A Survey on Quality of Service and Congestion Control

Ashima
Computer Science and Engineering
Amity University
Noida, U.P, India
batra_ashima@yahoo.co.in

Sanjeev Thakur
Computer Science and Engineering
Amity University
Noida, U.P, India
sthakur.ascs@amity.edu

Abhishek Shukla
Computer Science and Engineering
Axis Colleges
Kanpur, U.P, India
abhish123@gmail.com

Abstract— Several techniques have been developed to meet the demand for faster and more efficient access to the Internet. Different applications have different networking requirements. For example, real-time interactive voice requires a low latency network path, but does not require high bandwidth. QoS refers to the ability of a network to provide improved service to selected network traffic over various underlying technologies including Frame Relay, SONET, and IP-routed networks. In particular, QoS features provide improved and more predictable network service by providing the following services: Supporting dedicated bandwidth, improving loss characteristics, avoiding and managing network congestion, shaping network traffic, setting traffic priorities across the network. This paper introduces various tcp protocol problems and their solution used in avoiding and managing network congestion.

Keywords-Congestion Control, TCP, Traffic Control, QoS, Traffic Management

I. INTRODUCTION

Although an analogy between computer networks and automobile highways is simplistic, the view of networks as a type of infrastructure points out the need for traffic control. Highways have a limited capacity which can be exceeded when many people want to travel at the same time. Vehicles begin to slow down and back up in a congested area. The backup spreads if traffic approaches the congested area faster than traffic can leave. Similarly, computer networks are designed to handle a certain amount of traffic with an acceptable level of network performance. Network performance will deteriorate if the offered traffic exceeds the given network capacity.

Packets will suffer long queuing delays at congested nodes and possibly packet loss if buffers overflow. Traffic Management refers to the set of traffic controls within the network that regulate traffic flows for the purpose of maintaining the usability of the network during conditions of congestion. Traffic management has multiple goals. First, it attempts to distinguish different types of traffic and handle each type in the appropriate way. For example, real-time traffic is forwarded with minimal delay while best-effort traffic can afford to wait longer for any unused bandwidth. Second, traffic management responds to the onset of congestion. TCP is an example of a protocol that adapts the rate of TCP sources to avoid serious congestion. Third, traffic management seeks to maintain an acceptable level of network performance under heavy traffic conditions. The primary means of protection are admission control and access regulation (or policing) which limits the rate of traffic entering the network. By restricting the total amount of carried traffic, congestion will be avoided, and acceptable network performance will thus be sustained at the expense of blocking some amount of ingress traffic. Quality of service (QoS) [11] is the end-to-end network performance seen from the viewpoint of a user's particular connection. QoS parameters are defined to quantify the performance needed and experienced by a user's application. For example, a real-time application might require QoS guarantees such as an end-to-end packet delay of 10 msec and packet loss rate of 10-6. The user would be assured that his or her application would see this level of performance. At the same time, a different user may require a different QoS from the same network. Ultimately, a certain amount of subjectivity is involved. A user will judge the QoS provided by the network according to how well his or her application seems to work.

II. TCP CONGESTION NOTIFICATION

Since in wireless technology there is difference between route failure and congestion in the network. TCP rely mostly upon implicit signals it learns from the network and remote host. TCP must make an educated guess as to the state of the network and trust the information from the remote host in order to control the rate of data flow. [2]

Whenever TCP transmits a segment the sender starts a timer which keeps track of how long it takes for an acknowledgment for that segment to return. This timer is known as the retransmission timer. If an acknowledgement [5] is returned before the timer expires, which by default is often initialized to 1.5 seconds, the timer is reset with no consequence. If however an acknowledgement for the segment does not return within the timeout period, the sender would retransmit the segment and double the retransmission timer value for each consecutive timeout up to a maximum of about 64 seconds [8]. If there are serious network problems, segments may take a few minutes to be successfully transmitted before the sender eventually times out and generates an error to the sending application. Fundamental to the timeout and retransmission strategy of TCP is the measurement of the round-trip time between two communicating TCP hosts. The round-trip time may vary during the TCP connection as network traffic patterns fluctuate and as routes become available or unavailable.[1]

