On the Miscollaboration of Congestion Control Mechanisms at the Transport and the Network Layers

Ageliki Tsioliaridou, Christos V. Samaras, Vassilis Tsaoussidis
Department of Electrical and Computer Engineering,
Democritus University of Thrace, Greece
E-mail: {atsiolia, csamaras, vtsaousi}@ee.duth.gr

Abstract

Many sophisticated mechanisms have been implemented at both the transport and the network layers in order to estimate network conditions and avoid overload of network links. We evaluate the congestion control and avoidance scheme from a cross-layer perspective. We conduct simulations to investigate issues of miscollaboration between the network and transport layers that have negative impact on application performance. We identify several such scenarios; we discuss the reasoning and the extent of miscollaboration, and the appropriate measures to resolve the problem. We conclude that the common practise for evaluating research proposals lacks the appropriate experimental diversity, granularity or perspective and hence new mechanisms may favor a particular set of targeted protocols but damage some others. As a result, a portion of Internet traffic flows is frequently favored at the cost of co-existing traffic flows.

1. Introduction

Due to the layered architecture of computer networks and the corresponding separation of functionality, researchers have not investigated thoroughly the interrelation among mechanisms of different layers. For example, when a new algorithm is proposed, it is common to compare it against similar algorithms at the same level. Although a major design principle for the layered architecture was the autonomy of each layer, in practice, mechanisms of different layers do interact, implicitly. This situation calls for cross-layer evaluation schemes.

We investigate the impact of collaboration among mechanisms at different layers. More precisely, we focus on the network and the transport layers, and investigate congestion management interaction of these two layers. Many mechanisms have been proposed at both layers in order to avoid or control congestion; however, specific research results have not yet been reported on the impact of mis-collaborative mechanisms of these two layers.

Typically, for the evaluation of queue management algorithms, researchers select an appropriate transport protocol to exploit the router's congestion notification and demonstrate its performance gains. However, there are other protocols in use that are not designed in a manner suitable to exploit the router's particular congestion control signals. One cannot exclude the possibility that a new, sophisticated mechanism may deteriorate the performance of existing transport protocols, which cannot take advantage of the changes.

Furthermore, protocols at the transport layer respond differently at the presence of a congestion signal, each following a more aggressive or conservative recovery strategy than others. However, router's operating parameters are fixed and typically do not differentiate incoming traffic based on the responsiveness of the transport protocol. In such a situation, it may be that some protocols are favored more than others at the presence of a specific queuing algorithm.

In this paper, we report initial results on the miscollaboration among the congestion control protocols and mechanisms. Through experiments, we discover cases where the mis-interaction among congestion control mechanisms results in system performance degradation. For example, we show that TCP Vegas cannot estimate well the available bandwidth when the link implements the Random Early Detection (RED) queue management algorithm. Furthermore, it might be that RED penalizes aggressive protocols more than it should and results in a situation where a conservative protocol outperforms an aggressive one in terms of goodput.

The rest of the paper is organized as follows. In Sections 2 and 3, we discuss the background on congestion avoidance and control at the transport and the network layers and present related work. Next, in

Section 4, we present a series of experiments, which demonstrate the lack of collaboration or further the miscollaboration among mechanisms of the two layers. In Section 5, we discuss briefly important cross-layer evaluation issues. Finally, in Section 6 we conclude the paper.

2. Background

2.1. Network Layer

In traditional algorithms for queue management, packets are dropped when the queue buffer becomes full. The Drop-Tail is a widely deployed dropping mechanism, which drops incoming packets when the queue memory resources are completely utilized. Although simplicity is a fundamental goal when designing a router algorithm, Active Queue Management (AQM) [2] was proposed presenting more complex or sophisticated algorithms in order to achieve better system performance.

One of the most popular active queue management algorithms is Random Early Detection (RED) [20]. Rather than waiting for a congested queue to overflow, arriving packets are dropped with some drop probability whenever the average queue length exceeds a minimum threshold, while they are all dropped when it exceeds a maximum threshold. An extension of RED is the Adaptive RED [13], which improves RED’s performance by adjusting the maximum threshold value to follow the network dynamics.

