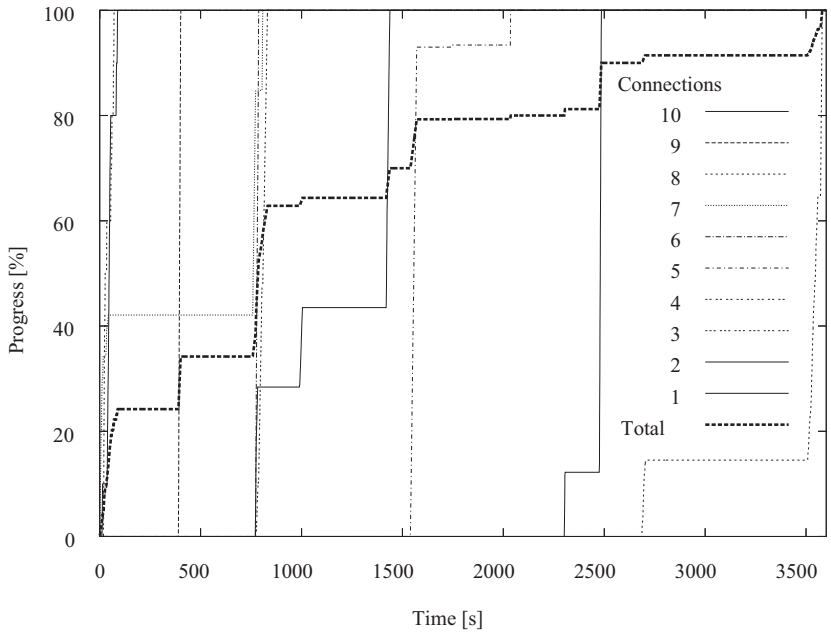
1) Parameters: There are a few parameters to be set in our framework. The most important ones are the segment and fragment sizes. Learned from some preliminary experiments, we set the segment length to 4 fragments and fragment length to 8 packets in all experiments. This results in a segment size of 32kB and a fragment size of 8kB.

B. Mobile Network

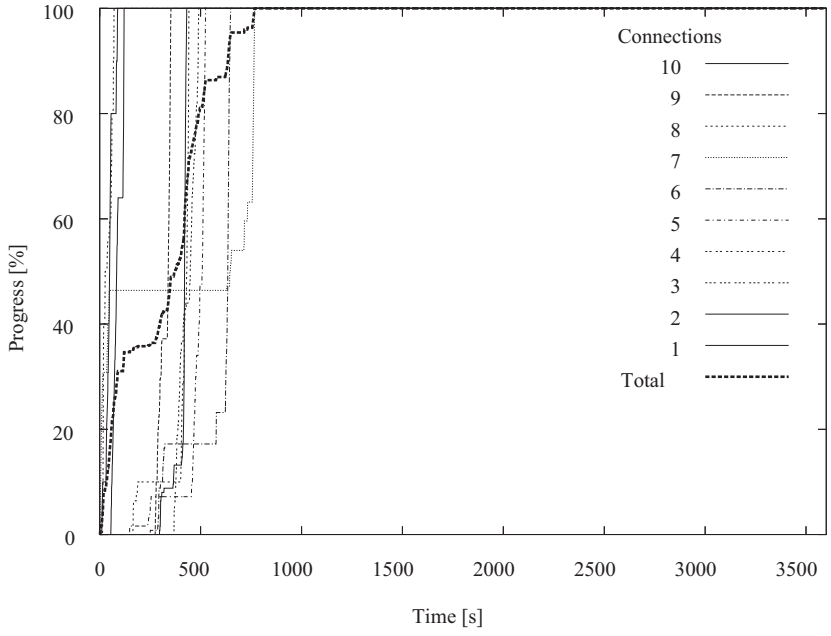
We show an example scenario of a messaging application in a mobile network. In an area of 1000m × 3000m, nodes move according to the Random-waypoint model [16]. Every node chooses a random point and a speed of 1–10m/sec. It then moves to this point and chooses the next waypoint, and so on. (We are aware of the discrepancy of random-waypoint mobility and real human mobility. Nevertheless, we consider this model to be a reasonable benchmark for the protocols under examination.) 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. Hence, the total amount of data to be transferred by the network is 10'000kB.

We run experiments for network sizes of 20, 30, 40, and 50 nodes and measure the time until all messages have arrived at the receivers. The average and the maximum values from 50 differently seeded runs of these experiments are plotted in Figure 2 and Figure 3, respectively. With our protocol, the average arrival 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 the end-to-end protocol TCP. With our hop-by-hop protocol, all connections finish already after 7 minutes.

