ISSN:0975-9646

V. Akhila Reddy et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (4) , 2012,4913 - 4916

Analysis and Study of Host To Host Congestion Control for TCP

[1] V.Akhila Reddy, [1] D.Jamuna, [1] L.Haritha,[2] G.Rakesh Reddy

[1]CSE Dept,
Jayaprakash Narayan College of Engg.
Mahabubnagar, Andrapradesh

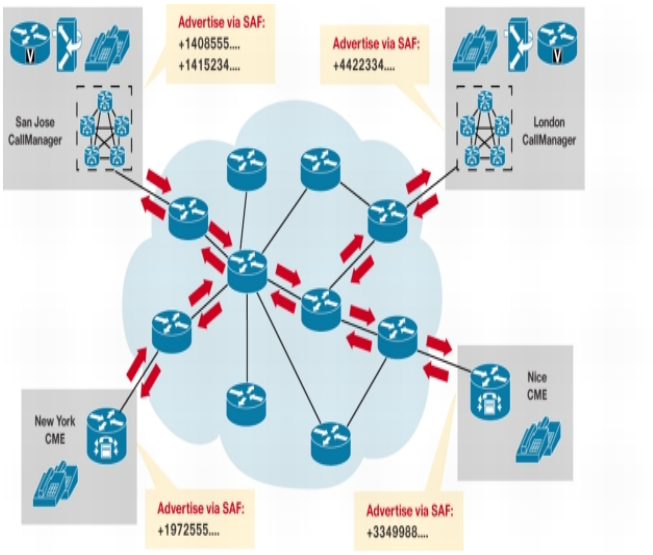
[2]Vivekananda college of Engg
Mahabubnagar, Andrapradesh

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 TCPbased 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 over the last 20 years 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.

I .INTRODUCTION

In data networking and queueing theory, **network congestion** occurs when a link or node is carrying so much data that its quality of service deteriorates. Typical effects include queueing delay, packet loss or the blocking of new connections. A consequence of these latter two is that incremental increases in offered load lead either only to small increase in network throughput, or to an actual reduction in network throughput.

Network protocols which use aggressive retransmissions to compensate for packet loss tend to keep systems in a state of network congestion even after the initial load has been reduced to a level which would not normally have induced network congestion. Thus, networks using these protocols can exhibit two stable states under the same level of load. The stable state with low throughput is known as **congestive collapse**.



Modern networks use congestion control and network congestion avoidance techniques to try to avoid congestion collapse. These include: exponential backoff in protocols such as 802.11's CSMA/CA and the original Ethernet, window reduction in TCP, and fair queueing in devices such as routers. Another method to avoid the negative effects of network congestion is implementing priority schemes, so that some packets are transmitted with higher priority than others. Priority schemes do not solve network congestion by themselves, but they help to alleviate the effects of congestion for some services. An example of this is 802.1p. A third method to avoid network congestion is the explicit allocation of network resources to specific flows.

II EVOLUTION OF TCP

To resolve the *congestion collapse* problem, a number of solutions have been proposed.

A.TCP TAHOE

Tahoe refers to the TCP congestion control algorithm which was suggested by Van Jacobson . TCP is based on a principle of 'conservation of packets', i.e. if the connection is running at the available bandwidth capacity then a packet is not injected into the network unless a packet is taken out as well. TCP implements this principle by using the acknowledgements to clock outgoing packets because an acknowledgement means that a packet was taken off the wire by the receiver. It also maintains a congestion window CWD to reflect the network capacity.

However there are certain issues, which need to be resolved to ensure this equilibrium.

1) Determination of the available bandwidth.

2) Ensuring that equilibrium is maintained.

3) How to react to congestion.

For congestion avoidance Tahoe uses 'Additive Increase Multiplicative Decrease'. A packet loss is taken as a sign of congestion and Tahoe saves the half of the current window as a threshold. value.

The problem with Tahoe is that it take a complete timeout interval to detect a packet loss. it doesn't send immediate ACK's, it sends cumulative acknowledgements, therefore it follows a 'go back n ' approach.

B.TCP RENO

This Reno retains the basic principle of Tahoe, such as slow starts and the coarse grain re-transmit timer. So Reno suggest an algorithm called 'Fast ReTransmit'. Whenever we receive 3 duplicate ACK's we take it as a sign that the segment was lost, so we re-transmit the segment without waiting for timeout.