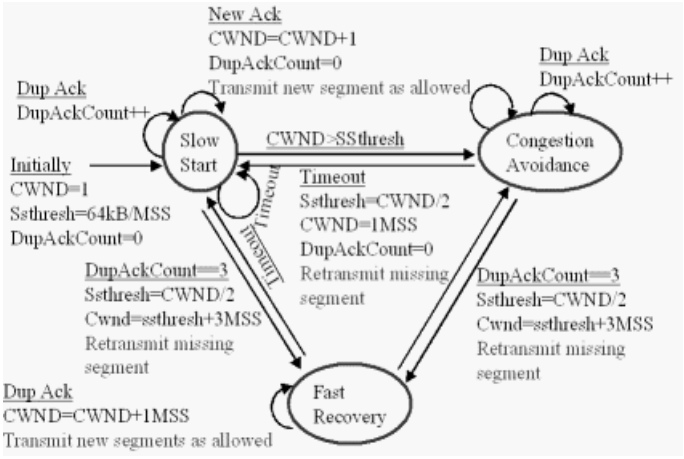

Figure 1. TCP Congestion Control State Diagram

III. CONGESTION MANAGEMENT

Congestion management features allow you to control congestion by determining the order in which packets are sent out an interface based on priorities assigned to those packets [11]. Congestion management entails the creation of queues, assignment of packets to those queues based on the classification of the packet, and scheduling of the packets in a queue for transmission. The congestion management QoS feature offers four types of queuing protocols, each of which allows you to specify creation of a different number of queues, affording greater or lesser degrees of differentiation of traffic, and to specify the order in which that traffic is sent. [6]

TABLE I. VARIOUS CONGESTION MANAGEMENT TOOLS

FIFO (First In First Out)	Weighted Fair Queuing	Priority Queuing	Low Latency Queuing
FIFO entails no concept of priority or classes of traffic. With FIFO, transmission of packets out the interface occurs in the order the packets arrive.	It is a flow-based queuing algorithm that does two things simultaneously: It schedules interactive traffic to the front of the queue to reduce response time, and it fairly shares the remaining bandwidth between high bandwidth flows.	PQ ensures that important traffic gets the fastest handling at each point where it is used. It was designed to give strict priority to important traffic.	The Low Latency Queueing feature brings strict priority queueing to Class-Based Weighted Fair Queueing (CBWFQ).

IV. CONGESTION AVOIDENCE

Congestion avoidance is a form of queue management. Congestion-avoidance techniques monitor network traffic loads to anticipate and avoid congestion at common network bottlenecks to oppose congestion-management techniques which control congestion after it occurs. [3]

A. AIMD (Additive Increase / Multiplicative Decrease)

Congestion-Window (cwnd) is a variable held by the TCP source for each connection which is based on the perceived level of congestion. The Host receives implicit (packet drop) or explicit (packet mark) indications of internal congestion. [9]

Effective Window = Max Window – (Last Byte Send – Last Byte Acknowledge)

In Additive Increase for each "cwnd's worth" of packets sent, increase cwnd by 1 packet. In practice, cwnd is incremented fractionally for each arriving ACK.

Increment = MSS x (MSS /cwnd)

cwnd = cwnd + Increment

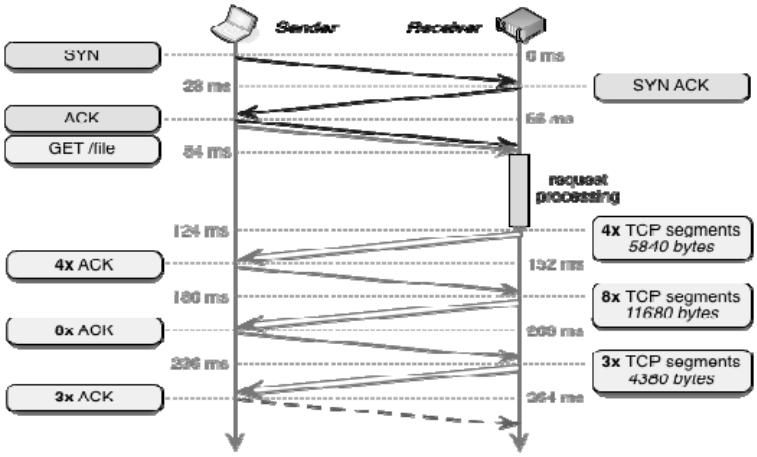

Figure 2. AIMD (Additive Increase / Multiplicative Decrease)

B. Slow Start

Linear additive increase takes too long to ramp up a new TCP connection from cold start. The slow start mechanism was added to provide an initial exponential increase in the size of cwnd. The source starts with cwnd= 1. Every time an ACK arrives, cwnd is incremented. Congestion-Window is effectively doubled per RTT "epoch". Two slow start situations:

a) At the very beginning of a connection cold start.

b) When the connection goes dead waiting for a timeout to occur (i.e., the advertized window goes to zero)

The TCP Tahoe congestion control strategy consists of multiple mechanisms. For each connection, TCP maintains a congestion window that limits the total number of unacknowledged packets that may be in transit end-to-end. The congestion window is an extension of the sliding window that TCP uses for flow control. When a connection is initialized, and after a timeout, TCP uses a mechanism called slow start to increase the congestion window. It starts with a window of two times the Maximum Segment Size (MSS). Although the initial rate is low, the rate of increase is very rapid. For every packet acknowledged, the congestion window increases by one MSS so that effectively the congestion window doubles for every RTT. [4]

