International Journal of Engineering Inventions
ISSN: 2278-7461, ISBN: 2319-6491
Volume 2, Issue 1 (January 2013) PP: 46-50

Fairness in Transfer Control Protocol for Congestion Control in Multiplicative Increase and Multiplicative Decrease

Sarika Gupta Agarwal[1], Parul Gupta[2], Mamta Narwaria[3]
[1,3]Department of Computer Science, Greater Noida-201308 India
[2]Department of Computer Science, Mumbai Maharashtra, India

Abstract:- High speed Network(Scalable TCP) uses Multiplicative Increase Multiplicative Decrease (MIMD) algorithm . This paper investigates fairness among sessions sharing a common bottleneck link, where one or more sessions use the MIMD algorithm. Congestion occur when Traffic reached Beyond the capacity of window. We first study how two MIMD sessions share the capacity in the presence of generalcombinations of synchronous and asynchronous losses. We show that, in the presence of rate dependent losses, the capacity is fairly shared whereas rate independent losses provide high unfairness.

Keyword:- MIMD, Scalable TCP, AIMD,CUBIC,WESTWOOD,Fairness

I. INTRODUCTON

TCP provides the mechanisms that provide data to be transferred across networks that are dynamic and have a large variety of resources. Until now the AIMD algorithm was found to provide satisfactory performance.However, in high speed networks , the additive increasing the sender's rate may lead to inefficient link utilization[1].To overcome this drawback in high speed networks, the MIMD algorithm has been proposed as an alternativeto the AIMD algorithm In the future, situations may arise when sessions using these two algorithms would compete for the same network resource. The share of the capacity obtained by each of these sessions will depend on the various parameters specific to the algorithms. The sharing of a resource gives rise to the question of how fairly is this resource shared among the sessions. Fairness issues have been addressed in several previous works. In [4], the authors considered a class of rate control algorithms in the presence of synchronous control signals. They showed that the AIMD algorithm converged to fairness whereas the MIMD algorithm did not converge. In [5], the author studied the MIMD algorithm under a more realistic assumption of rate dependent losses and argued that MIMD algorithm also converges to fairness. In [6] the convergence to fairness ofthe different flavors of TCP was studied both analyticallyand using simulations. The monotonic convergence to fairness for algorithms in rate-based TCP-friendly applications was studied in [7].

II. BASIC CONCEPTS OFTCP CONGESTION CONTROL ALGORITHMS

2.1 AIMD_Algorithim

The data transfer of TCP starts from a *slow start*, in which TCP tries to increase its sending rate exponentially, until it encounters the first loss. It then switches to another stage, called *congestion avoidance*, in which TCP employs the *Additive Increase*, *Multiplicative decrease* mechanism to slowly adapt to the available bandwidth. Further congestion, the TCP goes into the *Fast Recovery &Fast Retransmission* stages. In this scenario, when TCP do not receive an acknowledgment for a packet after some timeout period, it assumes that this packet is lost. & then retransmits that packet and doubles its retransmission timeout value (RTO) detecting packet loss. This process continues until the packet is successfully transmitted & acknowledged. TCP tries to clear congestion by cutting its sending rate in half.

2.1.1 Additive Increment

After receiving an ACK for new data, congestion window is increment by $(MSS)^2/Cwnd$, where MSS is maximum segment size, this formula is known as additive increment. The goal of additive increment is to open congestion window by a maximum of one MSS per RTT. Additive increment can be described by using the equation (1):

Cwnd = Cwnd + a*MSS2/Cwnd Eq (1)

Where the value of a is a constant, a = 1.

2.1.2 Multiplicative Decrement

Multiplicative decrement occurs after a congestion event, such as a lost packet or a timeout. After a congestion event occurs, the slow start threshold is set to half current congestion window. This update to slow start threshold follows equation (2)

ssthresh = (1 – b)*CWND

Eq(2)

CWND is equal to amount of data that has been sent but not yet ACKed and b is a constant, b = 0.5. The congestion window is adjusted accordingly. After a timeout occurs, congestion window is set to one MSS and slow start algorithm is reuse.

