Connection Management in a Resilient Transport Protocol

Truc Anh N. Nguyen[a] and James P.G. Sterbenz[a,b,c]
[a]Information and Telecommunication Technology Center, Department of Electrical Engineering and Computer Science, The University of Kansas, Lawrence, KS, 66045, USA
[b]School of Computing and Communications (SCC) and InfoLab21, Lancaster LA1 4WA, UK
[c]Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

Abstract

With the motivation to develop a resilient and survivable Future Internet, we are constructing a configurable and adaptive multipath transport-layer protocol called ResTP. While some other composite protocols mainly emphasize their flexibility to provide services to multiple application classes operating on top of various network environments, our ResTP focuses on increasing the resilience level of the protocol in particular and the network in which ResTP is deployed in general, by following a set of design guidelines derived from the ResiliNets framework. In this framework, resilience has a broad definition that subsumes multiple disciplines, including survivability, disruption tolerance, and dependability. The implementation of ResTP employs modular programming to decrease the complexity while increasing its extensibility. ResTP uses a tunable header to minimize unnecessary overhead. In this paper, we present the design and implementation of ResTP connection management, including its connection establishment, monitoring, and termination. Through simulations in ns-3, we show that ResTP achieves a comparable performance to UDP in its connectionless mode with its ability to optimize the header. We also compare ResTP opportunistic and TCP handshake schemes and show that the opportunistic outperforms the 3-way handshake, especially in the presence of long delay and SYN drop. When using the same connection management scheme as TCP with ARQ for error control and bulk-send application, ResTP outperforms TCP, especially in the presence of random loss.

1 Introduction and motivation

The Transmission Control Protocol (TCP) [1] and the User Datagram Protocol (UDP) [2] have been the standard transport-layer protocols in the Internet for decades. While the lightweight UDP provides a connectionless and unreliable data transfer service, TCP provides a connection-oriented, reliable service with flow control, congestion control, and in-order data delivery. UDP is typically used to carry network management traffic DNS name translation. On the other hand, TCP is typically used to carry traffic from electronic mail, remote terminal access, file transfer, and the killer application in the early 1990s, the Web. However, the emergence of new application classes together with the advancement in networking technologies and paradigms that introduce new network environments have challenged the fixed services provided by UDP and TCP and violated their design assumptions. For example, while some developers of multimedia applications such as Internet telephony, real-time video conferencing, and streaming of stored audio and video prefer to use UDP instead of TCP since these applications can tolerate some amount of data loss, other developers argue with this selection due to the lack of congestion control in UDP, which has been proved to play a crucial role in sustaining Internet stability. The TCP congestion control algorithm has its own drawbacks when the protocol is deployed in wireless networks. Wireless channels are known for their high error rate, and a packet loss is more likely corruption-based than congestion-based [3]. Unfortunately, since TCP was designed for wired environments, it invokes its congestion control to reduce its sending rate on every occurrence of data loss irrespective of its cause resulting in a significant degradation in TCP throughput. In addition, the TCP assumption of a stable end-to-end (E2E) path between communicating parties is violated by frequently partitioned networks such as in deep space environments partly due to long delay and high node mobility but with limited power. Over the years, numerous efforts have been made to address TCP drawbacks and enhance its performance leading to a plethora of TCP variants, but each variant is only a temporary solution that targets a single issue for a particular type of network. TCP extensibility is very limited, due in part to its rigid intertwined error, flow, and congestion control [4]. Furthermore, there are services that are not provided by the standard Internet transport protocols such as throughput and timing guarantees. Last but not least, the Internet with UDP and TCP is not a resilient system.

With the motivation to develop a resilient and survivable network and to address the limitations of the standard protocols in face of the rapid growth of networking technologies, we are constructing a configurable and adaptable resilient multipath transport-layer protocol called *ResTP*. ResTP does not only emphasize its flexibility in providing alternative configurations based on different service requirements and underlying environments similar to some other composable protocols including TP++ [5] and CTP [6], but also focuses on increasing the resilience

level of the protocol and the network in which it is deployed. The design of ResTP follows a set of principles derived from the ResiliNets framework [7], and its implementation employs modular programming to decrease complexity while increasing its extensibility. As a composable transport protocol, ResTP supports alternative algorithms for each service that it provides. The mechanisms across multiple services can be mixed and matched to meet the requirements and characteristics of various application and network types. ResTP is implementable at both end-systems and gateways. At a gateway, the protocol performs seamless splicing with the conventional TCP and UDP protocols, as well as splicing ResTP segments for Disruption-tolerant networks (DTNs).