In order to show the different behavior of the two approaches, we examine the plot from a randomly chosen



(a) End-to-end



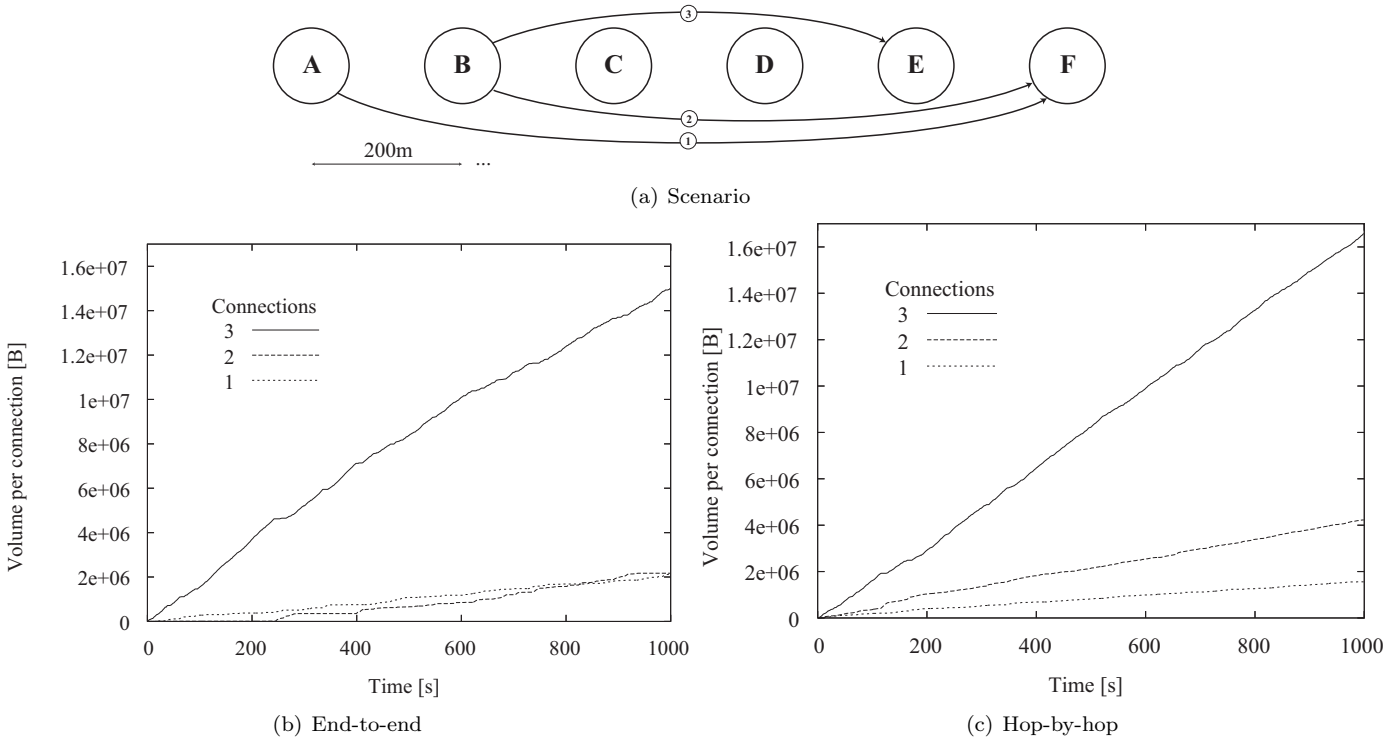
(b) Hop-by-hop

Fig. 4. **Mobile network.** Progress of each connection.

scenario with 30 nodes. In Figure 4, the progress of the 10 connections is plotted. These plots show, how the volume of data received by the destination nodes increases over time. With TCP (cf. Figure 4(a)), it takes up to one hour for all messages to arrive at the destination nodes. With our protocol, all connections finish within one quarter of the time (cf. Figure 4(b).

Furthermore, it is obvious, that the progress of the data transfer with the hop-by-hop protocol is much more steady than with TCP. Our analysis of the trace files has shown that TCP transmits data primarily over single-hop connections, and typically, only one connection is transmitting at a time. In contrast, our protocol uses multi-hop routes, sharing the limited bandwidth much more fairly among multiple connections.