There exist many AQM algorithms, such as Stabilized RED [22], Random Exponential Marking [16], Blue [14], Stochastic Fair Blue [15] and Weighted RED [19]. They all attempt to interact implicitly with the transport protocols at the end-hosts in order to control congestion avoid collapses. See the corresponding links, for more details.

2.2. Transport Layer

Protocols at the transport layer are usually responsible for end-to-end error recovery, ensuring complete and reliable data transfer. Each of them implements a different algorithm to adjust its sending rate according to channel conditions.

Authors of [3] survey congestion control approaches, highlighting different mechanisms that control and avoid congestion, depending on the awareness of network conditions. Protocols of the first category attempt to identify channel conditions, blindly. TCP Tahoe, TCP Reno, TCP New-Reno [1] and TCP Sack [24] are some protocols of this category. Senders receive feedback implicitly through packet drops, and adjust their congestion window based on the AIMD algorithm [10].

In the second category, protocols aim to capture network conditions in a more accurate way based on measurements. For instance, TCP Vegas [4] relies on RTT measurements, while TCP-Westwood [7] on the rate of returning ACKs. There are also receiver-oriented protocols, such as TCP Real [5, 6] and TFRC [21], where receivers estimate the level of congestion and inform sender about it.

In the third category, protocols receive information from routers about the network conditions in an explicit way. In VCP [17] routers estimate their output link utilization and notify the senders accordingly. While authors in [8], propose a mechanism by which routers can calculate the fair-share of the available bandwidth and inform the senders to adjust their sending window accordingly.

3. Related Work

The issue of network- and transport-layer miscollaboration has not been studied extensively in the past. Some reports exist, however, where authors highlight such cases of miscollaboration.

Authors of [11], for example, investigate fairness of TCP in the presence of different scheduling algorithms. They show that the sending window size of TCP Vegas doesn’t get influenced by the drop probability of RED, rather than by its own mechanism, despite the fact that RED is proposed to achieve fairness among users of the same channel.

Furthermore, in [12] authors investigate the performance of TCP Sack protocol in the presence of RED queuing algorithm in heterogeneous environment of wired and wireless links. On one hand, TCP Sack is a sophisticated protocol that attempts to recover more quickly from a wireless error; on the other hand, RED is an algorithm which attempts to penalize aggressive protocols. As a result, TCP Sack senders get often penalized by the routers' queue management scheme. In other words, the sophistication of the one algorithm is cancelled by the sophistication of the other.

Finally, authors of [18] observed through experiments that when RED algorithm is deployed, it cannot guarantee fairness among TCP Reno and TCP Vegas flows that co-exist at the same link. More specifically, TCP Reno, due to its aggressive nature, outperforms TCP Vegas at the presence of RED.

4. Simulation Results

4.1. Evaluation Plan and Performance Metrics

In order to evaluate how well mechanisms at the transport and the network layers cooperate with each other, we have selected various transport protocols in our simulations such as TCP Tahoe, TCP Reno and

TCP Vegas, and monitored their interaction with network mechanisms for queue management and network congestion avoidance such as Drop-Tail and Random Early Detection (RED). We present cases of miscollaboration between various congestion control mechanisms, in which cases, network/end-system performance deteriorates as far as throughput, goodput and fairness are concerned.

We have conducted a number of experiments in various network conditions using the ns-2 network simulator [9]. The topology used is the dumbbell topology as shown in Fig. 1. A set of TCP senders (Source 1 through N) is connected with a set of TCP receivers (Sink 1 through N). The number of flows (N), the link's bandwidth at the senders (bw_3), at the receivers (bw_1), and the backbone link's capacity (bw_2) vary during the different simulated scenarios. The simulation time was typically set to 200 seconds.