In this paper, we present the design and implementation of the ResTP connection management service, including connection establishment, monitoring, and termination. We perform simulations of the three connection management schemes supported by our protocol in ns-3 [8], including connectionless, connection-oriented with TCP's 3-way handshake, and opportunistic establishment. We show that in its connectionless mode, ResTP achieves a comparable performance to UDP due to its ability to optimize the header to reduce unnecessary overhead. We also compare the ResTP opportunistic algorithm with TCP's 3-way handshake and see that the opportunistic approach outperforms TCP, especially in the presence of high latency and SYN drop. The behavior of the opportunistic scheme is favorable to short traffic flows that impose a strict limit on data transfer duration. For similar configuration as TCP including the 3-way handshake, ARQ error control, and with bulk-send application, ResTP outperforms TCP when packet corruption occurs.

The rest of the paper is organized as follows: In Section 2, we start with a brief survey of available mechanisms to implement the connection management of a transport-layer protocol. The survey is then followed by an overview of the ResiliNets framework and our definition of *resilience*. We also provide more details on ResTP functionality with the emphasis on its header, five reliability modes, and connection management service. We conclude the section by referencing a few other composable transport protocols from the literature. In Section 3, we explain our implementation of ResTP connection management schemes in ns-3. In Section 4, we present our simulation setup, results, and analysis of these algorithms. Finally, in Section 5, we conclude our paper and give directions for future work.

2 Background and related work

We begin this section with an explanation of transport-layer connection management and a very brief survey of some available mechanisms proposed for this service. We then give an overview of the ResiliNets framework that guides us through our development of ResTP to achieve resilience and survivability and highlight the particular principles that we apply to design and implement ResTP connection management, one of the many services provided by the protocol that we study exclusively in this paper. We follow with a discussion on ResTP header and its five reliability modes, and end the section by quickly reviewing some other composable transport protocols.

2.1 Connection Management

Connection management refers to the mechanism employed by a transport protocol to allocate, synchronize, and deallocate state variables between communicating hosts while allowing each party to establish the necessary parameters of the ensuing data transfer [9]. In a reliable connection management that can be achieved by combining handshaking techniques with timers and unique connection identifiers, a complete data transmission with no ambiguity caused by duplicate data or acknowledgments from either the current connection or the previous ones is guaranteed [10]. Connection management mechanisms are classified into two categories: connectionless and connection-oriented.

In a connectionless transport such as UDP, the associating hosts exchange individual datagrams and normally maintain no information regarding the flow, thus cannot protect the data from missequencing and duplication [11]. On the other hand, in a connection-oriented transport such as TCP and its variants, the end systems exchange their data over a connection that is either explicitly or implicitly initialized and torn-down with state information maintained by both parties throughout the connection lifetime. Connection-oriented schemes are further divided into two groups: handshake-based (or packet-exchanged based) and timer-based. In handshake-based, the hosts are required to explicitly exchange control messages (signaling) to establish and terminate their connection. With this approach, installed state is normally maintained until explicitly removed by control messages, which is also referred as pure end-to-end (E2E) hard state management. APPN [12, 13, 14, 15] and Datakit [16, 17, 18] implement this pure hard state scheme. Furthermore, at the transport layer, signaling may be accomplished in-band or out-of-band. In-band signaling is multiplexed with data packets on the same connection while out-of-band signaling gets transmitted over a separate flow. Finally, timer-based protocols such as Delta-t [19], VMTP [20], and TP++ [5] implement the soft state management approach in which timers are used to determine the connection-state retention intervals at the end hosts, and all installed state will eventually time out unless they are refreshed. In fact, TCP connection establishment is handshake-based while its termination is a hybrid approach that combines both explicit control messages and timers.

2.2 ResiliNets Framework

In the ResiliNets framework, *resilience* is defined as *the ability of the network to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operation* [7]. This broad definition of resilience subsumes multiple disciplines, including those relating to challenge tolerance such as fault tolerance, survivability, disruption tolerance, and traffic tolerance, those relating to trustworthiness such as dependability with reliability and availability as the two major aspects, secu-

rity, and performability, and the two disciplines that connect challenge tolerance and trustworthiness: robustness and complexity. To direct the development of a resilient network, the framework establishes a set of design principles, and these guidelines are followed closely during our ResTP creation and implementation process.

Specifically, the first principle that we practice when designing each of ResTP services, including its connection management that is studied in this paper is **diversity**. While redundancy allows us to achieve fault tolerance, diversity prevents fate sharing and enables survivability. Each ResTP function implements different algorithms. The selection of these algorithms is carefully considered and their **resource tradeoffs** are examined through many simulation analysis since the **complexity** of a system can be increased due to the resilience mechanisms that are employed. The implementation of ResTP as a whole exploits modular programming techniques to reduce **complexity** and increase the protocol's **adaptability** and **evolvability**. Each service is a pluggable component that can be invoked when needed. The evovability of ResTP helps it cope with the evolution of network architecture and the emerging of new applications and allows any additions of new algorithms with very minimal modifications to the existing code.

