Title

TCP-RTA: Real-Time Topology Adaptiveness for Congestion Control in TCP

Authors

Srinivasan, Ramesh
Garcia-Luna-Aceves, J.J.

Publication Date

2022-10-01

Data Availability

The data associated with this publication are within the manuscript.

Peer reviewed

TCP-RTA: Real-time Topology Adaptive Congestion Control Strategy in TCP

Ramesh Srinivasan[1][0000−0002−2680−7254]
J. J. Garcia-Luna-Aceves[1][0000−0001−9914−6031]

University of California, Santa Cruz, CA 95064, USA

Abstract. The congestion-control mechanisms currently implemented in different variants of the Transmission Control Protocol (TCP) do not account for the possibility that an inherent topology change is the cause of changes in the perceived end-to-end round-trip time (RTT) in a TCP session, rather than network congestion. This results in low throughput and inefficient use of the available bandwidth, when a topology change is the real cause of the change in RTT. We introduce TCP-RTA (TCP Real-time Topology Adaptiveness), a TCP variant that dynamically detects a topology change and in real-time adapts to an appropriate congestion-control strategy in order to maximize the effective use of the total available bandwidth. Simulation results indicate a throughput increase of more than 35% in scenarios involving dynamic topology changes in the midst of a TCP session.

Keywords: TCP · Real-time · Congestion-Control.

1 Introduction

The congestion-control mechanisms that are used in the Transmission Control Protocol (TCP) today are not able to detect changes in the underlying topology that lead to drastic changes in the round-trip time (RTT) experienced in a TCP session. Instead, TCP senders interpret such changes as the presence of congestion. Furthermore, current TCP implementations reply on a specific congestion-control strategy that is fixed for the duration of a TCP session. This is rapidly becoming a major limitation of TCP in today's Internet, because of two key factors. First, the proliferation of very different types of transmission media that have disparate bandwidth-delay products and reliability renders the use of a single congestion-control strategy that is unaware of the impact of the underlying topology on the delays of TCP sessions highly ineffective. Second, end user applications and deployment scenarios including Anglova [15] require continuous availability of services, service providers need to attain the most efficient use of the available bandwidth over wired or wireless links, and more and more end users are mobile. Hence, the original approach used in TCP of interpreting increases in delay as the ensuing of congestion must be revisited to account for the fact that a given TCP session may use different types of transmission media as end users move and different transmission media are used as a result.

The key contribution of this paper is the introduction of a new approach to congestion-control in TCP that uses different congestion-control algorithms depending on the perceived use of different underlying transmission media resulting from changes in the measured RTT within ongoing TCP sessions. The approach uses a congestion-control algorithm that is best suited for a given range of RTT values and switches among different algorithms as needed. This is particularly relevant for the support of TCP sessions involving end-devices that are mobile during the midst of an ongoing TCP session. In addition, the proposed approach to congestion control in TCP is particularly attractive for future deployments of 5G networks and beyond, because it easily accommodates the use of heterogeneous transmission media.

Section 2 discusses related work. As our survey of prior variants of TCP reveals, TCP variants in the past have rely on a single congestion-control algorithm. The closest approach to our work is D-TCP (Dynamic TCP) [22], wherein the bandwidth-delay product is dynamically computed and a congestion metric derived off this computation. This is then used to determine the response of the congestion control algorithm to increase/decrease the congestion window during the RTT update and loss detection. However, this is done within a single algorithm and it is not very robust.

Section 3 presents the approach and architecture of TCP-RTA (TCP with Real-Time Topology Adaptiveness). TCP-RTA is a new TCP variant with a comprehensive set of enhancements, specific for dynamically detecting topology changes and according adapting to an appropriate congestion control strategy.

Based on various studies of observed RTT for various underlying network topologies [29], we categorize the initial starting topology of a new TCP session being initiated based on the observed RTT values during the initial three-way handshake. Thereafter the RTT values are monitored and anytime three consecutive RTTs change and in the range of a different underlying topology, TCP-RTA dynamically enables a change to the corresponding specific congestion-control strategy for the perceived topology being perceived. This ensures that the ongoing TCP session has the best congestion-control strategy in place for the topology being used by the connection. TCP-RTA invokes separate congestion-control algorithms for each of the specific topology being perceived through RTT measurements.