2.2Multiplicative Increase and Multiplicative Decrease(MIMD)

Scalable TCP uses a different congestion avoidance algorithm than Standard TCP. Scalable TCP uses a multiplicative increment multiplicative decrement (MIMD) rather than the AIMD of Standard TCP

2.2.1 Multiplicative Increment

The multiplicative increment occurs when standard additive increment would normally occurs. In equation (8) shows the formula used to adjust congestion window after receiving a new ACK.

Cwnd = Cwnd + a*Cwnd Eq. (3)

Where a is adjustable, the value of a used was 0.02

2.2.2 Multiplicative Decrement

The multiplicative decrement is same as Standard TCP except that the value of b in equation (2) is adjustable, the value of b used 0.125. The connection starts in the slow start algorithm until channel is filled. The connection uses the multiplicative increment portion of congestion avoidance to adjust congestion window. After a single drop occur around 1.4 seconds, fast retransmit and recovery algorithms are used to cut congestion window by 0.125, the value of b, and congestion avoidance is used again to reopen congestion window

III. COMPARISON BETWEEN STANDARD TCP AND SCALABLE TCP

Table 1: Comparison between Standard and Scalable TCP

	Standard TCP(AIMD)	**Scalable TCP(MIMD)**
no losses	Wn+1=Wn+1= linear increase dw/dt=1/T	$\mathbf{W_{n+1=}\propto * W_n=}$ multiplicative increase **dW/dt=log[∝]/T*W** =exponential growth
≥1 loss	$\mathbf{W_{n+1}=0.5*W_n}$ **multiplicative decrease**	$\mathbf{W_{n+1}=\beta * W_n}$ **multiplicative decrease**

IV. FAIRNESS IN MIMD SESSIONS (EQUAL RTTS)

We consider two sessions which share a link of capacity *C*. At time *t*, the rates obtained by the two sessions are denoted by x(*t*) ≡ (x1(*t*), x2(*t*)). At each control instant, the controller sends a control signal to each source. This control signal either informs on no congestion (a 0 signal) or of congestion (a 1signal). In the absence of congestion, the sources increase theirrate exponentially xk(*t* + *τ*) = *ατ/τ*0 · xk(*t*), *k*= 1, 2,

Table 2: (REACTION TO CONTROL SIGNALS)

control vector	**x1(*tj*+)**	**x2(*tj*+)**
(0,0)	x1(*tj*)	x2(*tj*)
(0,1)	x1(*tj*)	*β* · x(*tj*)(A Synchronous Congestion)
(1,0)	*β* · x1(*tj*)	x2(*tj*) (A Synchronous Congestion)
(1,1)	*β* · x1(*tj*)	*β* · x2(*tj*)(Synchronous Congestion)

Whereτ0 is the time constant (for example, the RTT) for the sessions, and *α* >1 is the increase factor The above Formulation is a continuous time equivalent of a multiplicativealgorithm in which, for every RTT without congestion signals, the sender multiplies the window by a factor of *α*. This can be seen by substituting *t* = *nτ*0. We assume that the two sources receive the control signals at the same instant. However, unlike the model in [4], the two sources can receive different control signals. That is, a congestion signal need not be sent to both the sources at the same instant. Hence, the congestion signals could be asynchronous. Let *β* <1 be the decrease factor. Let the*jth*control signal be received at time *tj*. Then, the four possibilities for the rate vector, x(*tj*+), just after *tj*, are givenin Table 1 The source continues with the increase algorithm on the reception of 0 signal. On the other hand, when a source receives a 1 signal, it instantaneously reduces its rate.

