SHIV SHAKTI

International Journal of in Multidisciplinary and Academic Research (SSIJMAR)

Vol. 3, No. 5, October 2014 (ISSN 2278 – 5973)

TCP CUBIC- Congestion Control Transport Protocol

Gulshan Amin Gilkar
Department of CSE
SITM.
amingulshan9@gmail.com

Syed Yasmeen Sahdad
Department of CSE
SITM.
jasmine.shahdad80@gmail.com

Impact Factor = 3.133 (Scientific Journal Impact Factor Value for 2012 by Inno Space Scientific Journal Impact Factor)

Indexing:

ABSTRACT-This paper enlightens the network congestion and how TCP CUBIC- A variant of transport control protocol (TCP) increases the performance of congestion control mechanism in a long fat network. The paper also presents the algorithm used in LINUX for changing the congestion window to cubic function. We have also tried to analyze the various advantages of TCP CUBIC as well as some of its limitations.

1. INTRODUCTION

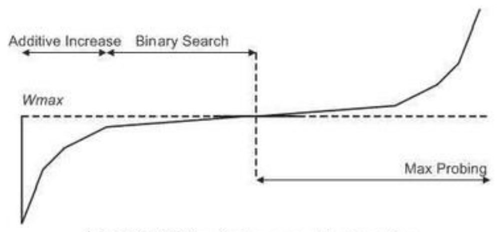
Sometimes the amount of data increases tremendously in a network causing many problems referred to as Network Congestion. Network Congestion refers to a network state where a node or link carries so much data that it may deteriorate network service quality, resulting in queuing delay, frame or data packet loss and the blocking of new connections. In a congested network, response time slows with reduced network throughput. Congestion occurs when bandwidth is insufficient and network data traffic exceeds capacity. [1]Increasing load results in the loss of data packets which continue to keep the network in the state of congestion as the network protocols try to compensate the loss of packets by the process of retransmission.

TCP had been used for transmission over the internet for decades now but doesn't seem to continue to serve for much time now due to the rapidly changing communication environment [3]. For wired networks its performance is still good to some extend but for wireless networks where data loss is comprised of several factors its performance becomes poor Using large bandwidths is a major for TCP as its congestion control mechanisms requires a very long time to examine a long fat network. Researchers have suggested a no of solutions for the above mention problems. The most common way is to change the congestion window. Congestion window is maintained by the sender and refers to the factor that determines number of bytes that are yet to be transferred at any time in the established link [2]. These are referred as TCP variants and have resulted in the increased performance of TCP. These include Stream Control Transmission Protocol (SCTP), CUBIC Transmission Control Protocol (CUBIC TCP), and Compound Transmission Control Protocol (CTCP).

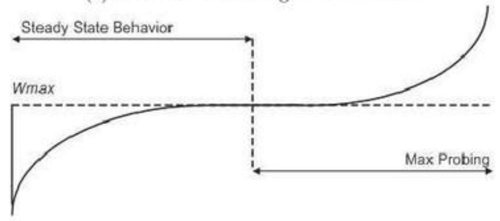
In this paper our attempt is to review an important variant of TCP known as TCP CUBIC. Also, we study the TCP CUBIC and the algorithms for its implementation and how the change in congestion window increases performance of TCP.

2. CUBIC – A TCP VARIANT

CUBIC is a high speed variant of standard TCP. To solve the problem of low performance and bandwidth delay product which TCP has, it uses a cubic function instead of a linear congestion window function for congestion control mechanism in order to improve scalability and stability under fast and long distance networks. The main



(a) BIC-TCP window growth function.



(b) CUBIC window growth function.

The main feature of CUBIC is that its window growth function is defined in real-time so that its growth will be independent of RTT. It enhances the fairness properties of BIC while retaining its scalability and stability. In CUBIC, we try to find the balance between the congestion window size before windows reduction and after windows reduction. Despite this, although the real-time increase of the window enormously enhances the TCP friendliness of the protocol, in short RTT network, CUBIC's window growth is slower than TCP. So in order to keep the growth rate the same as TCP, CUBIC uses a new TCP mode to help change this situation. In CUBIC's TCP mode, it uses the same congestion control mechanisms TCP while the RTT is short.

3. CUBIC TCP ALGORITHM IN LINUX

Before proceeding, we briefly describe the Cubic TCP algorithm used in Linux. Cubic-TCP combines the basic ideas first proposed in High-Speed TCP, and H-TCP. Namely, the cwnd additive increase rate is a function of time since the last notification of congestion (as in H-TCP), and of the window size at the last notification of congestion (similarly to HS-TCP). Pseudo code for the main functionality of the Cubic algorithm is shown in Algorithm 1. The features of this algorithm can be summarized as follows,

1) **SLOW START.** A modified slow start behaviour is Employed at startup. Once cwnd rises above thresh.