2.3 An Overview of ResTP

ResTP is the general-purpose version of the Aeronautical Transport Protocol (AeroTP) [21, 22], a domain-specific transport protocol for a highly-dynamic airborne telemetry network environment. ResTP was originally proposed, and its design was discussed in our previous publications [23, 24]. To provide the background for our study of ResTP connection management in this paper, in this section we give an overview of the protocol, focusing on its header, connection management, and reliability modes.

2.3.1 Header

Figure 1 displays the format of a full ResTP transport protocol data unit (TPDU). While a detailed explanation of each of the header fields can be found in our previous publications, we want to emphasize here that the header is tunable. Except for the first eight bytes present in every ResTP configuration, the other fields are only included when needed for a particular setting. The first four bytes are necessary since ResTP is a composite protocol with many attributes and algorithms to be negotiated between communicating endpoints. The source and destination port numbers are essential for multiplexing and demultiplexing. When operating in its connectionless and unreliable mode, the ResTP header only includes the addition of the header CRC, which is the sixth word in the complete header shown in Figure 1. This optimization reduces unnecessary overhead as we will see shortly in the simulation and analysis section below.

2.3.2 Connection Management

The implementation of ResTP connection management supports both connectionless and connection-oriented approaches. The connectionless mode with none of the flags in the flow header field set is similar to what is used for

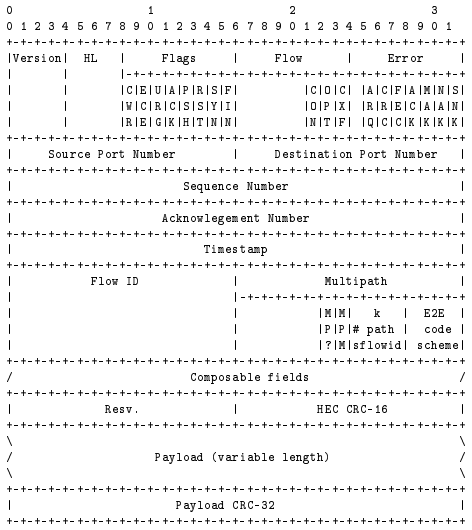


Figure 1 ResTP TPDU format

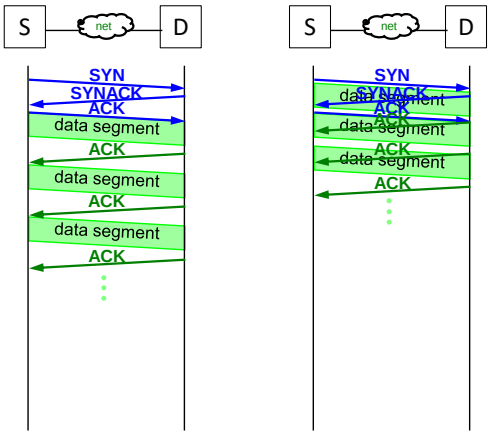
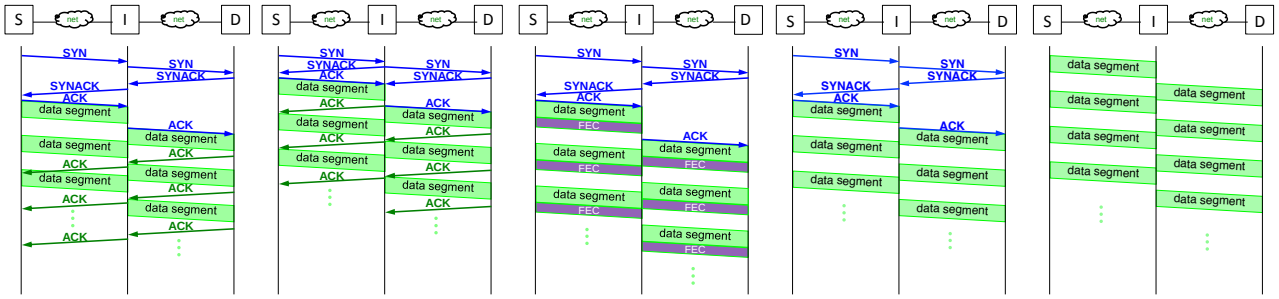


Figure 2 3-way handshake vs. opportunistic

UDP. The connection-oriented mode utilizes TCP's 3-way handshake connection establishment and its hybrid connection termination to ensure complete data transfer without missequencing and duplication (the `CON` Flow flag bit set). However, one of the drawbacks of TCP is the cost of one RTT to set up a connection, which is especially inefficient for short flows in a long-delay environment such as satellite or the interplanetary network. Hence, ResTP also provides an alternative opportunistic connection establishment algorithm in which data are allowed to be overlapped with signaling messages (the `OPT` Flow flag bit set). We will show in Section 4 that this overlapping assists ResTP in sustaining its performance when operating over high-latency channels and incurring SYN drop. Figure 2 illustrates the difference between the TCP's 3-way handshake (left side) and the opportunistic (right side) connection establishment techniques. In the opportunistic approach, immediately after a SYN segment is transmitted, a data segment is piggybacked with the `SYN` flag set. This `SYN` flag allows the connection to be initialized even when the initial SYN message is lost without the need to wait for SYN retransmission. In other words, the receiving side is able to accept the connection request and respond with a SYN-ACK when it receives either the initial SYN or the subsequent data segment. When being implemented at gate-