Section 4 describes the results of simulations conducted with TCP-RTA and with other deployed TCP versions including TCP Cubic and Section 5 outlines and compares the results observed. Section 6 concludes the paper.

2 Related Work

We provide a survey of the various versions of TCP that are deployed currently, and the way they handle the introduction of wireless links. A comparative study of the actual approaches used in the different TCP implementations can be found in [11].

There are several TCP implementations [26], including Tahoe [20], Reno [19], New-Reno [13], TCP-SACK [24], TCP-Vegas [6], TCP-Jersey [31], TCP-DCR [5], and TCP Santa Cruz [25] that address different short-comings of TCP.

A number of TCP variants have been proposed for high-speed network requirements, including FAST [16], HSTCP [12], STCP [23], CUBIC [17], SQRT TCP [18], TCP-Westwood [10], BIC TCP [32] Binary Increase Congestion control, TCP-Illinois [21], TCP-Hybla [8], YeAH-TCP [1], Compound TCP (CTCP) [30], and BBR [9].

Mobile end-devices may undergo underlying topology changes during the course of a TCP session that may be simply due to physical movement. The resulting changes observed in end-end packet round-trip-time (RTT) would be interpreted incorrectly as congestion in the network by existing TCP implementations.

For completeness, we note that RFC1185 and RFC1123 were among the first initiatives to enable TCP extensions for high-speed networks.

The approaches proposed to improve TCP performance over networks with wireless links can be divided into two major categories, namely those that work at the transport level, and those that work at the link level. Transport-level proposals include Explicit Bad State Notification (EBSN) [3], Freeze-TCP [14], Indirect-TCP (I-TCP) [2], Snoop [4], fast-retransmission [7]

Snoop [4] is a well-known link level proposal. In this scheme, the base station sniffs the link interface for any TCP segments destined for the mobile host, and buffers them if buffer space is available. Segments are forwarded to the mobile host only if the base station deems it necessary.

In WTCP [28] the base station is involved in the TCP connection. WTCP [28] requires no modification to the TCP code that runs in the mobile host or the fixed host. The base station locally retransmits lost segments based on duplicate acknowledgment or timeouts. In the case of a timeout, a potentially wasteful wireless transmission is avoided and interference with other channels is reduced by quickly reducing the transmission window. WTCP [28] also hides the wireless link errors from the source by subtracting the residence time of the segment at the WTCP [28] buffer from the RTT value computed at the source, thus the RTT computation excludes wireless link layer retransmission delays.

Prior work more closely related to the work presented in this paper includes Dynamic TCP (D-TCP) [22], which implements a congestion-control algorithm in which the bandwidth-delay product is dynamically computed and a congestion metric is derived off this computation. This is then used to determine the response of the congestion-control algorithm to increase or decrease the CWND during the RTT update and loss detection. A single parameter is dynamically modified and the underlying congestion-control algorithm is the same for all scenarios and through the life-cycle of a TCP session.

3 TCP-RTA APPROACH AND ARCHITECTURE

3.1 TCP-RTA Approach

There are several congestion-control strategies customized for specific environments. Examples of these strategies include TCP Hybla [8] for satellite links, HSTCP [12] for networks with a large bandwidth-delay product along with low-latency, as well as some generic TCP variants like TCP NewReno [13], among others.

We propose a new mechanism that leverages apriori categorization of values of some TCP parameters (e.g., RTT) as corresponding to some particular underlying topology characteristics. This information is used to help identify the actual environment encountered by a TCP session at a given point during the course of the given session. An example would be a significant consistent increase in RTT, which wouldbe a strong indicator of a change of the environment from a path being used in a terrestrial network to a path involving a satellite link. The proposed mechanism would respond as follows: On detecting the significant and consistent RTT increase, a switch of the congestion-control algorithm is enacted from the one used in TCP NewReno [13] to the one used in TCP-Hybla's [8].