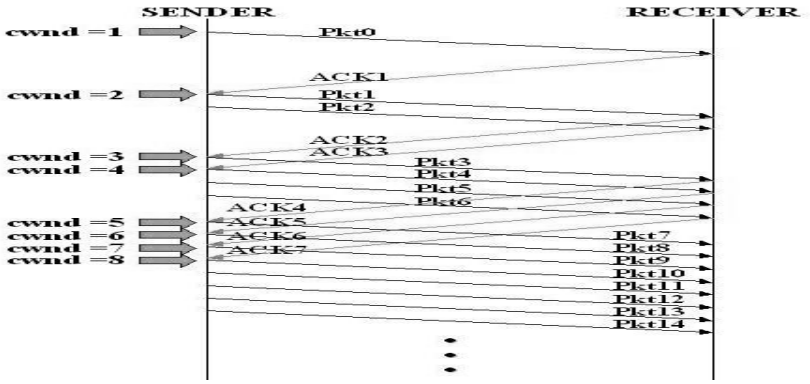

Figure 3. Slow Start

The window is doubled as follows: If the congestion window has two packets outstanding, and one packet is acknowledged, this means that the congestion window is increased to three packets, and only one packet is outstanding. I.e. the sender may now send two new packets. When the final packet (of the original two) is acknowledged, this allows the sender to increase the congestion window with one MSS yet again, bringing the total congestion window to four, and of these two are free. In other words, the congestion window has doubled. When the congestion window exceeds a threshold ssthresh, the algorithm enters a new state, called congestion avoidance. In some implementations, the initial ssthresh is large, resulting in the first slow start usually ending in a loss of a packet. The ssthresh is updated at the end of each slow start, and will often affect subsequent slow

starts triggered by timeouts. In the state of congestion avoidance, the congestion window is additively increased by one MSS every RTT, instead of the previous one MMS per acknowledged packet, as long as non-duplicate ACKs are received. When a packet is lost, the likelihood of receiving duplicate ACKs is very high. (It is also possible, though unlikely, that the stream has undergone extreme packet reordering, which would also prompt duplicate ACKs.) Triple duplicate ACKs are interpreted in the same way as a timeout. In such a case, Tahoe performs a "fast retransmit", reduces the congestion window to one MSS, and resets to the slow-start state.

C. Fast Retransmit

Upon receipt of three duplicate ACKs, the TCP Sender retransmits the lost packet. Coarse timeouts remained a problem, and Fast retransmit was added with TCP Tahoe. Since the receiver responds every time a packet arrives, this implies the sender will see duplicate ACKs. Generally, fast retransmit eliminates about half the coarse-grain timeouts. This yields roughly a 20% improvement in throughput. Fast retransmit does not eliminate all the timeouts due to small window sizes at the source. [10]

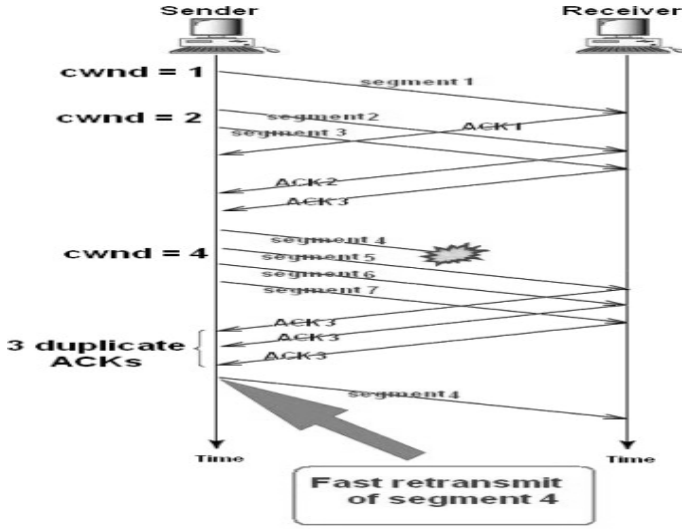

Figure 4. Fast Retransmit and Recovery

When fast retransmit detects three duplicate ACKs, start the recovery process from congestion avoidance region and use ACKs in the pipe to pace the sending of packets. The Reno version of TCP introduces a fast recovery phase. If three duplicate ACKs are received, Reno will halve the congestion window, perform a fast retransmit, and enter a state called fast recovery. In this state, TCP retransmits the missing packet that was signaled by three duplicate ACKs, and waits for an acknowledgment of the entire transmit window before returning to congestion avoidance. If there is no acknowledgment, i.e., if an ACK times out, TCP Reno experiences a timeout and enters the slow-start state, just like Tahoe.