(a) Fully reliable **(b)** Nearly reliable **(c)** Quasi reliable **(d)** Unreliable connection **(e)** Unreliable datagram

Figure 3 ResTP transfer mode flow diagrams

ways, ResTP supports custody transfer similar to what is used in the Bundle Protocol for Delay-Tolerant Networks (DTNs) [25] with the `CXF` Flow flag bit set.

2.3.3 Reliability Modes

Similar to connection management, for error control, ResTP also implements multiple alternative error correction algorithms, including automatic repeat request (ARQ), forward error correction (FEC), and the combined hybrid ARQ (HARQ). By coupling the connection management and error control techniques, ResTP defines five transfer modes to satisfy service requirements from different application types:

Fully-reliable connection mode: This mode guarantees correct data delivery by preserving the E2E ACK semantics as illustrated in Figure 3a.

Nearly-reliable connection mode: In this mode, since the gateway uses custody transfer and immediately returns ACKs back to the sender with the assumption that data will be eventually delivered to the destination using the ARQ system (Figure 3b). This mode provides reliability, but does not guarantee correct data delivery.

Quasi-reliable connection mode: This mode uses some level of statistical reliability by using FEC instead of ARQ and ACKs as shown in Figure 3c.

Unreliable connection mode: This mode implements the connection-oriented or opportunistic connection management methods but provides no error control as shown in Figure 3d.

Unreliable datagram mode: This mode provides no reliability with connectionless flow management technique as shown in Figure 3e.

Although the figures only demonstrate the use of the 3-way handshake connection management scheme, all of the connection modes are also compatible with the opportunistic technique. In this paper, together with the three connection management techniques, we study the performance of the fully-reliable and unreliable datagram modes. As highlighted previously, the ResTP header is optimized for each mode. Figure 4 illustrates the 12-byte unreliable datagram header, and Figure 5 illustrates the 24-byte fully-reliable header. Note that we only study the single-path mode of ResTP in this paper. The unreliable and fully-reliable header are 4 bytes larger than the UDP and TCP header, respectively, due to the first word.

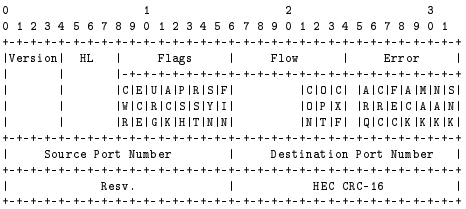


Figure 4 Optimized header for unreliable datagram mode

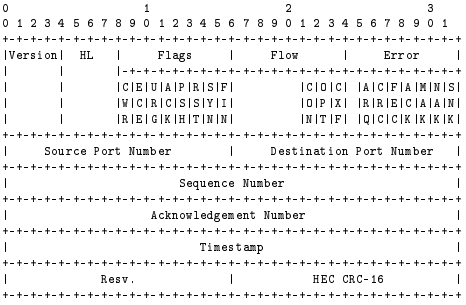


Figure 5 Optimized header for fully reliable mode

2.3.4 Related Work

There have been other composable transport-layer protocols proposed in the research community. Similar to ResTP, TP++ [5] is designed to carry multiple application classes, including transactions, bulk data transfer, and delay-sensitive services. However, TP++ is a single-path only protocol and only provides a mixture of mechanisms for its error control while utilizing a timer-based connection management scheme and assuming that congestion control is implemented by the underlying network. CTP [6] is a configurable and extensible transport protocol that is implemented using the Cactus microprotocol composition framework [26]. It provides various transport-layer services and functionalities, including reliability, ordering, security, jitter control, congestion control, flow control, data and header compression, MTU discovery, message fragmentation and collation, and connection management, where each function is implemented by multiple algorithms. CTP is also a single-path only protocol. Moreover, both TP++ and CTP do not target achieving resilience and survivability as our ResTP does.