Reno perform very well over TCP when the packet losses are small. But when we have multiple packet losses in one window then RENO doesn't perform too well and it's performance is almost the same as Tahoe under conditions of high packet loss.

New-Reno

It prevents many of the coarse grained timeouts of New-Reno as it doesn't need to wait for 3duplicate ACK's before it retransmits a lost packet.

Its congestion avoidance mechanisms to detect 'incipient' congestion are very efficient and utilize nework resources much more efficiently.

Because of its modified congestion avoidance and slow start algorithm there are fewer retransmits.

C.TCP SACK

TCP with 'Selective Acknowledgments' is an extension of TCP Reno and it works around the problems face by TCP RENO and New-reno namely detection of multiple lost packets, and re-transmission of more than one lost packet per RTT. SACK retains the slow-start and fastretransmit parts of RENO. It also has the coarse grained timeout of Tahoe to fall back on, incase a packet loss is not detected by the modified algorithm. SACK TCP requires that segments not be acknowledged cumulatively but should be acknowledged selectively. Thus each ACK has a block which describes which segments are being acknowledged.

The biggest problem with SACK is that currently selective acknowledgements are not provided by the receiver To implement SACK we'll need to implement selective acknowledgment which is not a very easy task.

D.TCP VEGAS

Vegas is a TCP implementation which is a modification of Reno. It builds on the fact that proactive measure to encounter congestion are much more efficient than reactive ones. It tried to get around the problem of coarse grain timeouts by suggesting an algorithm which checks for timeouts at a very efficient schedule. Also it overcomes the problem of requiring enough duplicate acknowledgements to detect a packet loss, and it also suggest a modified slow start algorithm which prevent it from congesting the network. It does not depend solely on packet loss as a sign of congestion. It detects congestion before the packet losses occur. However it still retains the other mechanism of Reno and Tahoe, and a packet loss can still be detected by the coarse grain timeout of the other mechanisms fail.

E.TCP FACK

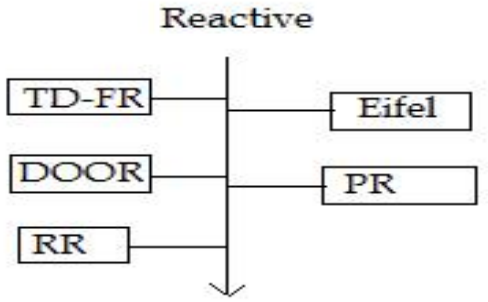
Although SACK (Section II-E) provides the receiver with extended reporting apabilities, it does not define any particular congestion control algorithms. We have informally discussed one possible extension of the Reno algorithm utilizing SACK nformation, whereby the congestion window is not multiplicatively reduced more than once per RTT. Another approach is the FACK (Forward Acknowledgments) congestion control algorithm . It defines recovery procedures which,unlike the *Fast Recovery* algorithm of standard TCP (TCP Reno), use additional information available in SACK to handle error recovery (flow control) and the number of outstanding packets (rate control) in two separate mechanisms. The flow control part of the FACK algorithm uses selective ACKs to indicate losses. It provides a means for timely retransmission of lost data packets, as well. Because retransmitted data packets are reported as lost for at least one RTT and a loss cannot be instantly recovered, the FACK sender is required to retain information about retransmitted data. This information should at least include the time of the last retransmission in order to detect a loss using the legacy timeout method (RTO). The rate control part, unlike Reno's and NewReno's *Fast Recovery* algorithms, has a direct means to calculate the number of outstanding data packets using information extracted from SACKs. Instead of the congestion window inflation technique, the FACK maintains three special state variables : (1) *H*, the highest sequence number of all sent data packets—all data packets with sequence number less than *H* have been sent at least once; (2) *F*, the forward-most sequence number of all acknowledged data packets—no data packets with sequence number above *F* have been delivered .

III PACKET REORDERING

The basic idea behind TCP-PR is to detect packet losses through the use of timers instead of duplicate acknowledgments. This is prompted by the observation that, under persistent packet reordering, duplicate

acknowledgments are a poor indication of packet losses. Because TCP-PR relies solely on timers to detect packet loss, it is also robust to acknowledgment losses.

Two issues arise when considering TCP-PR over networks without packet reordering: performance and fairness. The first issue is whether TCP-PR performs as well as other TCP implementations under "normal" conditions, i.e., no packet reordering. Specifically, for a fixed topology and background traffic, does TCP-PR achieve similar throughput as standard TCP implementations? The second concern is whether TCP-PR and standard TCP implementations are able to coexist fairly.



