Aggregate TCP Congestion Management for Internet QoS

Hemkumar D
Department of Computer Science & Engineering
National Institute of Technology, Karnataka, Surathkal
Mangalore, India
e-mail: 1986.hem.kumar@gmail.com

K. Vinaykumar
Department of Computer Science & Engineering
National Institute of Technology, Karnataka, Surathkal
Mangalore, India
e-mail: vinaykumarin2000@yahoo.com

***Abstract*— In this project we study the one of the most important performance in aggregate congestion control is fairness, i.e. the equal use of resources. The objective of Aggregate TCP congestion management is to achieve the fair sharing of the bottleneck bandwidth between the aggregate and other background TCP flows. There are many existing mechanism to prove the fairness property but none of mechanism proves fair share of the aggregated flows even where the number of flows in the aggregate is large. To solve this problem, unlike existing mechanism there are two congestion window loops, one is that loss event rate loop and other one is throughput control loop. Throughput control loop is used to adjust its window size with a weight N. Finally our simulation results prove an algorithm providing for fairly services to TCP flows that share a bottleneck link between the aggregate and other background TCP flows even where the number of flows in the aggregate is large.**

Keywords- TCP, Fairness, Congestion Control

I. INTRODUCTION

There are many existing protocols is performed in end-to-end measurement like SCTP[8], DCCP, TCP. These protocols are maintained for managing network information available from other flows, especially the information from other flows sharing the same bottleneck. This might result in low efficiency of resource utilization.

Aggregate TCP flow management mechanism is used for improving network utilization. Its main idea is to cooperation between flows sharing the same bottleneck. Multiple flows sharing the same bottleneck are aggregated into a single flow or a smaller number of flows in an aggregate flow management scheme. Thus a grain of resource management is performed, while details of individual flows are hidden. Aggregate flow management is applied at a route and flows traversing a bottleneck link. These flows are controlled by the end host or a router, rather than independently. Aggregate flow management has also been applied in various applications, such as edge-to-edge QoS overlay services, a coordination protocol for distributed media applications, wireless TCP proxies for addressing network heterogeneity, and an overlay network architecture providing DoS-limiting as well as resilience on network edges. And also it could be used in modern storage networks in which a huge amount of block data is exchanged over a single TCP connection.

In this mechanism one of the most important properties in aggregate congestion control is fairness, i.e. the equal share of resources. Flowshare was defined for characterizing fairness in aggregate congestion control. A flowshare is defined as the bandwidth utilized by a one congestion controlled flow, i.e. a one TCP connection.
In the aggregate TCP flow management, one standard TCP congestion window loop is used for the aggregate. Both the aggregate and each of the m background flows are allocated $\frac{1}{m+1}$share of the bottleneck link bandwidth, and also there are N flows within the aggregate. This is not equally share, since the share of the bottleneck bandwidth allocated to each flow within the aggregate TCPs is $\frac{1}{(m+1)N}$, while that allocated to each background flow is $\frac{1}{m+1}$. In figure 1, N flow shares should be allocated to an aggregate having N flows under a good aggregate congestion control mechanism, i.e. the share of the bottleneck bandwidth allocated to the aggregate TCPs should be $\frac{N}{N+m}$. Therefore, the aggregate congestion control should support multiple flow shares so as to ensure a fair share of the bottleneck link bandwidth for the aggregate.

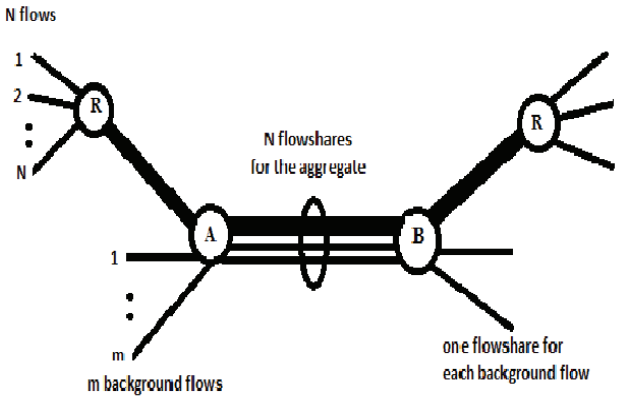

Figure 1: The aggregate traffic with the bandwidth of multiple flow shares

978-0-7695-4817-3/12 $26.00 © 2012 IEEE
DOI 10.1109/ICCS.2012.10

The contribution of this paper is to improve fairness of the bottleneck bandwidth in aggregate TCP congestion control mechanism; it extends MulTCP[6] with one more window, which improves fairness without the need of managing numerous window states. Our simulation results show that aggregate can hold its fair share of the bottleneck bandwidth while other TCP flows are also treated fairly.