3 Implementation

This section presents our implementation of the ResTP connection management schemes. Similar to TCP, a ResTP

association with connection-oriented connection management also cycles through a series of states during its lifetime, including LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, and CLOSED. In addition, ResTP does not modify the meanings of these states. The only difference between the TCP 3-way handshake and ResTP opportunistic connection establishment is the actions taken by the communicating entities in response to events that cause state changes. Listings 1 and 2 are used to implement the 3-way handshake and the opportunistic establishment on both sides of a connection, respectively. Although both TCP and ResTP are full-duplex, for simplicity, we only illustrate a unidirectional association.

```
switch (m_state){
  case CLOSED:
    Setup TCB;
    Send SYN;
    m_state = SYN_SENT;
    break;
  case SYN_SENT:
    if (receive SYN–ACK){
      m_state = ESTABLSIHED;
      Send ACK message;
      if (m_pendingDataAvailable)
        Send pending data;
      Notify connection success to application;
    }
    break;
  case LISTEN:
    if (receive SYN){
      Notify connection request to application;
      Send SYN–ACK;
      m_state = SYN_RCVD}
    break;
  case SYN_RCVD:
    if (receive ACK || receive data){
      Notify connection creation to application;
      m_state = ESTABLISHED;
      if (receive data)
        Send ACK for the data;  }
    break; }
```

Listing 1 3-way handshake

```
switch (m_state){
  case CLOSED:
    Setup TCB;
    Send SYN;
    if (m_pendingDataAvailable)
      Send pending data with SYN flag;
    m_state = SYN_SENT;
    break;
  case SYN_SENT:
    if (receive SYN–ACK || ACK of sent data){
      m_state = ESTABLSIHED;
      Send ACK message;
      if (m_pendingDataAvailable)
        Send pending data;
      Notify connection success to application;
    }
    break;
  case LISTEN:
    if (receive SYN || receive data with SYN
    flag){
      Notify connection request to application;
      Send SYN–ACK;
      if (receive data)
        Send ACK for the data;
      m_state = SYN_RCVD; }
    break;
  case SYN_RCVD:
    if (receive ACK || receive data){
      Notify connection creation to application;
      m_state = ESTABLISHED;
      if (receive data)
        Send ACK for the data;  }
    break; }
```

Listing 2 opportunistic

4 Simulation Results and Analysis

In this section, we present our simulation and analysis of two ResTP reliability modes: fully-reliable and unreliable datagram in comparison with TCP and UDP, respectively. The fully-reliable mode is configured with either the 3-way handshake or opportunistic connection establishment technique, while the unreliable datagram mode implements the connectionless flow management scheme.

4.1 Simulation Topology

Figure 6 depicts our simulation topology. At each edge of the topology is a node representing the data sender on one end and the receiver on the other end. The two endpoints communicate through a single bottleneck link where we introduce bit errors and extra latency. The bottleneck connects two network routers with drop-tail queues installed. Each of the access link that connects an endsystem to one of the routers has a negligible delay of 0.01 ms. Each link in the topology has a bandwidth of 5 Mb/s. Every data packet has an MTU size of 1446 bytes. Table 1 summarizes the parameters used in our simulations.

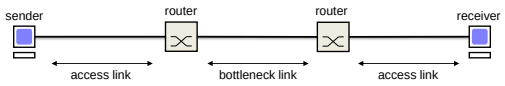


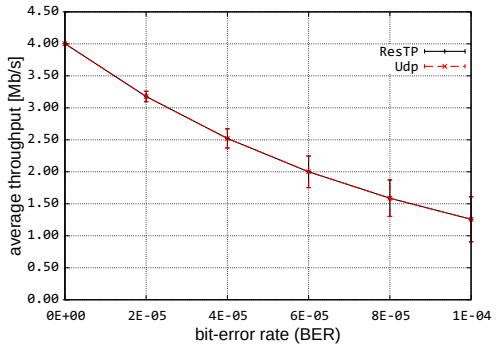
Figure 6 Simulation topology

Parameter	Values
Link bandwidth	5 Mb/s
Access link propagation delay	0.01 ms
Bottleneck link propagation delay	10 ms – 300 ms
Packet MTU size	1446 B
Error model	Rate Error Model
Error rate	10^{-7} – 10^{-4} BER
Application type	Bulk send and CBR
Simulation time	100 s
Queue size	BDP
Queue type	Drop-tail
ResTP connection timeout (fully-reliable)	3 s
TCP variant	NewReno

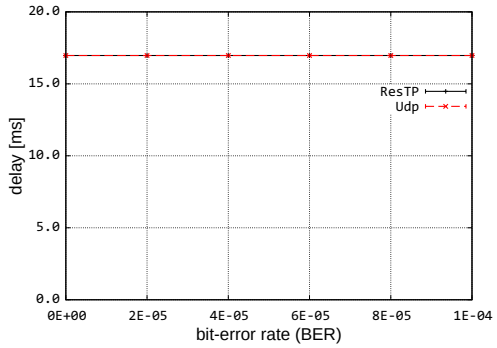
Table 1 Simulation parameters

4.2 ResTP Unreliable Datagram and UDP

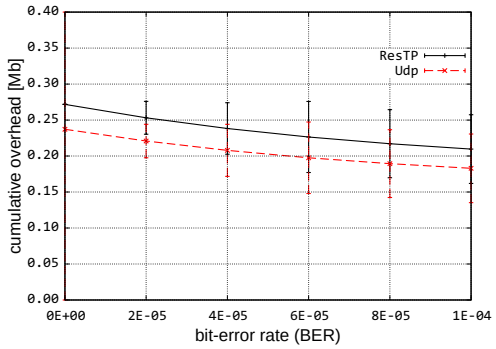
For the first set of simulations, we compare the ResTP unreliable datagram mode that implements connectionless connection management with UDP. In this scenario, the bottleneck link has a delay of 10 ms. The sender generates traffic at a constant rate of 4 Mb/s. We introduce