V. INTER PROTOCOL FAIRNESS (SAME RTT)

In this we study the fairness issue when sessions using two different congestion control algorithms share a common link, and the losses are synchronous. Recently Scalable TCP, which uses the MIMD algorithm, has been proposed as an enhancement for TCP in high-speed networks. Situations may, therefore, arise in which a user with Scalable TCP shares a link with a user with standard TCP. Specifically, we study the equilibrium behaviour of the window size, and the throughput obtained by a session of each algorithm at equilibrium in the presence of synchronous losses
only. We also look at conditions under which a user of one algorithm can obtain a better throughput than a user of the other algorithm. Previous work (e.g., [6], [8]) mainly studied the behaviour of sessions using the same algorithm.
In this section, we assume that each session has the same RTT, τ . As mentioned in Section II, window-based notation is equivalent to rate-based notation. In the rest of this paper, we use the window-based notation since we are interested in obtaining the equilibrium window sizes for the sessions

5.1 System Model

Consider *l* sessions which share a link of capacity *C bits/s*.Each session transmits data using packets of size *M* bits. Let Λ be the bandwidth-delay product (BDP) of the network. We assume that each session has the same RTT, τ , and that the RTT is mainly determined by the propagation delay and, hence, can be considered to be a constant. Let x(*t*) = (x1(*t*) x2(*t*) . . . x*l*(*t*)) denote the vector of window sizes of the *k* sessions at time *t*. A synchronous loss (i.e., a loss for each session) is assumed to occur at time *t* if

$$\sum_{i=1}^{l} \mathrm{x}i(t) > \Lambda \qquad \text{Eq.4}$$

The above condition is equivalent to saying that a synchronous loss occurs when the total number of outstanding packets in the network exceeds the total number of packets that the network can handle. Without loss of generality, let sessions 1, 2, *k* use the MIMD congestion control algorithm and the rest of the *l* –*k*sessions use the AIMD congestion control algorithm. In the absence of losses, the two algorithms increase the window in the following way

$\mathrm{x}i(t+\Delta) =\{ \mathrm{x}(t)\alpha\Delta/\tau, \quad 1 \le i \le k$ Eq.5

$\mathrm{x}i(t+\Delta) =\{ \mathrm{x}(t) +, \alpha a\ \Delta/ \tau, \quad k+ 1 \le i \le l,$ Eq.*6*

Whereαm and *αa* are the increase parameters of the MIMD and the AIMD algorithm, respectively. For example, *αm* =1.01 for Scalable TCP, and *αa*= 1 for standard TCP. Let *tn*denote the time instant when the *nth* congestion signal is received. We note that a congestion signal is generated when a synchronous loss occurs.
In response to a congestion signal the two algorithms decrease the window in the following way.

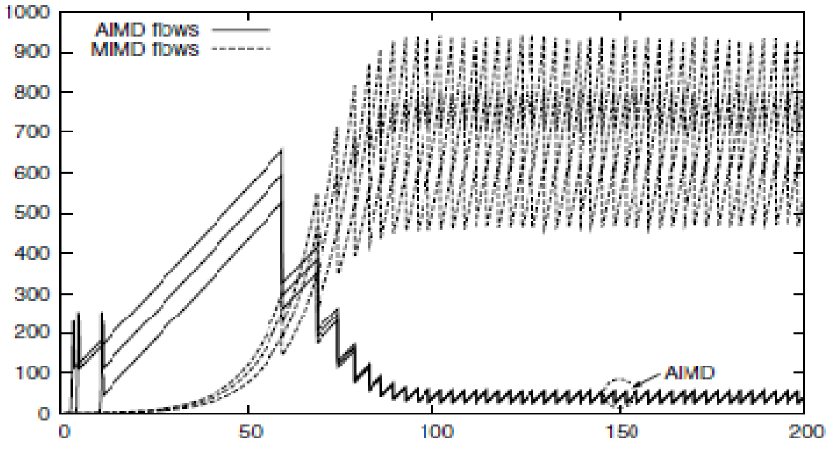
$\mathrm{x}i(t+n) = \{ \beta m\mathrm{x}(tn), \quad 1 \le i \le k,$ *Eq.7*

