DETECTING PACKET LOSS AND ROUTE CHANGES WHEN CONGESTION OCCURS IN TCP

G Sai Chaitanya Kumar, K Durga Prasad, B. Sudheer Reddy

1Dept. of CSE, Paladugu Parvathi Devi College of Engg & Tech, Vijayawada, A.P.,India,
2 Dept. of CSE, Sri Viveka College of Engg & Tech, Vijayawada, A.P.,India, dpkondisetty@gmail.com
3 Asst.Professor, Dept. of CSE, Montessori Siva Sivani Institute of Technology, Vijayawada, A.P.,India,

ABSTRACT

The Transmission Control Protocol (TCP) carries most Internet traffic, so performance of the Internet depends to a great extent on how well TCP works. Performance characteristics of a particular version of TCP are defined by the congestion control algorithm it employs. This paper presents a survey of various congestion control proposals that preserve the original Host-to-host idea of TCP—namely, that neither sender nor receiver relies on any explicit notification from the network. The proposed solutions focus on a variety of problems, starting with the basic problem of eliminating the phenomenon of congestion collapse, and also include the problems of effectively using the available network resources in different types of environments (wired, wireless, high-speed, long-delay, etc.). In a shared, highly distributed, and heterogeneous environment such as the Internet, effective network use depends not only on how well a single TCP based application can utilize the network capacity, but also on how well it cooperates with other applications transmitting data through the same network.

INTRODUCTION

Most current internet applications rely on the Transmission control protocol (TCP) to deliver Data reliably across the network. The TCP carries most Internet traffic, so performance of the Internet depends to a great extent on how well TCP works. Performance characteristics of a particular version of TCP are defined by the congestion control algorithm it employs.

We are presenting a survey of various congestion control proposals that preserve the original host-to-host idea of TCP—namely, that neither sender nor receiver relies on any explicit notification from the network. The proposed solutions focus on a variety of problems, starting with the basic problem of eliminating the phenomenon of congestion collapse, and also include the problems of effectively using the available network resources in different types of environments (wired, wireless, high-speed, long-delay, etc.). In a shared, highly distributed, and heterogeneous environment such as the Internet, effective network use depends not only on how well a single TCP based application can utilize the network capacity, but also on how well it cooperates with other applications transmitting data through the same network.

Our survey shows that many host-to-host techniques have been developed that address several problems with different levels of reliability and precision. There have been enhancements allowing senders to detect fast packet losses and route changes. Other techniques have the ability to estimate the loss rate, the bottleneck buffer size, and level of congestion. The survey describes each congestion control alternative, its strengths and its weaknesses. Additionally, techniques that are in common use or available for testing are described.

©2011, JPRO, All Right Reserved.
sharing system (e.g., road traffic) exceeds the total system capacity, the effective load will go to zero (collapses) as load increases 75% of data packets dropped on forward path and 75% of ACKs dropped on reverse: only 6.25% of packets are acknowledged.

![Fig 1: Congestion collapse rationale.](image)

With regard to TCP, the origins of this effect, known as a congestion collapse, can be illustrated using a simple example. Let us consider a router placed somewhere between networks A and B which generate excessive amounts of TCP traffic (Figure 1). Clearly, if the path from A to B is congested by 400% (4 times more than the router can deliver), at least 75% of all packets from network A will be dropped and at most 25% of data packets may result in ACKs. If the reverse path from B to A is also congested (also by 400%, for example), the chance that ACK packets get through is also 25%. In other words, only 25% of 25% (i.e., 6.25%) of the data packets sent from A to B will be acknowledged successfully. If we assume that each data packet requires its own acknowledgement (not a requirement for TCP, but serves to illustrate the point), then a 75% loss in each direction causes a 93.75% drop in throughput (good put) of the TCP like flow.

To resolve the congestion collapse problem, a number of solutions have been proposed. All of them share the same idea, namely of introducing a network-aware rate limiting mechanism alongside the receiver-driven flow control. For this purpose the congestion window concept was introduced: a TCP sender’s estimate of the number of data packets the network can accept for delivery without becoming congested. In the special case where the flow control limit (the so-called receiver window) is less than the congestion control limit (i.e., the congestion window), the former is considered a real bound for outstanding data packets. Although this is a formal definition of the real TCP rate bound, we will only consider the congestion window as a rate limiting factor, assuming that in most cases the processing rate of end-hosts is several orders of magnitude higher than the data transfer rate that the network can potentially offer.

II) PACKET REORDERING

All the congestion control algorithms discussed in the previous section share the same assumption that the network generally does not reorder packets. This assumption has allowed the algorithms to create a simple loss detection mechanism without any need to modify the existing TCP specification. The standard already requires receivers to report the sequence number of the last in-order delivered data packet each time a packet is received, even if received out of order. In the idealized case, the absence of reordering guarantees that an out-of-order delivery occurs only if some packet has been lost. Thus, if the sender sees several ACKs carrying the same sequence numbers (duplicate ACKs), it can be sure that the network has failed to deliver some data and can act accordingly (e.g., retransmit lost packet, infer a congestion state, and reduce the sending rate). Of course in reality, packets are reordered. This means that we cannot consider a single duplicate ACK (i.e., ACK for an already ACKed data packet) as a loss detection mechanism with high reliability.