Table 1. Definitions

Variable	Definition
RTT	Round Trip Time (implies end-end)
RTT-Current	RTT for the most recent segment
RTT-Prev	RTT for the segment prior to the last segment (prior segment)
RTT-Prev-Prev	RTT for the segment prior to the prior segment

3.2 TCP-RTA Architecture

A TCP session starts with a default configuration, including a congestion-control strategy that we have chosen to be that of TCP NewReno. However, this default TCP configuration can be any variant of TCP that is apt for the environment of the TCP session when it is first established. One of the proposed enhancements for future work is the inclusion of negotiation and convergence on the initial default configuration as part of the initial three-way handshake used in connection establishment in TCP. Comparing the observed RTT during the initial three-way handshake with the corresponding default RTT thresholds for each environment, the nature of the underlying topology is estimated and ais used to configure the default initial configuration of the TCP session. For our default TCP NewReno configuration, the RTT-threshold is accordingly initialized to 300 ms. TCP-RTA keeps track of the last three observed RTT at any point in time: RTT-Current, RTT-Prev and RTT-Prev-Prev. The salient steps in our proposed approach are the following:

1. The three variables in Table 1 are initialized at the start of a session to the default-RTT-threshold (300ms in our environment).
2. After receipt of an acknowledgement and the corresponding immediate computation of the observed RTT (RTT-new), the following variables are updated:
   i RTT-Prev-Prev with RTT-Prev
   ii RTT-Prev with RTT-Current
   iii RTT-Current with RTT-new
3. If the three consecutive observed values of RTT are all above 800 ms, an inference is made that there must have been an underlying topology change. Based on some of the observed RTT times for TCP sessions going over a satellite link [8], it is assumed that the path now involves a satellite link. Hence, a change is made in the congestion-control strategy that is more apt for the newly observed environment that includes a path witha significantly larger delay (typically attributed to a satellite link), namely TCP Hybla [8].
4. If only one or two of the observed RTT values are above the 800 ms threshold and subsequently the RTT comes back to prior regular values, then these transients are ignored and the congestion-control strategy is left unchanged.
5. The above steps are repeated until the end of the TCP session, with an additional check happening after every update to the observed RTT. If the last three observed RTT values are all below RTT-threshold for a non-satellite link, which we have set as 300 ms based on reported observations in [8], the algorithm reverts back the congestion-control strategy of TCP NewReno.

Algorithm 1 depicts the control flow of the newly proposed dynamic TCP-RTA and its congestion-control strategy. The underlying premise is that, if there is a distinct change suddenly observed in the RTT and that change is consistently maintained for at least three consecutive segments back to back without any packet loss, then the cause of such a change is assumed to be an underlying topology change rather than sporadic congestion in the network.

Since TCP does not know whether a delayed ACK is caused by a congestion experienced by a segment or possibly a topology change, it waits for a small number of additional ACKs to be received. It is assumed that, if there is just a temporary increase in RTT, there will be typically one or two delayed ACKs at most before either the RTT returns to prior normal values or it increases more and possibly ends in a timeout and a packet drop. The underlying premise is that any network congestion scenario is not a stable condition that would quickly either return to normalcy or become worse. So, if there are three or more ACKs consistently delayed by similar value and received in a row, then it can be taken as a strong indication that the segments are most probably using a new path (topology) with a different (increased or decreased) RTT.

We ensure the hand-off happens seamlessly from the current congestion-control strategy to the appropriate target congestion-control strategy for the newly identified topology to which the network has transition to. This is particularly relevant for mobile end-nodes that are ubiquitous with the proliferation of 5G technology.

Algorithm 1 TCP-RTA-Overview

```
1: Def_RTT_TCP_NEWRENO = 500
2: Def_RTT_TCP_HYBLA = 800
3: m_adaptiveAlgProg = TCP_NEWRENO
4: RTT-Current = Def_RTT_TCP_NEWRENO
5: RTT-Prev = Def_RTT_TCP_NEWRENO
6: RTT-Prev-Prev = Def_RTT_TCP_NEWRENO
   do
7: if (m_adaptiveAlg == TCP_NEWRENO) then
8:    if ((RTT-Current > Def_RTT_TCP_HYBLA) &&
       (RTT-Prev > Def_RTT_TCP_HYBLA) &&
       (RTT-Prev-Prev > Def_RTT_TCP_HYBLA)) then
9:      m_adaptiveAlg = TCP_HYBLA;
10:   end if
11: else if (m_adaptiveAlg == TCP_HYBLA) then
12:   if ((RTT-Current < Def_RTT_TCP_NEWRENO) &&
       (RTT-Prev > Def_RTT_TCP_NEWRENO) &&
       (RTT-Prev-Prev > Def_RTT_TCP_NEWRENO)) then
13:     m_adaptiveAlg = TCP_NEWRENO;
14:   end if
15: end if
16: wait till next ACK recd;
17: RTT-new = ComputeNewRTT();
18: RTT-Prev-Prev = RTT-Prev;
19: RTT-Prev = RTT-Current;
20: RTT-Current = RTT-new;
   while (TCP Session is still active)
```

