A STUDY ON CONGESTION AVOIDANCE USING TCL SCRIPT

Hemlata
Deptt. Of Comp. Sci. & Engg.
SKITM, (MD UNIVERSITY)
Bahadurgrh,India

V.K. Pandey
Deptt. Of Comp. Sci. & Engg.
SKITM, (MD UNIVERSITY)
Bahadurgrh,India

ABSTRACT:

As company intranets continue to grow it is increasingly important that network Administrators are aware of and have a handle on the different types of traffic that is traversing their networks.TCP(Transmission Control Protocol) is a protocol used along with the Internet Protocol (IP) to send data in the form of packets between computers over the Internet. TCP considers all packet timeouts in wired networks as due to network congestion and not to bit errors. It therefore considers all packet losses as due to congestion and consequently reduces the burst of packet, diminishing at the same time the network throughput. In Existing work the classical Slow-start algorithm along with reserved bits and SNR ratio has been developed for this problem. But Slow-start algorithm takes some time to receive the acknowledgement from the receiver. This paper proposes a new TCP RED congestion avoidance algorithm and a tcl script appropriate for wireless as well as wired networks and is capable of distinguishing congestion losses from error losses proactively. Proposed RED algorithm won't wait for the packet's acknowledgement.

Keywords:- Network Traffic, Congestion avoidance, Tcl Script, RED algorithm.

1. INTRODUCTION

NETWORK TRAFFIC MANAGEMENT

In data networking and queuing theory, **network congestion** occurs when a link or node is carrying so much data that its quality of service deteriorates. Typical effects include queuing delay, packet loss or the blocking of new connections. A consequence of the latter two effects is that an incremental increase in offered load leads either only to a 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 congestion avoidance techniques to try to avoid congestion collapse. These include: exponential back off in protocols such as 802.11 CSMA/CA and the original Ethernet, window reduction in TCP, and fair queuing 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.

2. Congestion Solution

Congestion control and two basic approaches

- Open-loop: try to prevent congestion occurring by good design.
- Closed-loop: monitor the system to detect congestion, pass this information to where action can be taken, and adjust system operation to correct the problem (detect feedback and correct) .

3. Congestion Avoidance

Congestion avoidance techniques monitor network traffic loads in an effort to anticipate and avoid congestion at common network and internetwork bottlenecks before it becomes a problem. These techniques are designed to provide preferential treatment for premium (priority) class traffic under congestion situations while concurrently maximizing network throughput and capacity utilization and minimizing packet loss and delay. WRED and DWRED are the Cisco IOS QoS congestion avoidance features.

Congestion Avoidance Algorithm

- Slow-start algorithm
- Fast retransmit
- Fast recovery

Slow –Start Algorithm

Slow-start is one of the algorithms that TCP uses to control congestion inside the network. It is also known as the exponential growth phase. Slow-start begins initially with a congestion window Size (cwnd) of 1, 2 or 10. The value of the Congestion Window will be increased with each acknowledgment received, effectively doubling the window size each round trip time ("although it is not exactly exponential because the receiver may delay its ACKs, typically sending one ACK for every two segments that it receives"). The transmission rate will be increased with slow-start algorithm until either a loss is detected, or the receiver's advertised window (rwnd) is the limiting factor, or the slow start threshold (ssthresh) is reached. - If a loss event occurs, TCP assumes that it is due to network congestion and takes steps to reduce the offered load on the network. These measurements depend on the used TCP congestion avoidance algorithm. - Once ssthresh is reached, TCP changes from slow-start algorithm to the linear growth (congestion avoidance) algorithm.

Fast Retransmit

TCP segment retransmission that when segments are received by a device out of order (meaning, non-contiguously), the recipient will only acknowledge the ones received contiguously. The Acknowledgment Number will specify the sequence number of the byte it expects to receive next. So, in the example given in that topic, Segment #1 and #2 were acknowledged while #4 was not, because #3 was not received. It is possible for a TCP device to in fact respond with an acknowledgment when it receives an out-of-order segment, simply "reiterating" that it is stuck waiting for a particular byte number. So, when the client in that example receives Segment #4 and not Segment #3, it could send back an Acknowledgment saying "I am expecting the first byte of Segment #3 next". Now, suppose this happens over and over. The server, not realizing that Segment #3 was lost, sends Segment #5, #6 and so on. Each time one is received, the client sends back an acknowledgment specifying the first byte number of Segment #3. Eventually, the server can reasonably conclude that Segment #3 is lost, even if its retransmission timer has not expired. The Fast Retransmit feature dictates that if three or more of these acknowledgments are received, all saying "I want the segment starting with byte #N", then it's probable that the segment starting with byte #N has been lost, usually because it was dropped due to congestion.

Fast Recovery

When *Fast Retransmit* is used to re-send a lost segment, the device using it performs *Congestion Avoidance*, but does not use *Slow Start* to increase the transmission rate back up again. The rationale for this is that since multiple *ACKs* were received by the sender all indicating receipt of out-of-order segments, this indicates that several segments have already been removed from the flow of segments between the two devices. For efficiency reasons, then, the transmission rate can be increased more quickly than when congestion occurs in other ways. This improves performance compared to using the regular *Congestion Avoidance* algorithm after *Fast Retransmit*.