$\mathrm{x}i(t+n) = \{ \beta a\mathrm{x}(tn), \quad k+ 1 \le i \le l,$ *Eq.8*

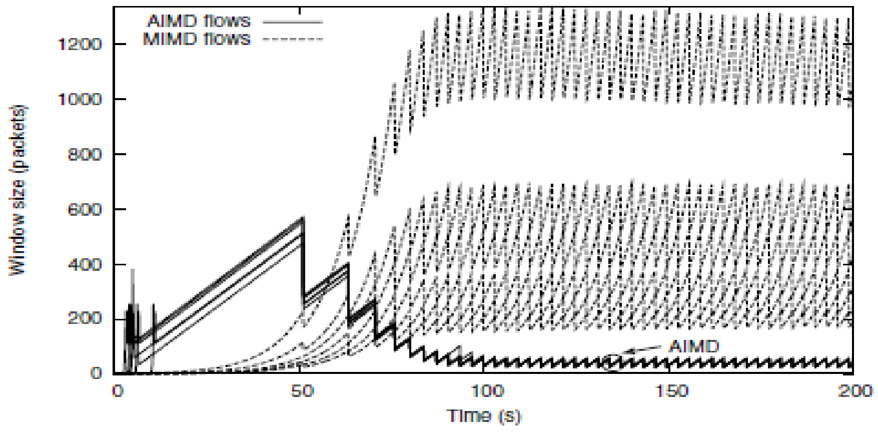
Whereβm and *βa* are the decrease parameters of the MIMD and the AIMD algorithm, respectively. For example, *βm* =0.875 for Scalable TCP, and *βa* = 0.5 for standard TCP.

VI. SIMULATIONS

We now compare these observations with simulations performed using *ns-2* (version 2.26) and wireshark. Unless stated otherwise, the simulation had the same set of parameters. The MIMD sessions used Scalable TCP, and the AIMD sessions used TCP New Reno. The packet size, *M*, for each session was set to 1040 bytes (1000 bytes of data + 40 bytes of header). The propagation delay, *σ*, was taken to be 100*ms*. The increase and decrease parameters for the two algorithms were set to *αm* = 1.01, *αa*= 1.0, *βm* = 0.75, and *βa* = 0.5. Since the *ψi*for AIMD increases with decrease in *βm*, we set *βm* to a value smaller than its recommended value so that the AIMD sessions also obtain a certain throughputwe note that the AIMD sessions indeed converge to the same equilibrium window size whereas the equilibrium window size of an MIMD session depends on its window just before the first synchronous loss. The *ψi*for AIMD sessions remains the same even though the link capacity is increased from 200Mbps to 300Mbps and the total number of sessions is increased from six to twelve. Let $\eta a$and ψadenote the throughput and the equilibrium window size, respectively, of any one of the AIMD sessions.

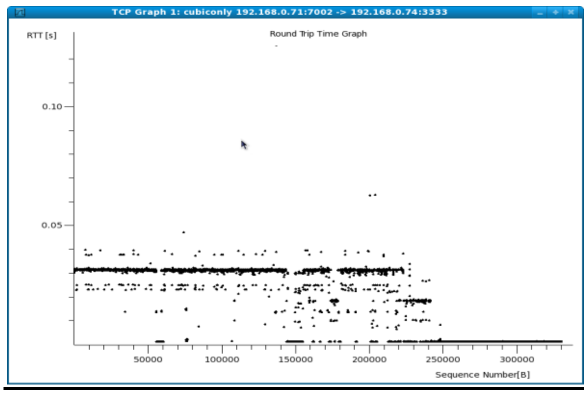


(a) $C = 200\text{Mbps}.\ l = 6.\ k = 3.$



(b) $C = 300\text{Mbps}.\ l = 12.\ k = 6.$

Fig:1 Window Evolution of Sessions[source: Ref.11]