The TCP-RTA approach also ensures that the adaptive congestion-control strategy does not respond to any sporadic one-off drastic behaviour in the TCP parameters in any significant manner. Another significant aspect of the proposed approach is that one-off sporadic changes are not even passed onto the TCP congestion-control mechanism, effectively acting like a low-pass filter preventing these exceptions from impacting the parameters impacting the throughput and performance of the TCP session.

TCP-RTA incorporates the following additional list of enhancements:

1. **Slow Start Enhancement** in TCP NewReno, cwnd is increased by one segment per acknowledgment. In TCP-RTA, cwnd is changed to SegAcked * Segment size. (similar to Cubic [17])
2. **Congestion Avoidance Enhancement** In TCP NewReno, cwnd is increased by (1/cwnd) In TCP-RTA, the following changes are introduced: In congestion avoidance phase, the number of bytes that have been ACKed at the TCP sender side are stored in a 'bytes_acked' variable in the TCP control block. When 'bytes_acked' becomes greater than or equal to the value

Algorithm 2 TCP-RTA::SlowStart

```
1: input Ptr_SocketState, segmentsAcked
2: if (segmentsAcked ≥ 1 && m_adaptiveAlg == TCP_NEWRENO) then
3:    sndCwnd = tcb→ m_cWnd;
4:    tcb→m_cWnd = min((sndCwnd+(segmentsAcked*tcb
      →m_segmentSize)),tcb →m_ssThresh);
5:    return segmentsAcked-((tcb →m_cWnd-sndCwnd) / tcb
      →m_segmentSize);
6: else if (segmentsAcked ≥ 1 && m_adaptiveAlg == TCP_HYBLA) then
7:    /* slow start
8:    INC = 2^ρ - 1 */
9:    increment = pow(2, m_ρ) - 1.0;
10:   incr = increment*tcb→m_segmentSize;
11:   tcb→m_cWnd = min (tcb→m_cWnd + incr, tcb→m_ssThresh);
12:   return segmentsAcked - 1;
13: end if
14: return 0;
```

of the cwnd, 'bytes_acked' is reduced by the value of cwnd. Next, cwnd is incremented by a full-sized segment (SMSS). (Similar to Linux Reno [19] implementation)

3. On Fast restransmit, we update ssthresh to half of current cwnd: ssthresh = bytesinflight/2; In order to recover faster, it is enhanced as follows: sstresh = (bytesInFlight * 2) /3.
4. Default boost of a factor of 10 (constant) of the Bandwidth*Delay product while switching from LAN to Satellite and vice versa.

TCP-RTA does not change any of the fairness with respect to other TCP co-existing as the underlying congestion control strategy adopted by us is that of TCP-Hybla, when the topology change is detected through a consistent increase in RTT. The fairness of TCP-Hybla and earlier that of TCP NewReno has been already established and proven and thus its applicable here as well. Even in the transition from congestion control strategies from TCP-Hybla to TCP NewReno, the only change is our non-responsiveness to transients and that too for only three segments. Thus fairness is guaranteed.

4 Testing and Simulation

We evaluated the performance of TCP-RTA using the ns-3 discrete-event simulator. The ns-3 simulator provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. The TCP implementation was modified to support the new proposed protocol including several of the enhancements listed earlier.