The outline of this paper is as follows. Related techniques supporting multiple TCP flows in aggregate congestion control are explained in Section 2. Our design of proposed mechanism is explained in Section 3. Our simulation results are shown in Section 4, finally, our conclusions are explained in Section 5.

II. Related Work

There are many protocols proposed for improving fairness by multiple TCP flowshares, e.g. MulTCP , CP and MPAT. The Additive Increase Multiplicative Decrease (AIMD) algorithm is the congestion control mechanism used in standard TCP. It is modified in MulTCP[6] for enabling an aggregate to show the behaviour of multiple concurrent TCP connections. And it has been demonstrated that beyond small values of *N*, a MulTCP flow fails to behave like *N* independent TCP flows. In other words, the throughput of MulTCP does not increase proportionally to the number of flows within the aggregate, especially when $N > 4$. And also noted in that, loss behavior of a single MulTCP[6] flow is quite different from that of *N* independent TCP flows. This leads to an increasingly unstable window adjustment as *N* increases. Hence it is clear that the MULTCP[6] not satisfy the fairness property, when *N* is larger than four.

Another existing mechanism MPAT, the number of congestion window loops in MPAT is the same as the number of flows within the aggregate. As a result, the several number of congestion window states are maintained. And also, each congestion window loop in MPAT probes the network condition independently, which results in competition among different congestion window loops. This may leads to degrade performance.

III. Design

As we know that, none of the existing mechanisms does not prove fairly share the bottleneck bandwidth between the aggregate and other background flows. Based on the drawbacks of existing mechanisms i.e., congestion window adjustment. Instead of one window loop we are maintaining two congestion window loops. These two AIMD window loops are not maintained two separate connections, i.e. one MulTCP connection and one connection. Instead of two connections we are maintained inside a single connection. The one standard TCP loop acts as a loss event rate detector, it calculates the loss event rate of a standard TCP flow. Other one act has a throughput controller. It adjusts its window size with a aggregate *N* TCPs, which is similar to that in MulTCP[6]. The maximum window size is calculated based on the loss event rate that is detected by the other window loop.

The maximum window size is calculated from the loss event rate measured by the probe window loop. Here AIMD (1, 1/2) is used for the probe window loop. The measurement of loss rate is also important to the window loop. Loss rate should be measured as loss event fraction, whereas loss event consists of one or more packets dropped within a single RTT. A loss event rate is calculated by constructing a loss history and identifying loss event rate. But the loss event rate is calculated on the sender's side, rather than on the receiver's side.

Other window loop is used for throughput controller. Based on the loss event rate, it adjusts its congestion window size with a weight N, which is similar to that in MulTCP[6], but under a limitation on its maximum window size. The maximum window size is measured based on the loss event rate that is detected by the loss rate window loop. Here, AIMD ($\sqrt{N}, \frac{2}{1+3N^{1.5}}$) is used for the throughput controller window loop. Another challenge in the design of the throughput controller window loop is how to adjust the congestion window size, given the loss rate that is calculated by the loss rate window loop. These two AIMD window loops are not maintained two separate connections i.e. one MulTCP[6] connection and one standard TCP connection. Instead of two connections we maintained inside a single connection.

The proposed mechanism is based on one of the existing mechanism i.e., MulTCP, MulTCP is a collection of multiple virtual TCPs. The working procedure of MULTCP[6] is that during slow start a TCP opens its congestion window exponentially and by sending two packets for every acknowledgement received. In MULTCP[6] achieved the same behavior, if it sent N packets at start and then two packets for every acknowledgement received. If N is large, this leads to large amount of losses. So MulTCP uses a smoother option. It starts sending a single packet like a normal TCP. After that, it sends three packets for each acknowledgement received until it has opened its congestion window as far as N TCPs would have.

Crowcroft and Oechslin proposed MulTCP, AIMD($N, \frac{1}{2N}$) is used in MulTCP, instead of AIMD(1, 1/2) in the standard TCP, for achieving the behavior of N concurrent TCP connections using the same congestion window loop.

The parameters behind AIMD ($N, \frac{1}{2N}$) ,The congestion window size of a single TCP flow is increased by one packet per RRT in the standard TCP, similarly the congestion window size of *N* virtual TCP flows is increased by *N* packets per RRT in MulTCP then the Increase parameter $a = N$.