To solve this problem of a false loss detection, a solution employed as a rule of thumb establishes a threshold value for the minimal number of duplicate ACKs required to trigger a packet loss detection (e.g., three). However, there is a clear conflict with this approach. Loss detection will be unnecessarily delayed if the network does not reorder packets. At the same time, the sender will overreact (e.g., retransmit data or reduce transmission rate needlessly) if the network does in fact reorder packets. Packet reordering can stem from various causes. For example, it can be erroneous software or hardware behavior, such as bugs, misconfigurations, or failures. But packets can also be reordered in some networks as a side effect of a normal delivery process. For example, packets can be reordered if a router enforces diverse packets handling services (differentiated services and internally reschedules packets in its queue (active queue management). Also if the network provides some level of delivery guarantees (e.g., wireless networks), the underlying layer (physical or link layer) can retransmit some portion of the data without TCP's prompting and cause a shuffling of the upper layer packets. Finally, channel bundling and packet processing parallelism will likely contribute a good portion of the future Internet.

In this section we present a number of proposed TCP modifications that try to eliminate or mitigate reordering effects on TCP flow performance. All of these solutions share the following ideas: (a) they allow nonzero probability of packet reordering, and (b) they can detect out-of-order events and respond with an increase in flow rate (optimistic reaction).

Eifel Algorithm

Ludwig and Katz introduced the Eifel2 algorithm as an alternative method to alleviate the negative effects of packet reordering in TCP throughput. Instead of the TD-FR approach of introducing additional delay to the loss detection process based on duplicate ACKs Eifel tries to distinguish reordering and real loss events. It does not try to guess the event type upon reception of the first duplicate, but rather postpones the decision until the first non-duplicate ACK is received. In other words, if the TCP sender receives a number of duplicate ACKs, as in NewReno, it enters Fast Recovery. When a non-duplicate ACK is received, Eifel checks its content and makes a decision whether to continue Fast Recovery or abort recovery and restore the original congestion window value. On the one hand, the defined actions of Eifel do not affect normal operations of the base congestion control algorithm when there is no packet reordering. On the other hand, when some packets are reordered, the original sending rate will be restored very quickly.

III) WIRELESS NETWORKS

The growing spread of wireless networks has highlighted the need for TCP protocol modification. Originally designed for wired networks where congestion is the primary cause of packet losses, TCP is unable to react adequately to packet losses not related to congestion. Indeed, if a data packet is lost due to short-term radio frequency interference, then there are no router buffer overflows and TCP's decision to reduce the congestion window is wrong. Instead, it should just recover from the loss and continue the transmission as if nothing had
happened. Several solutions have been proposed to resolve this problem. One group gives up the idea of a pure host-to-host data transfer either by requiring routers to disclose the network state (e.g., using explicit congestion notification by relying on network channels to recover from the no congestion-related losses (e.g., link-layer retransmission or TCP packet snooping and loss recovery by intermediate routers), or by (c) isolating the wireless error-prone and wired error-safe transmission paths using an intermediate host. These approaches are beyond the scope of this survey and have been thoroughly discussed by Lochert et al.

In this section we focus on solutions that keep the host to-host idea and at the same time provide some level of resistance to non-congestion related packet losses. The bandwidth estimation technique proposed by Mascolo et al. as a part of TCP Westwood laid the foundation for the sender-side distinguishing between a congestion-related and an unrelated (random) loss without any support from the network. Research that follows identified several of Westwood’s weaknesses, for example, bandwidth overestimation, insufficient robustness in networks with extreme levels of transmission errors, etc.

CONCLUSION

In this work we have presented a survey of various approaches to TCP congestion control that do not rely on any explicit signaling from the network. The survey highlighted the fact that the research focus has changed with the development of the Internet, from the basic problem of eliminating the congestion collapse phenomenon to problems of using available network resources effectively in different types of environments (wired, wireless, high-speed, long-delay, etc.).

In the first part of this survey, we classified and discussed proposals that build a foundation for host-to-host congestion control principles. The first proposal, Tahoe, introduces the basic technique of gradually probing network resources and relying on packet loss to detect that the network limit has been reached. Unfortunately, although this technique solves the congestion problem, it creates a great deal of inefficient use of the network.

The second part of the survey is devoted to a group of congestion control proposals that are focused on environments where packets are frequently reordered. These proposals show that in such environments, efficiency can be improved significantly by (1) delaying the control actions (TD-FR), or (2) by undoing previously applied actions if reordering is detected (Eifel, DOOR), or (3) by refining the network state estimation heuristic (PR, RR).

In the third part of our survey, we showed that basic host-to-host congestion control principles can solve not only the direct congestion problem but also provide a simple traffic prioritizing feature. Two algorithms examined (Nice and LP), applying slightly different techniques to achieve the same goal, have the same aim: to provide an opportunity to send non-critical data reliably without interfering with other data transfers.

REFERENCES