Numerous scenarios and options were tried out to validate the gains and benefits of the proposed approach. After our analysis of the various findings,

Algorithm 3 TCP-RTA::CongestionAvoidance

```
1: input‘ Ptr_SocketState, segmentsAcked
2: if (segmentsAcked > 0 && m_adaptiveAlg == TCP_HYBLA) then
3:    INC = ρ^2 / W
4:    segCwnd = tcb → GetCwndInSegments ();
5:    increment = std::pow (m_ρ, 2) / static_cast<double> (segCwnd);
6:    m_cWndCnt += increment;
7:    segmentsAcked -= 1;
8: end if
9: if (segmentsAcked > 0 && m_adaptiveAlg == TCP_NEWRENO) then
10:   if (m_adaptiveAlgProg ≠ ALOG_INPROGRESS) then
11:      w = tcb → m_cWnd / tcb → m_segmentSize;
12:      if (w == 0) then
13:         w = 1;
14:      end if
15:      if (m_cWndCnt ≥ w) then
16:         m_cWndCnt = 0;
17:         tcb → m_cWnd += tcb → m_segmentSize;
18:      end if
19:      m_cWndCnt += segmentsAcked;
20:      if (m_cWndCnt ≥ w) then
21:         delta = m_cWndCnt / w;
22:         m_cWndCnt -= delta * w;
23:         tcb → m_cWnd += delta * tcb → m_segmentSize;
24:      end if
25:   end if
26: else
27:   m_adaptiveAlgProgCnt–;
28:   tcb → m_cWnd = m_bd / tcb → m_segmentSize;
29: end if
30: if (m_cWndCnt ≥ 1.0 && m_adaptiveAlg == TCP_HYBLA) then
31:   inc = m_cWndCnt;
32:   m_cWndCnt -= inc;
33:   if (m_adaptiveAlgProg ≠ ALOG_INPROGRESS) then
34:      tcb → m_cWnd += inc * tcb → m_segmentSize;
35:   end if
36: else
37:   tcb → m_cWnd = m_bd / tcb → m_segmentSize;
38: end if
```

wedecided to use TCP NewReno and TCP-Hybla to simulate and study the behaviour for the significantly impactful change of a TCP session going through a LAN in a home office or a corporate network to a data path involving a satellite for wireless, which induces a highly significant additional delay in the observed RTT. We observed the behaviour during the initial slow-start phase of the TCP-session as well as in the congestion avoidance phase. The throughput was observed across the above phases and during the transition of the "underlying topology". Consistently we have seen that TCP-RTA outperforms the others significantly in every one of the phases and across all topology transitions.

As the results below indicate, there is a clear increase in the CWND size on transition for the satellite environment. We injected a significant delay from the period of time starting at 5 seconds and ending at 5 seconds to simulate a transition to a satellite back-haul and a subsequent transition back from it.

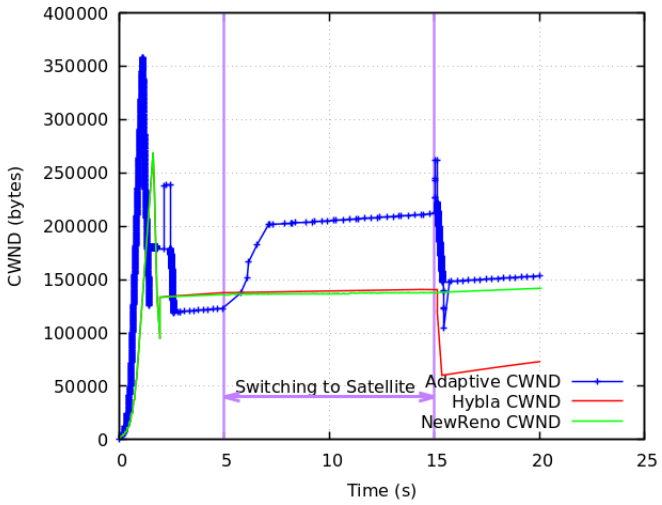

Fig. 1. CWND - TCP-RTA vs TCP Hybla vs TCP NewReno

5 Results

For our simulation studies, we earmarked RTT values, consistently observed in the range upwards of 800ms to denote an environment/topology with a satellite backhaul. Similarly, we earmarked RTT values, consistently observed in the range downwards of 800ms for one set of experiments and for others used a lower value 500ms, to denote a environment/topology without a satellite backhaul.