4. RED ALGORITHM

The RED algorithm is a congestion avoidance technique used in communication networks to avoid network congestion. Compared to existing algorithms, RED monitors network traffic loads in an effort to anticipate and to avoid congestion at common network bottlenecks, where the system triggers before any congestion actually occurs . The proposed RED algorithm when operating in wireless mode exploits the SNR ratio of the communication line to decide whether a timed-out packet was due to congestion or error loss. When a TCP connection first begins, the alternative RED algorithm computes the average of the queue. Then it checks the average value with the threshold value. If the average is less than the threshold value, the packet will be queued. If the average is greater than the threshold value the algorithm checks the reserved bits which indicates the connection is wired or wireless. The first bit of the reserved field is set according to the type of the link, i.e., b=0 for wired and b=1 for wireless connection. If it is equal to 0 (wired link), then timeout is considered to be due to congestion. So the packet will be discarded. In contrast, if the reserved bit b is equal to 1 (wireless link), the SNR ratio of the connection is checked. In case it is within a high range, i.e., greater than 5dB (SNR>5dB), then timeout is considered to be due to congestion and the packet will be dropped. However, if SNR ratio is within a low range, i.e., less than 5dB (SNR<5dB), then timeout is considered to be due to error and the timed-out packet is retransmitted to the receiver

Signal-to-noise ratio

Signal-to-noise ratio is defined as the ratio of the power of a signal (meaningful information) and the power of background noise (unwanted signal):

$$\mathrm{SNR} = \frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}},$$

where *P* is average power. Both signal and noise power must be measured at the same or equivalent points in a system, and within the same system bandwidth.

If the variance of the signal and noise are known, and the signal is zero-mean:

$$\mathrm{SNR} = \frac{\sigma^2_{\mathrm{signal}}}{\sigma^2_{\mathrm{noise}}}.$$

RED algorithm pseudo-code is outlined as follows:

1. Computes the queue average value

2. Check the average avg with threshold value

3. If avg<threshold packet will be queued for sending to receiver.

4. If avg>threshold check the reserved bit b

5. If the reserved bit value b is 0 the connection is consider to be wired and the timeout due to congestion so the

packet is discarded.

6. If the reserved bit value b is 1 the connection is consider to be wireless and the SNR ratio will be checked.

1. If the SNR>5dB then the timeout due to congestion and the packet will be dropped.

2. If the SNR<5dB then the timeout due to error and the packet will retransmit to the receiver.

Existing RED algorithm

Initialization
avg←0
count←-1
for each packet arrival
calculate new avg of queue size
if the queue is nonempty
avg ←(1 - wq) avg + wq q
else
m ←f (time - q_time)
avg ←(1 – wq)m avg
if minth <= avg < maxth
increment count
calculate probability pa :
pb ←maxp (avg – minth) / (maxth – minth)
pa ←pb / (1 – count.pb)

with probability pa :
mark the arriving packet
count ←0
else if maxth <= avg [15]

Enhancement of RED algorithm

mark the arriving packet
count ←0
else b=1 then wired
if SNR>5dB
mark the arriving packet
count ←0
else SNR<5dB
mark the arriving packet
count ←0
retransmit the packet
else count ←-1
when queue becomes empty
q_time ←time