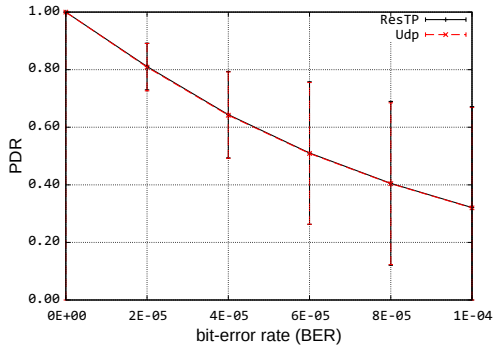
(a) Throughput



(b) Delay



(c) Overhead



(d) PDR

Figure 7 ResTP unreliable datagram mode vs. UDP

bit errors into the bottleneck channel with the probability ranging from 0 to 10^{-4}. Each simulation has a duration of 100 seconds and is replicated 20 times. We plot the average throughput, average delay, average overhead, and packet delivery ratio (PDR) of the two protocols as the error rate increases. As shown in Figure 7, this configuration of ResTP performs comparable to UDP. When there are no errors, both ResTP and UDP are able to deliver all generated traffic successfully to the other end (Figure 7d) and achieve a throughput close to 4 Mb/s. However, as the number of packet corruptions increases, the number of packets arrived at the receiver side decreases, causing the throughput to drop, since both ResTP unreliable datagram mode and UDP have no mechanisms to recover from errors. ResTP has a slightly larger overhead than UDP (Figure 7c). This is due to the extra first four bytes in the ResTP header that cover the version, header length, and all the flags for connection management, flow, and error control. As we explained in the previous section, these fields are needed for the ResTP entities to communicate their operating modes because ResTP provides multiple sets of services. However, this difference in the overhead between the two protocols is very small, given that ResTP header is already optimized for this configuration to exclude unnecessary fields.

4.3 ResTP Fully Reliable and TCP

For the second set of simulations, we compare the ResTP fully-reliable connection mode that implements the TCP 3-way handshake connection management with TCP NewReno. This configuration of ResTP implements ARQ error correction with positive ACKs to be comparable with TCP. The traffic generator uses `ns3::BulkSendApplication` to place a large amount of traffic into the network, trying to fill up the channel bandwidth. We set the buffer size at each router to be the bandwidth-×-delay product (BDP). With our setting, the only source of packet losses is corruption when errors are introduced into the bottleneck link.

BER	ResTP Throughput (Mb/s)	TCP Throughput (Mb/s)
10^{-7}	4.927 ± 0.003	4.817 ± 0.025
10^{-6}	4.520 ± 0.017	2.153 ± 0.044
10^{-5}	1.515 ± 0.052	0.352 ± 0.012
10^{-4}	0.069 ± 0.022	0.008 ± 0.002

Table 2 Average throughput of ResTP and TCP vs. BER

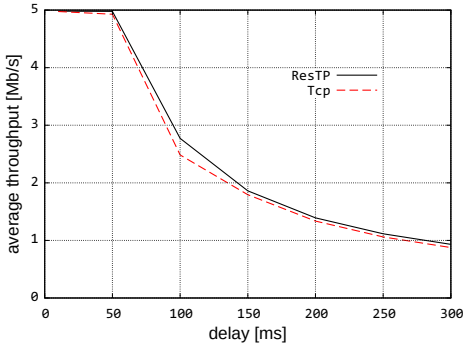


Figure 8 Average throughput of ResTP & TCP vs. delay

Table 2 presents the average throughput of ResTP and TCP

when the bit error rate (BER) varies from 10^{-7} to 10^{-4}. To increase the accuracy of our results, we repeat each simulation 20 times and calculate the 95% confidence interval. As shown in the table, the higher the error rate, the lower the throughput achieved by ResTP and TCP due to the increase in number of retransmissions. However, TCP, with its rigid intertwined error, flow, and congestion control together with the inability to distinguish between corruption and congestion-based data loss, suffers more drastically than ResTP. The congestion control algorithm in TCP gets invoked whenever a loss occurs, which unnecessarily reduces its sending rate. On the other hand, ResTP, as a configurable protocol that has each of its services implemented as a pluggable component, is able to disable its congestion control when operating in a lossy but uncongested channel. In this scenario, ResTP maintains its sending rate while trying to recover lost packets. At 10^{-6} BER, ResTP achieves twice the average throughput of TCP, and when the BER reaches 10^{-4}, ResTP throughput is more than eight times higher than TCP.