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). Nonetheless, these proposals have fundamental differences due to a range of acceptable degrees of packet reordering, from moderate in *TD-FR* to extreme in *TCP PR*, and different baseline congestion control approaches. The development of these proposals is highlighted in the above Figure .

IV HIGH-SPEED/LONG-DELAY NETWORKS

In connectionless networks, the role of flow control is to modify the natural sending rate of an application to match the realities of network capacity, and to make the data stream better behaved. This is done by insisting that some assertions about the data stream are always valid, for example, that no more than 12 kbtyes of data will be outstanding (sent but unacknowledged) at any given time. The test of a flow control protocol is its effectiveness in making network operation smoother as a result of this modification**.**

In a high speed network with large delays, problems can arise from two sources.

a) Delay in knowing about network state can cause buffer buildups, and eventual congestion.

b) High speed sources can inject data rapidly into the network, causing problems for other sources.

The PP flow control protocol consistently matches or outperforms the competing schemes,because of its short start up times, and ability to monitor network state. Inflexible protocols such as 'generic' are unsuitable for high-speed networks with propagation delays. Schemes that involve a slow start phase, such as JK (and DECbit will discriminate against conversations with a long propagation delay, which will suffer loss of throughput.

PP works well in the simulated scenarios, since it can rapidly adapt to changes in the network state.

V FUTURE ENHANCEMENT

Currently we have as situation where there is no Single congestion control approach for TCP that can universally be applied to all network environments. One of the primary causes is a wide variety of network environments and different (and sometimes opposing) network owners' views regarding which parameters should be optimized. A number of the congestion control algorithms.

Moreover, the current version of Linux kernel provides an API for software developers to choose any one of the supported algorithms for a particular connection. However, there are not yet the well-defined and broadly-accepted criteria to serve as a good baseline for appropriately selecting a congestion control algorithm. Additionally, objective guidelines to select a proper congestion control for a concrete network environment are yet to be defined.

VI 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. As we showed, solutions to the efficiency problem include algorithms that (1) refine the core congestion control principle by making more optimistic assumptions about the network (Reno, NewReno); or (2) refine the TCP protocol to include extended reporting abilities of the receiver (SACK, DSACK), which allows the sender to estimate the network state more precisely (FACK, RR-TCP); or (3) introduce alternative concepts for network state estimation through delay measurements (DUAL, Vegas, Veno).

REFERENCES

[1]Alexander Afanasyev, Neil Tilley, Peter Reiher, and Leonard Kleinrock" Host-to-Host Congestion Control for TCP"

[2] J. Postel, "RFC793—transmission control protocol," *RFC*, 1981.

[3] C. Lochert, B. Scheuermann, and M. Mauve, "A survey on congestion control for mobile ad hoc networks," *Wireless Communications and Mobile Computing*, vol. 7, no. 5, p. 655, 2007.

[4] Stephan Bohacek, Jo˜ao P. Hespanha, Junsoo Lee ,Chansook Lim ,Katia Obraczka "TCP-PR: TCP for Persistent Packet Reordering"

[5]Srinivasan Keshav" Flow Control in High-Speed Networks with Long Delays"

V.Akhila Reddy Pursuing Mtech(CSE),at Jayaprakash narayan college of engineering, Mahabubnagar, Andrapradesh . Btech from Jayaprakash narayan college of engineering. Mahabub nagar, Andrapradesh.
Her areas of interest include Information Security currently focusing on TCP Networks

Prof.D.Jamuna ,M.Tech, (,Ph.D).
Professor & HOD. At Jaya Prakash Narayan College of Engineer ing, mahabub nagar, Andra pradesh. M.Tech. degree in SE from School of Information Technology, JNTU, Hyd and Pursuing Ph.D from Rayalaseema University, Kurnool. Her areas of interest include Wireless networks, Information Security currently focusing on IP Networks

L.Haritha, working as **Associate Professor** in Jaya Prakash Narayan College of Engineering, mahabubnagar, Andrapradesh. Her areas of interest include Wireless networks, Information Security currently focusing on IP Networks .

G.Rakesh Reddy working as **Assistant professor** in Vivekananda institute of technology, shadnagar, mahabubnagar, Andrapradesh. MTECH from satyabhama university, Chennai. His areas of intrest include Network Information Security.