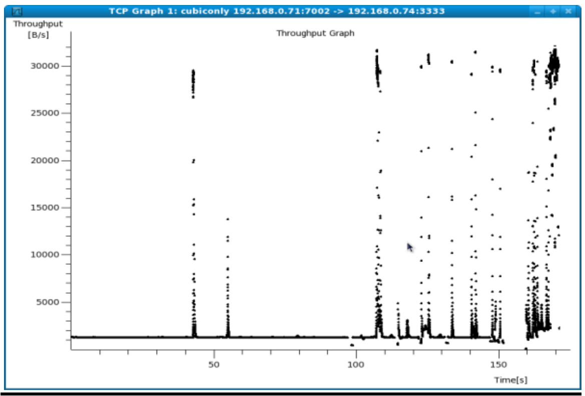


Fig:2RTT Graph for TCP CUBIC+Vegas+Reno Fig:3 Throughput Graph for TCP Cubic+Vegas+Reno

VII. CONCLUSIONS

Scalable TCP implements simple changes to the currently used congestion control algorithm. These changes have both a positive and negative effects on the existing network traffic. Each algorithm provides higher channel utilization for high speed and long delay environment. However, the alternative algorithms do not shares chainnnel equally, when mixed with Standard TCP traffic. In a homogenous environment, the overall channel utilization and sharing between streams increments as compared to a mixed environment. Future work is needed to study the effects of more than two competing streams. In This Paper , we studied the fairness in sessions using MIMD congestion control algorithm. For sessions with the same RTT, it was observed that there was extreme unfairness when the asynchronous losses were rate independent. It was shown that fair sharing could be achieved by introducing a stream of rate dependent losses. For sessions with different RTTs, it was observed that the arrival rate of these rate dependent losses had to be greater than a certain minimum rate in order to achieve fairness. Therefore, in networks with sessions using MIMD algorithms, a stream of rate dependent losses, using, for example, some
buffer management scheme, would be necessary to ensure fair sharing.

REFERENCES

1) KulvinderSingh"Experimental Study of TCP Congestion Control Algorithms",IJCEM International Journal of Computational Engineering & Management, Vol. 14, October 2011 ISSN (Online): 2230-7893
2) S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649, Experimental, December 2003.
3) T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. *Computer Comm. Review*, 33(2):83–91, April 2003.
4) G. Vinnicombe. On the Stability of Networks Operating TCP-like Congestion Control. In *Proceedings of the IFAC World Congress*, 2002.
5) D. Chiu and R. Jain. Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of ComputerNetworks and ISDN*, 17(1):1–14, 1989.
6) S. Gorinsky. Feedback Modeling in Internet Congestion Control. In *Proceedings of the NEW2AN*, February 2004. Also see a techincal report at http://www.arl.wustl.edu/∼gorinsky/ TR2002-39.ps.
7) G. Hasegawa, M. Murata, and H. Miyahara. Fairness and Stability of congestion control mechanisms of TCP. *Telecommunication Systems*,November 2000.
8)]D. Loguinov and H. Radha. End-to-End Rate-Based Congestion Control:Convergence Properties and Scalability Analysis. *IEEE/ACM Transactionson Networking*, 11(4), August 2003
9) L. Xu, K. Harfoush, and I. Rhee. Binary Increase Congestion Control(BIC) for Fast Long-Distance Network. In *Proceedings of the IEEEINFOCOM*, March 2004.
10) A. Budhiraja, F. Hernndez-Campos, V.G. Kulkarni, and F. D. Smith. Stochastic Differential Equation for TCP Window Size: Analysis and Experimental Validation. *Prob. in the Engg. and Informational Sciences*,18:111–140, 2004.
11) E. Altman K., E. Avrachenkov, B. J. Prabhu. Fairness in MIMD Congestion Control Algorithim 2004.