Algorithm 1 : Pseudo code of main functionality in Linux 2.6.18 Cubic algorithm

```
1: Initialize:
2: last max = 0; loss cwnd = 0; epoch start = 0; ssthresh = 100
3: b = 2.5; c = 0.4
4:
5: On each ACK:
6: delay min = min(RTT, delay min)
7: if cwnd < ssthresh then
8: cwnd++ //slow start
9: else
10: if epoch start = 0 then
11: epoch start = current time
12: K = max(0, 3p(b _ (last max − cwnd)))
13: origin point = max(cwnd, last max)
14: end if
15: t =current time +delay min − epoch start
16: target = origin point + c _ (t − K)3
17: if target > cwnd then
18: cnt = cwnd/(target − cwnd)
19: else
20: cnt = 100 _ cwnd
21: end if
22: if delay min > 0 then
23: cnt = max(cnt, 8 _ cwnd/(20 _ delay min)) //max AI rate
24: end if
25: if loss cwnd == 0 then
26: cnt=50 // continue exponential increase before first backoff
27: end if
28: if cwnd cnt > cnt then
29: cwnd++
30: cwnd cnt = 0
31: else
32: cwnd cnt++
33: end if
34: end if
35:
36: On packet loss:
37: epoch start = 0
38: if cwnd < last max then
39: last max = 0.9 _ cwnd
40: else
41: last max = cwnd
42: end if
43: loss cwnd = cwnd
44: cwnd = 0.8 _ cwnd // backoff cwnd by 0.8
```

(which is initalised to a value of 100 packets in Cubic), Cubic exits normal slow start and changes to use a less aggressive exponential increase where cwnd is increased by one packet for every 50 acks received or, equivalently, cwnd doubles approximately every 35 round-trip times.
See lines 25-26 in Algorithm 1.

2) Backoff factor 0.8**.** On packet loss, cwnd is decreased by a factor of 0.8 (compared with a factor of 0.5 in the standard TCP algorithm). See line 44 in Algorithm 1.

3) Clamp on maximum increase rate**.** The additive increase rate during AIMD operation is limited to be at most 20_delay min packets per RTT, where delay min is an estimate of the round-trip propagation delay of the flow.
See line 23 in Algorithm 1. Converting from packets per RTT to packets per second, this clamp is roughly equivalent to a cap on the increase rate of 20 packets independent of RTT.

4) INCREASED CUBIC FUNCTION. Subject to this clamp, the additive increase rate used is target−cwnd packets per RTT. Note that the effect of this increase is to adjust cwnd to be equal to target over the course of a single RTT. The value of target is calculated (see line 16 in algorithm) from: target = Wmax + c(t − 3p(b(Wmax − 0.8W))3 (1) where t is the elapsed time since last back off (approximately– the value of delay min is added to this value, see line 15) and Wmax is related to the cwnd at last back off and is denoted origin point in the code. W is the cwnd value immediately before the last back off, so that 0.8W is the cwnd value just after backoff has occurred.

5) ADAPTATION OF CUBIC FUNCTION**.** The value of Wmax is adjusted depending on whether the last backoff occurred before or after cwnd reached the previous Wmax value. Let W denote the cwnd value immediately before backoff. Then, Wmax is set equal to the W when W is larger than the previous value of Wmax. Otherwise Wmax is set equal to 0.9W. See lines 38-42 in algorithm.

4. Advantages

CUBIC TCP ia a nice solution for BDP network. With the development of Internet, the route trip times usually become very high (around 100 to 200ms). In this case, standard TCP has a low utilize radio. Nowadays many High Speed TCP variants came out and tried to fix this problem. But the difficulties are not only raising the efficiency, but also maintain the RTT fairness and TCP friendly. CUBIC works much better than those previous high speed TCP variants. And now, CUBIC TCP is implemented and used by default in Linux kernels 2.6.19 and above.

5. Limitations

Although CUBIC TCP seems a nice solution for high speed transport layer, there are some limitations of it. CUBIC TCP suffers from slow convergenceyields poor network responsiveness, prolonged unfairness between flows, increases unfairness between long and short lived flows. CUBIC TCP is a good solution for standard Internet environment, but it is not aimed for special usage, such as satellite network or mobility usage.

6. Conclusion

The information presented in this paper establishes that TCP CUBIC increases the performance in the network and provides a good mechanism for congestion control.

References

[1] Techopedia

[2] Wikipedia

[3] P. G. Bridges,G.T.Wong, M. Hiltunen, R. D. Schlichting, and M. J. Barrick. A configurable extensible transport protocol. IEEE/ACM Trans.Netw.,15(6):1254–1265, 2007.

[4] S. Iren, P. D. Amer, and P. T. Conrad. The transport layer: tutorial and survey. ACM Comput. Surv.,31(4):360–404, 1999.

[4] D.J. Leith, R.N.Shorten, G.McCullagh,Hamilton Institute, Ireland

[5] Jiawei Chen Helsinki University of Technology