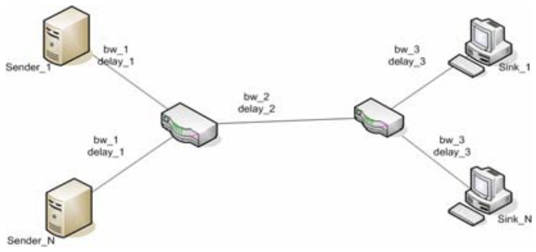

Figure 1. Simulation topology

We used traditional metrics for evaluating protocol efficiency. For example:

System Goodput and Throughput are used to measure the overall system efficiency in terms of bandwidth utilization and are given by the following formulas (where *n* is the number of flows):

$$System\ Goodput = \frac{\sum_{i=1}^{n} Original\ Data}{Transmission\ Time}$$

where *Original Data* is the number of bytes delivered from a sender to the corresponding receiver during their connection (*Transmission Time*), excluding the size of retransmitted packets of this flow and the overhead induced by packet headers.

$$System\ Throughput = \frac{\sum_{i=1}^{n} Total\ Data\ Sent}{Transmission\ Time}$$

where *Total Data Sent* is equal to the sum of original data, retransmitted data and packets' headers size (in Bytes).

4.2. Miscollaboration between sophisticated mechanisms at the transport and the network layers

Since the first implementations of transport and network protocols, several other mechanisms have been proposed and often implemented and deployed, in order to achieve better network/end-system performance. Given the fact that those mechanisms at both layers evolve in parallel and often in an independent way, they frequently end up being unaware of the presence of each other. Even worse, a sophisticated mechanism in the network can cancel an intelligent mechanism incorporated into a transport protocol or vice versa. To strengthen our claim, we show an example of how TCP Vegas, following a more or less sophisticated congestion control strategy, cannot cope that well with the increased functionality at the network introduced by RED (see Fig. 2, 3, 4 and 5).

TCP Vegas's congestion control mechanism is proactive, that is, it attempts to sense incipient congestion by observing changes in the throughput rate. TCP Vegas compares the measured throughput to its notion of expected throughput and adjusts its window accordingly. On the other hand, RED is an active queue management algorithm that detects incipient congestion and discards packets before queue overflows.

RED algorithm was designed to interact well with AIMD-based protocols, as these protocols infer congestion only after a packet has been lost. In particular, AIMD-based protocols react to such packet drops by reducing their congestion window. However, since TCP Vegas senders do not back off sharply at the presence of a packet drop and they also adjust their transmission rate much more smoothly, RED algorithm keeps dropping TCP Vegas packets in a high rate as those packets continue to take up much space in the RED queue buffers.

Our experiments below, explore the interaction between TCP Vegas and RED. In our simulations, bw_1, bw_2 and bw_3 are set to 10Mbps (Fig. 1), delay_1 and delay_2 are set to 5ms while the value of delay_3 is equal to 15ms; finally the queue buffer size is set to 50 packets. The experimental results show that due to incompatibility among TCP Vegas's and RED's mechanisms, the system goodput is degraded compared to the corresponding goodput when Drop-Tail is used (Fig. 3). We see in (Fig. 2) that throughput increase is justified only by packet retransmissions and does not correspond to extra data transmission. This situation is reflected by the significant deviation of the graphs, which depict the amount of packets that is dropped and retransmitted (Fig. 4). Finally, Fig. 5 shows how the queuing algorithms affects TCP Vegas packet drop rate as contention increases.