In the second scenario (Figure 8), we introduce extra latency into the bottleneck channel instead of BER. The delay is varied from 10 ms to 300 ms to cover different delays for various network environments. The average throughputs of both ResTP and TCP degrade with the increasing latency, and the two protocols perform similar in this case.

4.4 Opportunistic and 3-Way Handshake

In the last set of simulations, we compare the opportunistic and conventional 3-way handshake connection establishment mechanisms that can be implemented in the ResTP fully-reliable mode, particularly when encountering SYN loss. Figure 9 plots the average throughput of both algorithms when the bottleneck delay increases from 10 ms to 300 ms. Overall, the opportunistic approach performs better than the 3-way handshake.

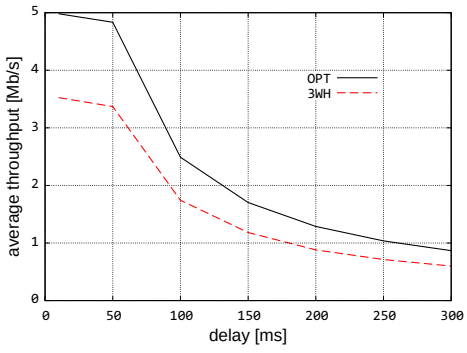


Figure 9 Average throughput of OPT and 3WH vs. delay

We examine the instantaneous throughputs when the delay is 100 ms to take a closer look at the reason behind the greater performance of the opportunistic mechanism. As shown in Figure 10, due to the SYN drop, ResTP with the 3-way handshake is only able to complete its connection initialization when the SYN message is retransmitted more than 3 seconds after the start of the simulation. On the other hand, with the opportunistic algorithm, the protocol can finish its initial setup upon the arrival of the data packet that is piggybacked with the SYN. Since this data segment

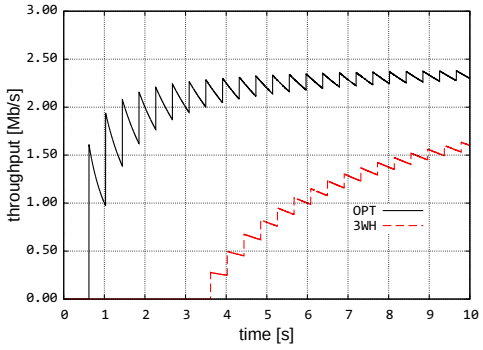


Figure 10 Inst. throughput of OPT and 3WH (SYN drop)

also carries a SYN flag, the receiving end views the message as a regular SYN and immediately responds with a SYN-ACK, which allows additional data to be transmitted during the first 3 seconds of the simulation. This behavior is especially beneficial to short flows that may have a strict bound in their completion time. The benefit is even more significant when these flows must traverse through long-delay channels.

5 Conclusion

In this paper, we discuss the design and implementation of alternative connection management schemes for our resilient transport-layer protocol, ResTP. We study the performance of these algorithms together with the two reliability modes defined in ResTP under various network scenarios through multiple simulations in ns-3. Our results show that ResTP unreliable datagram mode with the connectionless flow management technique performs comparably to UDP. The optimized header is one of the key factors that constitute to this achievement. ResTP fully-reliable mode that implements the 3-way handshake connection establishment and ARQ error control with positive ACK outperforms TCP, particularly in the presence of segment loss. The high configurability allows ResTP to only select those mechanisms that are necessary for a particular scenario. To efficiently handle corruption-based losses, ResTP can disable its congestion control module and allow the data sender to maintain its sending rate while recovering from losses. Finally, we show that the opportunistic flow establishment outperforms the 3-way handshake when facing SYN loss and high latency. This approach is more favorable for short traffic flows that have a strict delay bound.

As we continue our design, implementation, and optimization of ResTP, we will extend our study of the protocol under different application and network types. We plan to analyze the performance of the other three reliability modes of ResTP, including its nearly-reliable mode with custody transfer, quasi-reliable with FEC, and unreliable connection mode. We will also study the multipath mode of ResTP, which is designed to enhance the survivability of a network.

Acknowledgments

The authors would like to thank the members of the ResiliNets group for discussions which led to this work. This research was supported in part by NSF grant CNS-1219028 (Resilient Network Design for Massive Failures and Attacks).

6 Literature

[1] J. Postel, "Transmission Control Protocol." RFC 793 (Standard), Sept. 1981. Updated by RFCs 1122, 3168.

[2] J. Postel, "User Datagram Protocol." RFC 768 (Standard), Aug. 1980.