Various other RTT ranges could be added as needed to correspond to specific topology/environments, for which we have an specific TCP variants with their

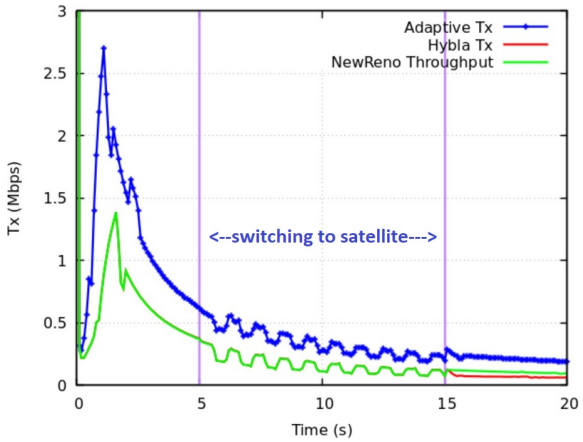

Fig. 2. TX - TCP-RTA vs TCP Hybla vs TCP NewReno

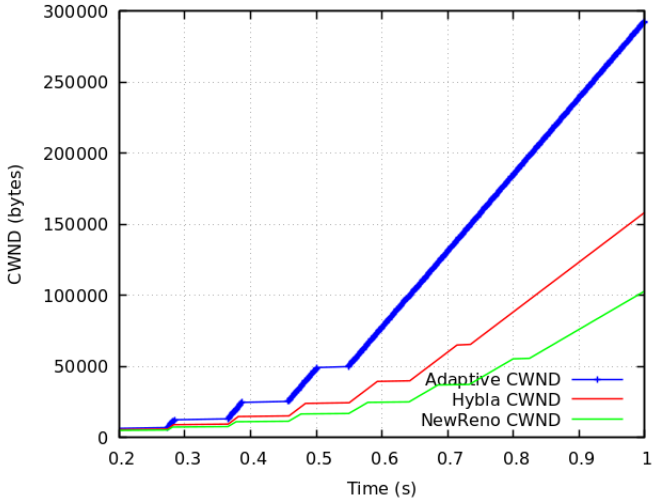

Fig. 3. CWND before transition to satellite - TCP-RTA vs TCP Hybla vs TCP NewReno

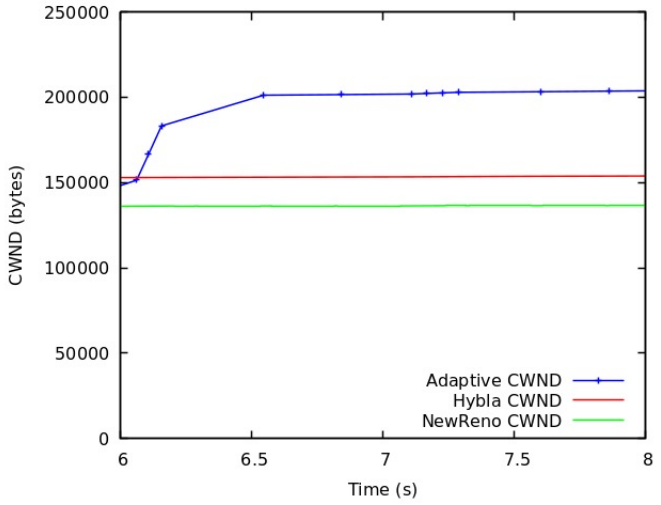

Fig. 4. CWND after transition to Satellite - TCP-RTA vs TCP Hybla vs TCP NewReno

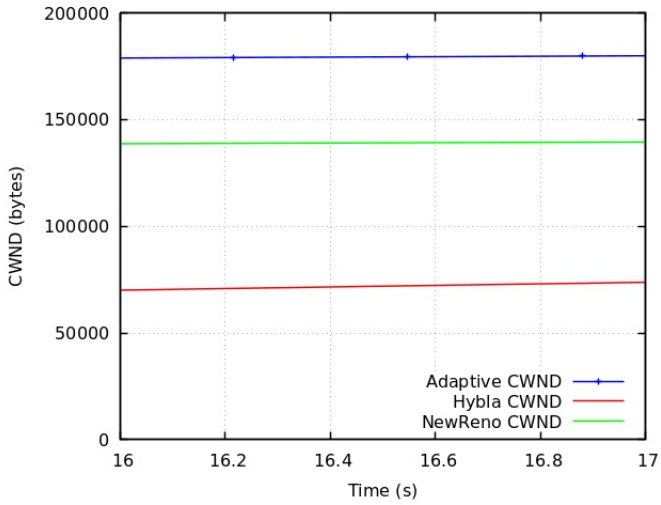

Fig. 5. CWND after transition from Satellite - TCP-RTA vs TCP Hybla vs TCP NewReno