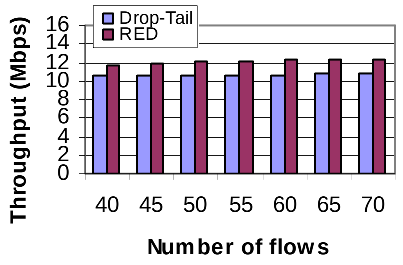

Figure 2. System throughput

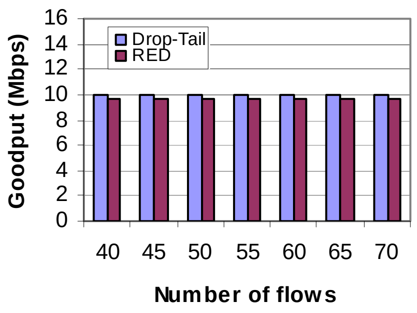

Figure 3. System goodput

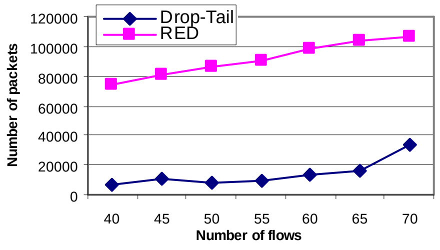

Figure 4. Total number of retransmitted and dropped packets

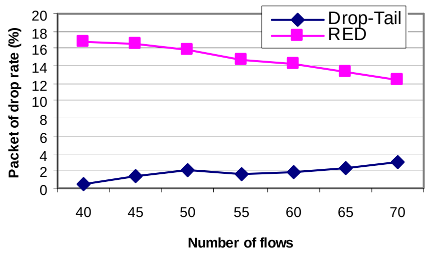

Figure 5. Packet drop rate

4.3. New mechanisms may not be interpreted by existing protocols

Miscollaboration can occur when the functionality of a relatively new mechanism cannot be well interpreted by existing mechanisms that rely on a different design philosophy. To reinforce our point by being more specific, we evaluate the performance of a conservative transport protocol, such as TCP Tahoe, when a sophisticated active queue management algorithm (namely, RED) is present at the network. TCP Tahoe assumes a single packet drop as congestion indication and backs off sharply by shrinking its congestion window. By that time when TCP Tahoe was designed, avoiding congestion collapses were of high importance, and packet drops at the routers occurred when there was no more buffer space available at a queue. On the other hand, RED algorithm does not drop packets just because its buffer is out of space but rather it drops each incoming packet with a certain probability, which depends on the average queue size. Such network functionality cannot be interpreted by TCP Tahoe and results in poor performance even when network bandwidth is available.

Next, we present a scenario of high contention where 100 TCP Tahoe flows pass through a 50Mbps bottleneck link (Fig. 1). The edge links' bandwidth is set to 1Mbps and the propagation delay for all links in the topology equals 10ms. The simulation time is 200 seconds. However, after the 20th second a number of flows abort data transmission and leave the channel, thus freeing up network bandwidth (100 initial flows are decreased to 10, 20, 30, 40 of 50 flows respectively, Fig. 6). TCP Tahoe performance is measured against two different queue management algorithms, namely Drop-Tail and RED. As shown in our experiments (Fig. 6), TCP Tahoe performance deteriorates in the presence of RED: packet drops result in abrupt reduction of the congestion window. Even when a number of TCP connections end, RED congestion avoidance mechanism insists on dropping packets based on the average queue size, thus preventing TCP Tahoe senders from exploiting the available network capacity.

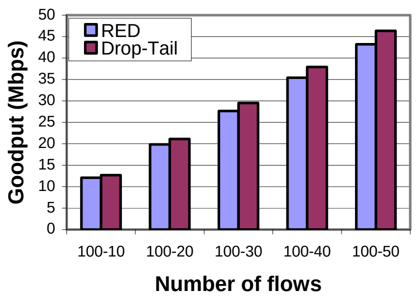

Figure 6. System goodput

4.4. Less functionality in the network can favor greedy flows

One other interesting aspect to discuss about is that incorporating no intelligence into the network can actually empower greedy flows to obtain larger share of network resources. On the contrary, putting more functionality into the network can restrain aggressive transport protocols from stealing bandwidth, and can safeguard fairness. However, there exist situations where aggressive protocols might significantly be punished by the network, thus permitting smoother transport protocols to achieve higher goodput.