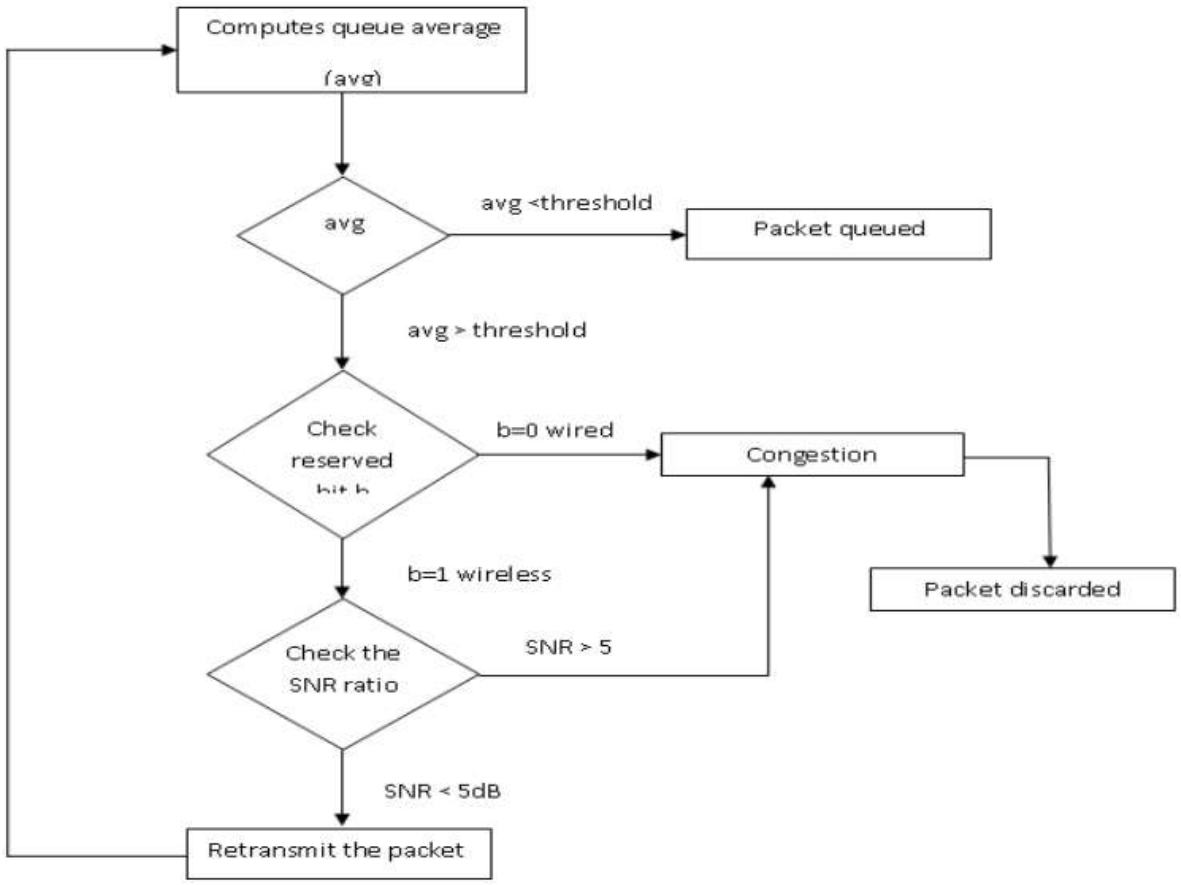
Saved variables

avg: average queue size
q_time: start of the queue idle time
count: packets since last marked packet

Fixed parameters

wq: queue weight
minth: minimum threshold for queue
maxth: maximum threshold for queue
maxp: maximum value for pb

Flow chart for RED Algorithm:-



5. TCL SCRIPT FOR RED ALGORITHM

#Mengke Li RED

#Simulation the RED queue to get the average queue size of RED,

#the packet mark probability, and the bandwidth.

set ns [new Simulator]

#define the RED parameter

Queue/RED set thresh_ 50

```
Queue/RED set maxthresh_ 100

Queue/RED set mean_pktsize_ 500

Queue/RED set q_weigth_ 0.002

Queue/RED set linterm_30

Queue/RED set drop_rand_ true

# Create a simple 22 nodes topology:

# s1 d1

#  \/

# 100Mb,2ms  \10Mb,100ms / 100Mb,5ms

# ... r1 --------- r2 ...

# 100Mb,2ms /  \100Mb,5ms

# s10 d10

#set the two RED queue router

set node_(r1) [$ns node]

set node_(r2) [$ns node]

#define the source and sink node number

set nodenum 10

#define the simulation time

set finish_time 100

#create node_(s1) to node_(s10) as the source node

#create node_(d1) to node_(d10) as the sink node
```

```
for {set i 0} {$i < $nodenum} {incr i} {

set node_(s$i) [$ns node]

set node_(d$i) [$ns node]

}
```

6. CONCLUSION

This paper presented a proposed RED algorithm which solves the congestion avoidance problem. Implement a RED algorithm using TCL Script. Its aim is to allow the TCP protocol to distinguish between transmission timeouts due to congestion and those due to error proactively. The goal of this project is to avoid the congestion proactively. In existing work classical slow-start algorithm is used with SNR ratio and the reserved bits of TCP. Slow-start algorithm waits for some time to receive the acknowledgement from the receiver. In this proposed RED algorithm shouldn't wait for any acknowledgement. It will avoid the congestion proactively.

7. References

[1]Youssef Bassil "TCP Congestion Control Scheme for Wireless Networks based on TCP Reserved Field and SNR Ratio", IJRRIS, June 2012.

[2]Alexander K¨uchler, Matthieu-Patrick Schapranow "Congestion Control", COMMUNICATION NETWORKS, winter semester 2004/2005 Seminar.

[3]V.Jacoban "Congestion Avoidance and Control", Proc. SIGCOMM '88, Vo1 18 No. 4, August 1988

[4]Renu Dangi and Neeraj Shukla" A New Congestion Control Algorithm for High Speed Networks", International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 2, Issue 1. Dr. Z. Huang "Congestion Control", TELE202 Lecture 8

[5]Peterson, Davie and Morgan Kaufmann "Congestion Control and Resource Allocation", Computer Networks A Systems Approach, 2007.

[6]S Chen "Congestion Control Overview", ELEC3030 Computer Networks.

[7]Sridhar Madipelli, David Raj Gillella and Sudhakar Devaraya "The RED Algorithm – Averaged Queue Weight Modeling for Non Linear Traffic"