Additionally, the hop-by-hop protocol starts to transmit data much earlier than TCP. The late start of TCP is in part due to the frequent pseudo route failures that occur with this protocol. Such failures are not due to node mobility but are caused by TCP's bandwith probing mech-



(a) Scenario

(b) End-to-end

(c) Hop-by-hop

Fig. 5. **Multiple connections sharing a collision domain.** Volume transferred vs. time.

anism. As explained in Section II, our hop-by-hop protocol estimates the optimal retransmission timeout and sending rate on every link individually, and multiple connections that use the same next hop share these estimates.

From this experiment, we conclude that the hop-by-hop approach is particularly well suited for networks with low node density, i.e., for networks where connections are intermittent.

In the next section, we will have a closer look at the behavior of TCP and our hop-by-hop protocol in a static scenario where multiple connections run through one collision domain.

C. Analysis

The static scenario serves to analyze the specific characteristics of our hop-by-hop approach in an observable environment. Since bandwidth is the limiting resource in wireless networks, we are particularly interested in the behavior of TCP and our protocol when it comes to sharing this resource among multiple connections. It is well known that when several TCP connections share one link, TCP devotes considerably more bandwidth to connections with lower RTTs (cf. [12]). We examined the behavior of TCP and our hop-by-hop protocol in the following scenario: six nodes are placed in a row with a distance of 200m. Three connections are in place, as shown in Figure 5(a). The first connection spans all five hops, the second one crosses four hops and the third connection only runs over three hops.

For every connection, we plot the amount of transmitted data over time. The results of TCP are shown in Figure 5(b). While the three-hop connection transfers about 15MB in 1000s, the longer connections only obtain a share of around 2MB each. As can be seen, connection 2 starts transmitting after around 4 minutes. The analysis of the trace files reveals that the source node B only finds a route to its destination F at this time. Apparently, the AODV routing protocol has great difficulty to establish this route due to the heavy use of the middle section of the route by the other two connections. At around $t = 268$s, the route is lost and re-established at $t = 401$s.

With our hop-by-hop appraoach, the bandwidth of the shared links is allocated more equally, as shown in Figure 5(c). There are no apparent periods with zero throughput, and the achieved performance of every connection is directly related to the number of hops it has to cross. Since the end-to-end congestion control mechanism of our protocol applies the same limits to all connections, the source of the six-hop connection has to wait for acknowledgement packets more frequently than the source of the three-hop connection. While this reduces the transmission rate of long connections, it allows for a higher total throughput of the network.

V. Related Work

The standard reliable transport protocol of the Internet, TCP (transmission control protocol) [5], is the most widely used reliable transport protocol of today. However, TCP

was designed for wired networks with stable topology, and extended analysis of its performance in mobile networks ([1], [3], [4]) has revealed that TCP is not suitable for such environments. TCP fails to respond appropriately to problems at lower layers such as route failures and link errors. Consequently, a number of modifications to the primary TCP mechanisms have been proposed to address these problems.

In [2] (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 quite similar approach is proposed in [20] (TCP-F). Another proposed protocol called ATCP [21] tries to solve this problem utilizing network layer feedback.

While the aforementioned approaches rely on cooperation and feedback control messages from lower layers, [22] employs a heuristic technique to distinguish route failures and congestion. When timeouts occur 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 [23] (TCP-DOOR), and [7] (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.

A recently published TCP-related protocol is the Transport Protocol for ad hoc Networks (TPA) [24]. TPA employs a similar congestion control mechanism as TCP, but introduces a novel retransmission mechanism. This mechanism first delays retransmissions and continues to send new packets, which increases the chance that a delayed acknowledgement arrives before the corresponding packet is retransmitted unnecessarily. However, the TPA protocol is only evaluated in a static scenario (six nodes in a row). In this basic setting, TCP exhibits good performance, but TPA achieves a slightly higher throughput while requiring a lower number of packet retransmissions. If background traffic across the three central hops of the topology is added, TPA largely outperforms TCP and only requires a fraction of the number of retransmissions of TCP, resulting in higher bandwdith efficiency.

The TCP-variant protocols cited above improve the performance of TCP in certain types of mobile networks, but since these protocols are limited to operate at the end points of the connection, their performance depends to a large extent on the availability of end-to-end routes. Particularly, when route failures occur, routing protocols are recalculating complete routes from a sender to a destination. This procedure is time-consuming and degrades TCP performance as nodes remain inactive for large periods of time.

The Ad hoc Transport Protocol (ATP) [8] is a recently published transport protocol that operates mainly end-to-end, but differs fundamentally from TCP in that it uses rate-based flow control based on feedback from intermediate nodes. This protocol depends not only on information from the routing protocol, but also from lower layers for the detection, avoidance and control of congestion, estimation of a reasonable transmission rate, and detection of route 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. In stark contrast, our hop-by-hop transport protocol only requires minimal cross-layer information, i.e., the address of the next hop to the destination.