In support of our claim, we evaluate the coexistence of TCP Reno and TCP Tahoe, an aggressive and a conservative protocol respectively when considered together. In our experiments, an equal number of TCP Reno and TCP Tahoe connections compete for the same channel. The number of total connections varies from 20 to 80 among different simulation runs, each of which lasts for 200ms. Every link in the dumbbell topology has a bandwidth of 10Mbps and the propagation delay for the edge and the bottleneck links are 3ms and 15ms, respectively. Also, the buffer size for the bottleneck link is set to 50 packets. As presented in our results (Fig. 7), when Drop-Tail is used, TCP Reno flows achieve higher goodput by means of behaving more aggressively than TCP Tahoe connections. In the presence of RED though (Fig. 8), TCP Reno gets punished by the congestion avoidance mechanism of RED, thus permitting TCP Tahoe protocol to utilize better the network capacity.

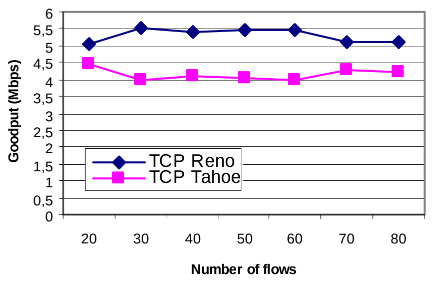

Figure 7. System goodput (with Drop-Tail)

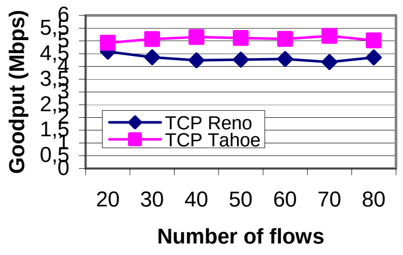

Figure 8. System goodput (with RED)

5. Discussion

In this paper we highlight our observation that the interaction of mechanisms, which lie in different layers, may have significant impact on various network performance metrics (such as network congestion and fairness) as well as on end-system performance. We demonstrate through experiments that congestion control algorithms of the transport and network layers implicitly interact with each other; furthermore, we present cases of miscollaboration among these layers.

In [23] authors present an explicit cross-layer interaction scheme. They refer to the various possible types of explicit cross-layer signaling and present the difficulties of deploying such a communication between the end-hosts and the routers. We claim that the interaction of these layers may also involve implicit impact on some protocols and mechanisms. We explore cases where implicit interaction of mechanisms results in undesirable situations; indeed, implicit schemes may deteriorate the performance of protocols and mechanisms in both layers.

It becomes increasingly important to seek for congestion control mechanisms that are compatible; otherwise, we will arrive at a point where specific mechanisms at one layer will only cooperate well with specific ones at the other. Inevitably, the interaction itself needs to become an integral part of the evaluation procedure, which needs to be further refined.

Furthermore, we argue that the impact of new mechanisms or protocols on existing ones needs to be quantified; the compatibility of interactive algorithms needs to be verified; and the sophistication needs to be justified in this context. Although we do not discuss it in this paper, new metrics should be introduced, in order to evaluate a mechanism in such perspective.

6. Conclusions

In this paper, we explore interactions between widely used congestion avoidance mechanisms of the transport and the network layers, and present cases of miscollaboration. Sophisticated mechanisms may cancel the functionality of each other. Existing protocols may not be able to interpret the function and collaborate with relatively newer mechanisms that are deployed in the Internet. Less functionality in the network may favor greedy flows, while certain congestion control mechanisms in the network layer might punish aggressive transport protocols and facilitate more conservative ones. All these issues call for cross-layer evaluation schemes.

Present work can be further extended: more complex topologies can be used for our experiments, interaction of transport and network mechanisms can be elaborated in a more detailed manner under various network conditions, presence of a larger number of mechanisms and their collaboration or mis-interaction can be tested. We essentially attempt to reconsider the congestion control scheme in a more accurate way and our initial work steps towards this direction. Our ultimate goal is to lay the foundations of an implicit cross-layer approach such that mechanisms in different layers do not undermine or cancel each other but they rather complement each other and allow for future evolution and deployment of new protocols or mechanisms in the Internet.

ACKNOWLEDGEMENT

This work was funded by the Europen Commission and the project PENED 2003 of GSRT.

7. References

[1] S. Floyd and T. Henderson, “The NewReno Modification to TCP's Fast Recovery Algorithm”, *RFC 2582*, 1999.