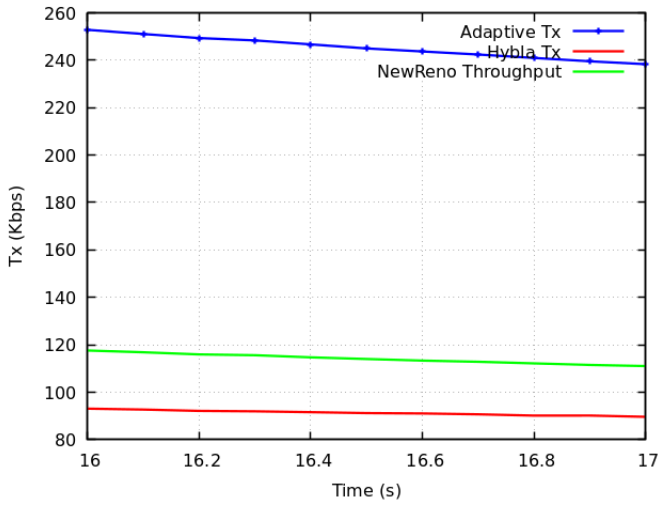

Fig. 6. TX after transition from Satellite - TCP-RTA vs TCP Hybla vs TCP NewReno

own congestion-control algorithm to provide the best optimal bandwidth usage and performance for that environment. It can be observed from the results that there is a significant boost in the congestion-window size when the transition to a satellite link happens, with its corresponding increase in the bandwidth-delay product, as can be seen in the Figure 4. Similarly, it can be observed in the usage of the network bandwidth close to 33% increase on the average across many simulation scenarios. However, for the very specific transitory phase to a satellite link in Figure 6, the throughput increase is almost twice that of TCP Hybla as well as TCP NewReno.

6 Conclusion and Future Work

TCP-RTA provides a framework to leverage the appropriate congestion-control strategy for a given dynamic scenario, thus ensuring we are at all times using the network in the most optimal efficient manner. The underlying design and approach used in TCP-RTA lends itself to seamlessly incorporating other specific scenarios and the dynamic transition to the corresponding congestion control strategy. Currently the mechanism used to detect a topology change is based on RTT variations. However, our approach does not preclude using other metrics or a combination of metrics to identify and determine any significant network change. Future work could involve using TCP-RTA in tandem with ECN[27] to clearly differentiate between longer RTTs due to real congestion versus due to topology change.