To our knowledge, not many transport protocols were proposed that operate hop-by-hop. One notable proposal is Split TCP [12]. The authors of this proposal found that in mobile networks, long (multi-hop) TCP connections suffer severely from route failures due to mobility. They propose a scheme that separates long connections at certain intermediate nodes (so-called proxies) into shorter segments. If a node is a proxy, it buffers TCP packets and acknowledges them to the previous proxy with a local acknowledgement packet. The buffered packets are then forwarded as usual and buffered again at the next proxy. To guarantee end-to-end reliability, the destination node sends an acknowledgement back to the source. The basic operation principle of our framework is similar to the one of Split TCP. However, our framework is dedicated to be resilient against route changes, failures, and high packet loss rate. By sharing the most recent measurements of the link properties among connections, our protocol needs less probing, and as a result, achieves higher efficiency.

The Delay Tolerant Networking group [25] is a research project that considers hop-by-hop transport mechanisms. In its extreme case [13], the authors consider the network as an interplanetary Internet. However, Bundling, [26] their hop-by-hop transport protocol, does not replace TCP but is rather an intermediate layer between an existing transport protocol and the application layer. Its task is 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 are required to buffer all incoming bundles until they have been forwarded successfully (custodial transfer). The Bundling protocol is thus unsuitable for the networks we target in this paper where intermediate nodes are battery-driven and not reliable.

The Haggle project [27] is exploring a network model where user mobility is used to disseminate data and these networks are called pocket-switched networks [28]. Applications are assumed to be delay-tolerant and end-to-end connectivity intermittent. Since messages are not acknowledged by the destination, no reliability can be provided. Rather, Haggle is exploring ways of communicating effectively with minimal requirements regarding connectivity and control overhead.

VI. Conclusion

In this paper, we address the problem of reliable data transfer in wireless mobile networks. It has been shown in related studies that classical end-to-end protocols that were designed for fixed networks do not run as smoothly in such environments. Consequently, we consider alternative approaches in our work.

Specifically, we have developed a framework that allows to design, analyze, and evaluate transport protocols following the hop-by-hop principle. This framework provides protocols that guarantee 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 the hop-by-hop approach, we have compared a protocol simulated by our framework with a traditional end-to-end transport protocol in a mobile network. We have found that hop-by-hop 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.

Our further investigation in static environments has shown that the performance advantage of our hop-by-hop protocol is due to three features inherent to the hop-by-hop principle: (i) the precise control it can exert on every single link of the end-to-end path, (ii) its ability to recover from packet loss locally, and (iii) its lower end-to-end communication overhead.

We believe that the performance improvement of our hop-by-hop protocol justifies the newly introduced overhead in terms of computing and memory efforts at intermediate nodes. Moreover, a hop-by-hop transport protocol offers opportunities to develop new applications for mobile networks.

References

[1] M. Gerla, K. Tang, and R. Bagrodia, "TCP Performance in Wireless Multi-hop Networks," in *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, February 1999.

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

[3] 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.

[4] Z. Fu, X. Meng, and S. Lu, "How Bad TCP Can Perform in Mobile Ad Hoc Networks," in *IEEE Seventh International Symposium on Computers and Communications (ISCC'02)*, July 2002.

[5] J. Postel, "Transmission Control Protocol," RFC 793 (Standard), Sept. 1981, updated by RFC 3168. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[6] 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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] V. Jacobson and M. Karels, "Congestion Avoidance and Control," in *Proceedings of the ACM Symposium on Communications Architectures and Protocols (SIGCOMM '88)*, August 1988, pp. 314–329. [Online]. Available: citeseer.ist.psu.edu/654992.html

[15] 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

[16] 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.

[17] "Network Simulator ns-2," source code available: http://www.isi.edu/nsnam/ns/.

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

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

[20] 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.

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

[22] 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.

[23] 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.

[24] E. Ancillotti, G. Anastasi, M. Conti, and A. Passarella, "TPA: A Transport Protocol for Ad hoc Networks," in *Proceedings of the Tenth IEEE Symposium on Computers and Communications (ISCC 2005)*, June 2005, pp. 27–30.

[25] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, J. Scott, K. Fall, and H. Weiss, "Delay tolerant network architecture," Mar. 2007.

[26] K. Scott and S. Burleigh, "Bundle protocol specification," Aug. 2006.

[27] "The Haggle Project," http://www.haggleproject.org/.

[28] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket switched networks and human mobility in conference environments," in *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, 2005.