[3] R. Krishnan, J. P. G. Sterbenz, W. M. Eddy, C. Partridge, and M. Allman, "Explicit transport error notification (ETEN) for error-prone wireless and satellite networks," *Computer Networks*, vol. 46, no. 3, pp. 343–362, 2004.

[4] A. Bhatia, J. Sterbenz, and G. M. Parulkar, "Comments on Proposed Transport Protocols," 1988.

[5] D. Feldmeier, "An overview of the TP++ transport protocol project," in *High Performance Networks: Frontiers and Experience* (A. N. Tantawy, ed.), vol. 238 of *Kluwer International Series in Engineering and Computer Science*, ch. 8, Boston, MA, USA: Kluwer Academic Publishers, 1993.

[6] P. G. Bridges, G. T. Wong, M. Hiltunen, R. D. Schlichting, and M. J. Barrick, "A Configurable and Extensible Transport Protocol," *IEEE/ACM Trans. Netw.*, vol. 15, pp. 1254–1265, Dec. 2007.

[7] J. P. G. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks*, vol. 54, no. 8, pp. 1245–1265, 2010.

[8] "The ns-3 Network Simulator." http://www.nsnam.org, July 2009.

[9] J. F. Kurose and K. W. Ross, *Computer networking: a top-down approach*. Pearson Education, Inc, 2017.

[10] R. W. Watson and S. A. Mamrak, "Gaining efficiency in transport services by appropriate design and implementation choices," *ACM Transactions on Computer Systems*, vol. 5, pp. 97–120, May 1987.

[11] W. A. Doeringer, D. Dykeman, M. Kaiserswerth, B. W. Meister, H. Rudin, and R. Williamson, "A survey of light-weight transport protocols for high-speed networks," *IEEE Transactions on Communications*, vol. 38, pp. 2025–2039, Nov 1990.

[12] A. Baratz, J. Gray, P. Green, J. Jaffe, and D. Pozefsky, "SNA Networks of Small Systems," *IEEE Journal on Selected Areas in Communications*, vol. 3, pp. 416–426, May 1985.

[13] J. Martin, K. K. Chapman, *et al.*, *SNA: IBM's networking solution*. Prentice-Hall, Inc., 1987.

[14] T. J. Routt, "Distributed SNA: A network architecture gets on track," in *Systems network architecture*, pp. 167–180, IEEE Press, 1992.

[15] R. Sundstrom, J. Staton, G. Schultz, M. Hess, and G. Deaton, "SNA directions—a 1985 perspective," in *AFIPS Conference Proceedings; vol. 55 1986 National Computer Conference*, pp. 537–551, AFIPS Press, 1986.

[16] G. Chesson, "Datakit software architecture," *Proceedings ICC, pages*, vol. 20, pp. 1–20, 1979.

[17] A. Fraser, "Towards a universal data transport system," *IEEE Journal on Selected Areas in Communications (JSAC)*, 1983.

[18] A. G. Fraser and W. T. Marshall, "Data transport in a byte stream network," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 1020–1033, Sep 1989.

[19] R. W. Watson, "The Delta-t Transport Protocol: Features and Experience," in *Local Computer Networks, 1989., Proceedings 14th Conference on*, pp. 399–407, October 1989.

[20] D. Cheriton and D. Cheriton, "VMTP: A transport protocol for the next generation of communication systems," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 16, pp. 406–415, Sept. 1986.

[21] J. P. Rohrer, E. Perrins, and J. P. G. Sterbenz, "End-to-end disruption-tolerant transport protocol issues and design for airborne telemetry networks," in *Proceedings of the International Telemetering Conference (ITC)*, (San Diego, CA), October 2008.

[22] J. P. Rohrer, A. Jabbar, E. K. Çetinkaya, E. Perrins, and J. P. Sterbenz, "Highly-dynamic cross-layered aeronautical network architecture," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, pp. 2742–2765, October 2011.

[23] J. P. Rohrer, R. Naidu, and J. P. G. Sterbenz, "Multipath at the Transport Layer: An End-to-End Resilience Mechanism," in *Proceedings of the IEEE/IFIP International Workshop on Reliable Networks Design and Modeling (RNDM)*, (St. Petersburg, Russia), pp. 1–7, October 2009.

[24] T. A. N. Nguyen, J. P. Rohrer, and J. P. G. Sterbenz, "ResTP–A Transport Protocol for FI Resilience," in *Proceedings of the 10th International Conference on Future Internet Technologies (CFI)*, June 2015.

[25] K. Scott and S. Burleigh, "Bundle Protocol Specification." RFC 5050 (Experimental), Nov. 2007.

[26] M. A. Hiltunen, R. D. Schlichting, X. Han, M. M. Cardozo, and R. Das, "Real-time dependable channels: Customizing QoS attributes for distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 6, pp. 600–612, 1999.