References

1. Baiocchi, A., Castellani, A.P., Vacirca, F.: Yeah-tcp: yet another highspeed tcp. In: Proc. PFLDnet. vol. 7, pp. 37–42 (2007)
2. Bakre, A.V., Badrinath, B.: Handoff and systems support for indirect tcp/ip. In: Symposium on Mobile and Location-Independent Computing. pp. 11–24 (1995)
3. Bakshi, B.S., Krishna, P., Vaidya, N.H., Pradhan, D.K.: Improving performance of tcp over wireless networks. In: Proceedings of 17th International Conference on Distributed Computing Systems. pp. 365–373. IEEE (1997)
4. Balakrishnan, H., Seshan, S., Katz, R.H.: Improving reliable transport and handoff performance in cellular wireless networks. Wireless Networks **1**(4), 469–481 (1995)
5. Bhandarkar, S., Sadry, N.E., Reddy, A.N., Vaidya, N.H.: Tcp-dcr: A novel protocol for tolerating wireless channel errors. IEEE Transactions on Mobile Computing **4**(5), 517–529 (2005)
6. Brakmo, L.S., O'Malley, S.W., Peterson, L.L.: Tcp vegas: New techniques for congestion detection and avoidance. In: Proceedings of the conference on Communications architectures, protocols and applications. pp. 24–35 (1994)
7. Caceres, R., Iftode, L.: Improving the performance of reliable transport protocols in mobile computing environments. IEEE journal on selected areas in communications **13**(5), 850–857 (1995)
8. Caini, C., Firrincieli, R.: Tcp hybla: a tcp enhancement for heterogeneous networks. International journal of satellite communications and networking **22**(5), 547–566 (2004)
9. Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H., Jacobson, V.: Bbr: congestion-based congestion control. Communications of the ACM **60**(2), 58–66 (2017)
10. Casetti, C., Gerla, M., Mascolo, S., Sanadidi, M.Y., Wang, R.: Tcp westwood: end-to-end congestion control for wired/wireless networks. Wireless Networks **8**(5), 467–479 (2002)
11. Fall, K., Floyd, S.: Simulation-based comparisons of tahoe, reno and sack tcp. ACM SIGCOMM Computer Communication Review **26**(3), 5–21 (1996)
12. Floyd, S.: Rfc3649: Highspeed tcp for large congestion windows (2003)
13. Floyd, S., Henderson, T., Gurtov, A.: Rfc3782: The newreno modification to tcp's fast recovery algorithm (2004)
14. Goff, T., Moronski, J., Phatak, D.S., Gupta, V.: Freeze-tcp: A true end-to-end tcp enhancement mechanism for mobile environments. In: Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064). vol. 3, pp. 1537–1545. IEEE (2000)
15. Group, N.I..R.T.: 5g network deployment scenarios. https://anglova.net/ (2022)
16. H., C.J.D.X.W.S.: Caltechfast tcp: From theory to experiments,. Low Engineering & Applied Science, (2005)
17. Ha, S., Rhee, I., Xu, L.: Cubic: a new tcp-friendly high-speed tcp variant. ACM SIGOPS operating systems review **42**(5), 64–74 (2008)
18. Hatano, T., Fukuhara, M., Shigeno, H., Okada, K.I.: Tcp-friendly sqrt tcp for high speed networks. Proceedings of APSITT (November 2003) pp. 455–460 (2003)
19. Jacobson, V.: "modified tcp congestion avoidance algorithm,". Technical Report, (1990)
20. Jacobson, V.: Congestion avoidance and control. ACM SIGCOMM computer communication review **18**(4), 314–329 (1988)

21. Jacobson, V., Braden, R., Borman, D.: Tcp extensions for high performance. Tech. rep., RFc 1323, May (1992)
22. Kanagarathinam, M.R., Singh, S., Sandeep, I., Roy, A., Saxena, N.: D-tcp: Dynamic tcp congestion control algorithm for next generation mobile networks. In: 2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC). pp. 1–6 (2018). https://doi.org/10.1109/CCNC.2018.8319185
23. Kelly, T.: Scalable tcp: Improving performance in highspeed wide area networks. ACM SIGCOMM computer communication Review **33**(2), 83–91 (2003)
24. M. Mathis, J. Mahdavi, S., Romanow., A.: "tcp selective acknowledgment options. RFc 2018, October (1996)
25. Parsa, C., Garcia-Luna-Aceves, J.J.: Improving tcp congestion control over internets with heterogeneous transmission media. In: Proceedings. Seventh International Conference on Network Protocols. pp. 213–221. IEEE (1999)
26. Polese, M., Chiariotti, F., Bonetto, E., Rigotto, F., Zanella, A., Zorzi, M.: A survey on recent advances in transport layer protocols. IEEE Communications Surveys and Tutorials **21**(4), 3584–3608 (2019). https://doi.org/10.1109/COMST.2019.2932905
27. Ramakrishnan, K.K., Floyd, S., Black, D.L.: The addition of explicit congestion notification (ecn) to ip. RFC **3168**, 1–63 (2001)
28. Ratnam, K., Matta, I.: Wtcp: An efficient mechanism for improving tcp performance over wireless links. In: Proceedings Third IEEE Symposium on Computers and Communications. ISCC'98.(Cat. No. 98EX166). pp. 74–78. IEEE (1998)
29. Sessini, P., Mahanti, A.: Observations on round-trip times of tcp connections (01 2006)
30. Tan, K., Song, J., Zhang, Q., Sridharan, M.: A compound tcp approach for high-speed and long distance networks. In: Proceedings-IEEE INFOCOM (2006)
31. Xu, K., Tian, Y., Ansari, N.: Tcp-jersey for wireless ip communications. IEEE Journal on selected areas in communications **22**(4), 747–756 (2004)
32. Xu, L., Harfoush, K., Rhee, I.: Binary increase congestion control (bic) for fast long-distance networks. In: IEEE INFOCOM 2004. vol. 4, pp. 2514–2524. IEEE (2004)