If one packet is lost, only one of the N TCP flows half of its congestion window size in the standard TCP. Simillarly, the congestion window size becomes $(1 -$

$\frac{1}{2N}$)NW in MulTCP[6] then the Decrease parameter b = $\frac{1}{2N}$. In MULTCP it uses the parameter AIMD ($N, \frac{1}{2N}$) but in throughput control window loop, it uses the parameter of AIMD ($\sqrt{N}, \frac{2}{1+3N^{1.5}}$).

After K round trip times N TCPs have a congestion window of N∗ 2^K . One MulTCP sending three packets for each acknowledgement would have a window of 3^K. Thus they have the same window after K_N round trip times where $K_N = \frac{\log N}{\log 3 - \log 2}$

```
If (cwnd < ssthresh)
  {
If (cwnd<=pow(3.0,1og(√N)/(log(3)-log(2))))
      cwnd += 2;
      Else
      cwnd += 1;
  }
```

It happens only when the window has a size of $W_n = 3^{K_N}$ when packet is loss, it half of its congestion window and sets slow start threshold to the new value of the congestion window and goes back to linear increase. When N TCPs are sending data and one packet is lost, only one TCP will half of its window. Thus MulTCP[6], when it happens loss, only half one Nth of its congestion window and by setting cwnd and ssthresh to $\frac{(N-0.5)}{N_2}$ of cwnd. But we are used parameter AIMD ($\sqrt{N}, \frac{2}{1+3N^{1.5}}$), so set the ssthresh to $\frac{(3N^{1.5}-1)}{(3N^{1.5}+1)}$ of cwnd.

```
If (cwnd < ssthresh)
   cwnd = cwnd/2;

Else
   cwnd = cwnd *(3N^1.5−1)/(3N^1.5+1);
   ssthresh = cwnd;
```

When there are too many losses within one RTT, timeouts occur and transmission restarts with a slow start after the last acknowledged packet. N TCPs are less timeout than that of one MulTCP[6]. Since the losses are distributed over N connections and the probability that one TCP experiences enough losses within one RTT to make it is smaller. And if one TCP should stall, the (N-1) others can still go on sending. Thus after the slow-start is over the MulTCP will have the same window as N TCPs would after one of them has done a slow-start. This ensures that the loss rate loop experiences a loss event rate, which is similar to that of standard TCP flow sharing the same bottleneck link.

So this mechanism is designed to improve efficiency and fairness of the aggregate congestion control, while achieving the desired throughput given the weight N for an aggregate.

IV. EVALUATION

Our simulation is performed on ns-2[1]. The Aggregate TCP flow management mechanism is implemented based on MulTCP[6], A network topology having only one congested bottleneck link is used, which has 10ms delay and drop-tail queue management. The buffer size is always set to the delay-bandwidth product. The packet size is 1500 bytes. MulTCP cannot provide fair sharing for the aggregate. We will show that proposed mechanism improves fairness property. Our simulations demonstrate that, aggregate TCPs competes with *m* standard TCP flows over a shared bottleneck link, where *N* spans from 2 to 30 .The bandwidth of the bottleneck link is set to 3 Mbps and The simulation time is 200 seconds.

We ran 5 aggregates, each with a different value of N, together with respective background TCP flows. The five aggregates use N = 10, 14 and 20. We then replaced each aggregate with the different number of independent TCP flows. Figure 2 shows the bandwidth for each of the five aggregates, with data points sampled every 1sec.

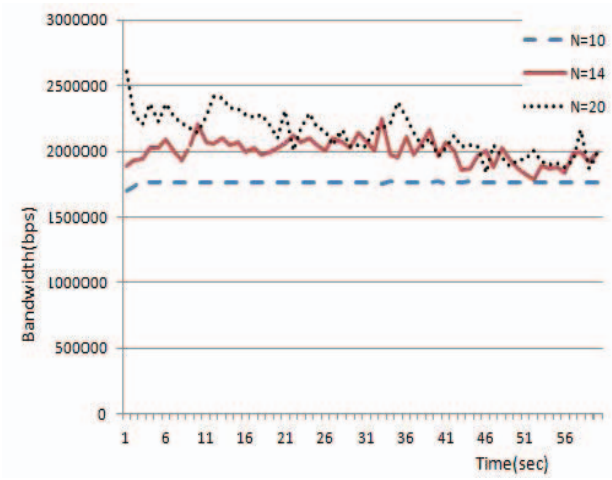

Figure 2: Bandwidth utilization of aggregate N TCP flows

Figure 3 shows fairness index under different weight N. Table contains number of aggregate TCP flows along with number of background flows, throughput of aggregate and throughput of background flows. We define fairness index as the performance ratio for calculating fairness of the long-term sending rate. It is the average throughput of N flows within the aggregate divided by the average throughput of m background standard TCP flows:

No.Aggre-TCP	No.BG-TCP	Aggre-Throughput	BG-throughput	Fairness Index
12	6	1899975	973145.7	0.98
14	6	1973789	827015.1	1.022
16	7	2018955	878030.2	1.006
18	7	2049769	785206	1.0151
20	7	2205317	750834.2	1.02
24	11	1980422	868663.3	1.04

Figure 3: Fairness index under different weight N

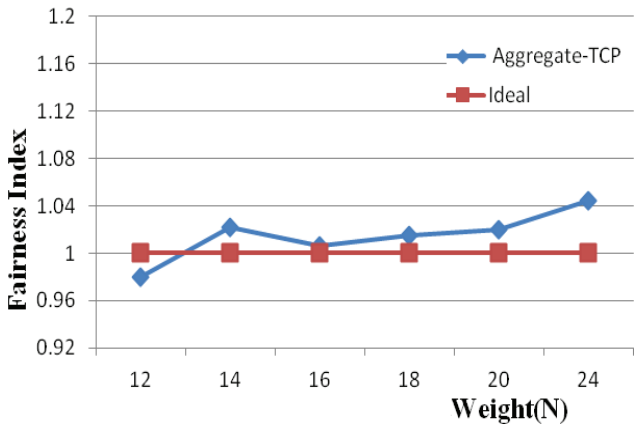

Figure 4: Fairness property of aggregate N TCP flows

$$\text{Fairness index} = \frac{\frac{T}{N}}{\frac{1}{m}\sum_{i=1}^{m} T_{tcp_i}}$$

Where T is the throughput of the aggregate and T_{tcp_i} is the throughput of the ith background standard TCP flow. If the fairness index is equals one, it means that the aggregate TCP's fairly shares the bandwidth with the other background TCP flows. If the fairness index is less than one, the aggregate losses the bandwidth to the background TCP's. Otherwise, the aggregate is more aggressive than the background TCP flows. Figure 4 shows that the aggregates i.e., N=12,14,16,18,20 and 24 are approximately equal to ideal. So we prove that, to achieve the fair sharing of the bottleneck bandwidth between the aggregate and other background TCP flows, where the number of flows in the aggregate is large.

V. CONCLUSION

The paper demonstrates that providing differential services, among a group of TCP flows that share the same bottleneck link. Single flowshare in aggregate congestion control mechanism results in unfairness between the aggregate and the background flows, which are sharing the same bottleneck. There are many existing mechanism to prove the fairness property but none of mechanism proves fair share of the aggregated flows even where the number of flows in the aggregate is large. But this paper concludes this challenge. We demonstrate on ns-2 simulation that fair sharing of the bottleneck bandwidth between the aggregate and other background TCP flows. That result is fairer than MulTCP[6] over a wide range of the weight N, only changes congestion condition. This Mechanism can adjust the bandwidth maintain and proportional bandwidth within the aggregate. Aggregate congestion control mechanism could be applied to a wider range of scenarios, such as mobile or wireless TCP proxies, edge-to-edge overlays QoS [3] provisioning and mass data transport in storage networks.

VI. REFERENCES

[1] NS simulator, version 2.29. http://nsnam.isi.edu/nsnam/.

[2] F.-C. Kuo and X. Fu, "Probe-Aided MulTCP: An Aggregate Congestion Control Mechanism," Institute for Informatics, University of Goettingen, Technical Report IFI-TB-2007-01, 2008.

[3] L. Subramanian and I. Stoica and H. Balakrishnan and R. Katz, "OverQoS: An Overlay Based Architecture for Enhancing Internet QoS," Accepted for publication in First Symposium on Networked Systems Design and Implementation (NSDI), March 2004.

[4] H. Y. Hsieh and K.H. Kim and R. Sivakumar, "On Achieving Weighted Service Differentiation: an End-to-end Perspective," in Proceedings of IEEE IWQoS, June 2003.

[5] H. Balakrishnan and H. S. Rahul and S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts," in Proceedings of ACM SIGCOMM, pages 175-187, Sept 2002.

[6] J. Crowcroft and P. Oechslin, "Differentiated End-to- End Internet Services using a Weighted Proportional Fair Sharing TCP," ACM Computer Communication Review, 28(3):53-69.

[7] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, "Stream Control Transmission Protocol," Internet Engineering Task Force, October 2000. RFC 2960.

[8] M. Singh, P. Pradhan and P. Francis, "MPAT: aggregate TCP congestion management as a building block for Internet QoS," in Proceedings of IEEE International Conference on Network Protocols, pages 129– 138, Oct. 2004.