[2] B. Braden et al, "Recommendations on Queue Management and Congestion Avoidance in the Internet", *RFC 2309*, 1998.

[3] L. Mamatas, V. Tsaoussidis and Chi Zhang, "Approaches to Congestion Control in Packet Networks”, available from http://citeseer.ist.psu.edu/mamatas04approaches.html.

[4] L. Brakmo and L. Peterson, “TCP Vegas: End to End Congestion Avoidance on a Global Internet”, *IEEE JSAC*,, October 1995.

[5] C. Zhang and V. Tsaoussidis, “TCP Real: Improving Real-time Capabilities of TCP over Heterogeneous Networks”, in Proc. of *the 11th IEEE/ACM NOSSDAV* 2001, NY, 2001.

[6] C. Zhang and V. Tsaoussidis, “The interrelation of TCP responsiveness and smoothness in heterogeneous networks”, in Proc. *7th IEEE Symposium on Computers and Communications (ISCC 2002),* Italy, 2002.

[7] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi and R. Wang, “TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links”, in Proc. of *ACM MOBICOM*, pp. 287-297, 2001.

[8] P. C. Attie, A. Lahanas, and V. Tsaoussidis, “Beyond AIMD: Explicit fair-share calculation”, in Proccedings of *the 8th IEEE ISCC 2003*, 2003.

[9] The Network Simulator, ns-2, www.isi.edu/nsnam/ns.

[10] Dah-Ming Chiu and Raj Jain, “Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks”, *Computer Networks and ISDN Systems*, pp. 1-14, 1989.

[11] G. Hasegawa, T. Matsuo, M. Murata, and H. Miyahara, "Comparisons of packet scheduling algorithms for fair service among connections on the Internet," in Proc. of *IEEE INFOCOM 2000*, March 2000.

[12] C. Zhang, M. Khanna and V. Tsaoussidis, "Experimental Assessment of RED in Wired/Wireless Networks", *International Journal of Communication Systems*, pp. 287-302, Wiley, 2004.

[13] S. Floyd, R. Gummadi, S. Shenker, “Adaptive RED: An Algorithm for Increasing the Robustness of RED’s Active Queue Management”, 2001.

[14] W. Feng, D. Kandlur, D. Saha, K. Shin, “Blue: A New Class of Active Queue Management Algorithms”, *UM CSE-TR-387-99,* 1999.

[15] W. Feng, D. Kandlur, D. Saha, and K. Shin, “Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness”, in Proc. of *IEEE INFOCOM*, 2001.

[16] S. Athuraliya, S. H. Low,V. H. Li, and Q.Yin, “REM: Active queue management”, *IEEE Network Magazine*, pp. 48–53, 2001.

[17] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, “One More Bit Is Enough”, in SIGCOMM 2005, 2005.

[18] J. Mo, R. J. La, V. Anantharam, and J. C. Walrand, “Analysis and comparison of TCP Reno and Vegas”, in Proc. of *IEEE INFOCOM '99*, New York, 1999.

[19] Technical Specification from Cisco, “Distributed Weighted Random Early Detection”, URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.pdf.

[20] Sally Floyd and Van Jacobson, “Random Early Detection Gateways for Congestion Avoidance”, *IEEE/ACM ToN*, V.1 N.4, p. 397-413, August 1993.

[21] M. Handley, S. Floyd, et al., “TCP Friendly Rate Control (TFRC): Protocol Specification”, *RFC 3448*, January 2003.

[22] T. J. Ott, T. V. Lakshman, and L. H. Wong, “SRED: Stabilized RED”, in Proc. of *INFOCOM’99*, 1999.

[23] P. Sarolahti and S. Floyd, "Cross-layer Indications for Transport Protocols", Internet draft, work in progress, October 2006, http://www.icir.org/floyd/papers.html.
[24] Mathis, M., Mahdavi, J., Floyd, S., and Romanow, A., "TCP Selective Acknowledgement Options", *RFC 2018*, April 1996.