Random Early Detection (RED) [7] is an active queue management algorithm, as well as a congestion avoidance algorithm. In the traditional tail drop algorithm, a router buffers as many packets as it can, and simply drops the ones it cannot buffer. If buffers are constantly full, the network is congested. Tail drop distributes buffer space unfairly among traffic flows. Tail drop can also lead to TCP global synchronization as all TCP connections "hold back" simultaneously, and then step forward simultaneously. RED monitors the average queue size and drops packets based on statistical probabilities. If the buffer is almost empty, all incoming packets are accepted. As the queue grows, the probability for dropping an incoming packet grows too. When the buffer is full, the probability has reached 1 and all incoming packets are dropped. RED is more fair than tail drop, in the sense that it is not biased against bursty traffic that uses only a small portion of the bandwidth. The more traffic a host transmits, the more likely it is that its packets are dropped, as the probability of a host's packet being dropped is proportional to the amount of data it has in a queue. Early detection helps avoid TCP global synchronization.

WRED combines the capabilities of the RED algorithm with the IP Precedence feature to provide for preferential traffic handling of higher priority packets. WRED can selectively discard lower priority traffic when the interface begins to get congested and provide differentiated performance characteristics for different classes of service. WRED differs from other congestion avoidance techniques such as queuing strategies because it attempts to anticipate and avoid congestion rather than control congestion once it occurs. By randomly dropping packets prior to periods of high congestion, WRED tells the packet source to decrease its transmission rate. If the packet source is using TCP, it will decrease its transmission rate until all the packets reach their destination, which indicates that the congestion is cleared. WRED reduces the chances of tail drop by selectively dropping packets when the output interface begins to show signs of congestion. By dropping some packets early rather than waiting until the queue is full, WRED avoids dropping large numbers of packets at once and minimizes the

chances of global synchronization. Thus, WRED allows the transmission line to be used fully at all times. WRED is only useful when the bulk of the traffic is TCP/IP traffic. With TCP, dropped packets indicate congestion, so the packet source will reduce its transmission rate. With other protocols, packet sources may not respond or may resend dropped packets at the same rate. Thus, dropping packets does not decrease congestion. WRED is also RSVP-aware, and it can provide the controlled-load QoS service of integrated service.

V. CONCLUSION

QoS provides differentiated services, which provide higher-priority to flows, or guaranteed services that provide an assured service level. Both of these are contrasted by best-effort services, which are provided by what is generally considered a lack of QoS. Congestion-management tools (PQ, CQ, WFQ, and CBWFQ) all manage the delivery of packets when there is more bandwidth than the link can handle. Queue management (WRED) is used for congestion avoidance within individual queues, as well as to prevent congestion in the internetwork. Using the behavior of TCP, WRED can throttle the speed of flows by dropping certain flows. It can also provide priority by dropping low-priority flows before high-priority flows. This paper demonstrates various congestion control protocols and a brief survey on the problems and solution associated with them. These TCP congestion control protocols can be aggregated with internet QoS to form fair and scalable network services.

REFERENCES

[1] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581 (Proposed Standard), Apr. 1999. Updated by RFC 3390.
[2] J. Nagle, RFC 896: Congestion Control in IP/TCP Internetworks, IETF Internet Standard, January 1984.
[3] Van Jacobson. Modified TCP Congestion Control Avoidance Algorithm. end-2-end-interest mailing list, April 30, 1990.
[4] Jamshid Mahdavi. Private e-mail to John Kristoff, December 12, 1999.
[5] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgement Options, October 1996, RFC 2018.
[6] V. Jacobson, "Congestion avoidance and control," SIGCOMM Comput. Commun. Rev., vol. 18, pp. 314–329, August 1988.[Online].Available: http://doi.acm.org/10.1145/52325.52356
[7] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance, "Networking, IEEE/ACM Transactions on, vol. 1, no. 4, pp. 397 –413, aug 1993.
[8] S. Floyd, "TCP and explicit congestion notification," SIGCOMM Comput. Commun. Rev. vol. 24, pp. 8–23, October 1994.[Online] Available:http://doi.acm.org/10.1145/205511.205512
[9] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: new techniques for congestion detection and avoidance," SIGCOMM Comput. Commun. Rev., vol. 24, pp. 24–35, October 1994. [Online] Available: http://doi.acm.org/10.1145/190809.190317
[10] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast sRecovery Algorithms, January 1997, RFC 2001.
[11] http://www.cisco.com/en/US/tech/tk543/tk543/tk544/tsd_technology_support_protocol_home.html