draft-leith-tcp-htcp-01
Internet-Draft
Expires: June 4, 2006

D. Leith
R. Shorten
Hamilton Institute

Dec 2005

H-TCP: TCP Congestion Control for High Bandwidth-Delay Product Paths

Status of this Memo

Copyright Notice

Abstract

Our objective in this document is to renew discussion on how the TCP congestion control algorithm might best be modified to improve performance in high bandwidth-delay product paths. We focus on changes to the additive increase element of the TCP AIMD algorithm.

1. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

2. Introduction

The current TCP congestion control algorithm is known to perform poorly on paths where the TCP congestion window becomes very large. [Kelly02, Flo03, FAST04]. Following congestion, the congestion window is halved and only increases at a rate of 1 packet per RTT. As a result flows can take an unacceptably long time to recover their window size after a congestion event.

A direct solution is to make the time between congestion events smaller. This can be achieved by, for example, adjusting the AIMD additive increase rate to be greater for flows with larger congestion window. Backward compatibility with legacy TCP can be ensured through the inclusion of a separate mode of operation that behaves as legacy TCP in the appropriate circumstances.

The logic that orchestrates switching between the legacy and more aggressive modes of operation can clearly be designed several ways. One approach is to make the AIMD increase parameter, which we denote here by alpha, a function of the flow congestion window. That is, alpha is increased as congestion window increases thereby resulting in an additive increase algorithm that directly scales with congestion window. This is precisely the approach adopted in the High-Speed TCP [Flo03] proposal. In addition to adjusting the AIMD increase parameter alpha as a function of congestion window, this proposal also increases the multiplicative decrease factor beta to further increase the aggressiveness of a flow. (Note. On multiplicative decrease, the congestion window cwnd is updated to beta x cwnd. We use this definition of the backoff factor beta throughout this document).

While such modifications might appear straightforward, it has been shown [Sho04, Yi05] that they often negatively impact the behaviour of networks of TCP flows. High-speed TCP[Flo03] and BIC-TCP [BIC04] can exhibit extremely slow convergence following network disturbances such as the start-up of new flows; Scalable-TCP [Kelly02] is a multiplicative-increase multiplicative-decrease strategy and as such it is known that it may fail to converge to fairness in drop-tail networks [Jain89].

Our objective in this document is to therefore to renew discussion on how the TCP congestion control algorithm might best be modified to improve performance when the congestion window is large. Large congestion windows are associated with high bandwidth-delay product (BDP) paths and with the ongoing increase in network speeds, high BDP paths are becoming increasingly prevalent. In this document we focus on changes to the additive increase element of the TCP AIMD algorithm (we leave discussion of modifications to the backoff factor to a

later date). In particular, we present proposed changes to the additive increase algorithm that we argue are promising enough (based on the outcome of experimental tests carried out by a number of groups [Hegde04, Yi05, Cot05]) to warrant further discussion within the wider networking community.

Scope

Our focus in this document is on the behaviour of long-lived flows and so we do not consider changes to slow-start. We also seek to make the smallest possible changes to the existing TCP congestion control algorithm, and so confine consideration to the AIMD packet-loss based paradigm. Use of jumbo packets is viewed as complementary to the changes proposed here. We confine consideration to drop-tail queues as this is the prevalent queueing discipline in the current Internet and leave discussion of active queueing to a later date.

3. Additive Increase for High Bandwidth-Delay Product Paths

The AIMD algorithm used in TCP has two key features that underpin its convergence behaviour. Firstly, flows with the same RTT increase their congestion windows at the same rate. Secondly, the backoff mechanism is multiplicative. Hence, following congestion, flows with a larger congestion window will reduce their congestion window by more, in absolute terms, than flows with a smaller congestion window. Thus larger flows yield more bandwidth than smaller flows. Since flows increase congestion window at the same rate, flows with smaller congestion window thereby gain a certain advantage over flows with larger congestion window, and it is this that enables flows with small congestion window to seize bandwidth from flows with large congestion window until balance is reached in the network.

It follows from this observation that modifying the AIMD backoff factor can have a very significant impact on network responsiveness, and this is discussed in more detail elsewhere [Sho04, Sho05]. In this document we do not consider changes to the backoff factor. Instead, we confine attention to modifications to the AIMD increase rate with the aim of improving performance in high bandwidth-delay product paths. Provided we retain appropriate symmetry between the increase rates of competing flows, modifying the increase rate affects the interval between congestion events but otherwise does not affect the responsiveness of TCP.

We therefore propose generalising the AIMD algorithm by allowing the increase parameter alpha to vary as a function of the elapsed time since the last congestion event. Specifically, if we let Delta denote the time in seconds that has elapsed since the last congestion event experienced by a flow, we adjust the AIMD increase parameter according to some function which we denote f_alpha(Delta). To provide backward compatibility with legacy TCP flows we consider adjusting the increase parameter as follows

```
if Delta <= Delta_L
   alpha  = 1
else
   alpha = f_alpha(Delta)
```

where Delta_L is the threshold for switching from standard/legacy operation to the new increase function. The choice of function f_alpha is governed by the rate at which bandwidth should be acquired.

We can immediately make the observation that, because the adjustment is based on time since the last backoff, a degree of symmetry is maintained between competing network flows and in particular flows

already in high speed mode are not awarded a long-term advantage over newer flows. Specifically, when packet drops are synchronised Delta is necessarily the same for all flows. Hence all flows share identical increase profiles and symmetry is maintained [Sho04]. When drops are not synchronised, Delta is the same *on average* for all flows provided flows share the same probability of backing off on congestion. Hence, symmetry is still maintained, albeit in an average sense.

We select the increase function f_alpha such that the duration of the congestion epochs remains reasonably small as the bandwidth-delay product on a path increases. Below, we discuss one choice of increase function that yields convergence times that seem reasonable. However, the precise responsiveness requirement in future networks is currently not well defined and so we leave this, and the associated specific choice of increase function, as a question for further debate.

4. Choice of Increase Function

We consider, as an illustrative example, use of the increase function

f_alpha(Delta) = 1 + 10(Delta-Delta_L)+0.5(Delta-Delta_L)^2 (1)

and Delta_L=1 second. This choice yields the congestion epoch duration for a single flow, as a function of congestion window size, shown in Table 1.

Congestion window (packets)	Congestion epoch duration (s)
100	1.1
1000	3.1
2000	4.3
5000	6.6
10000	9.2
20000	12.8
50000	19.4

Table 1 - Congestion epoch duration vs congestion window size for an RTT of 100ms

4.1. RTT unfairness

It follows from the introductory discussion that (when RTT scaling is not used) the level of unfairness between flows with different RTT's is similar to that with the current AIMD algorithm. This behaviour is confirmed in experimental and simulation tests [HTCP04, Yi05].

4.2. Friendliness

The mean AIMD increase parameter is shown in Table 2 for a range of bandwidth-delay products. This an indication of the number of standard TCP flows (neglecting statistical multiplexing of backoffs) whose aggregate would be equivalent to a flow using increase function (1). That is, an indication of friendliness and also of the packet drop overhead associated with the AIMD probing action.

Congestion window (packets)	Effective number of standard TCP flows		
	10ms RTT	100ms RTT	250ms RTT
10	1	1	1
100	1	2	5
1000	3	12	22
2000	4	19	32
5000	8	33	55
10000	12	49	82
20000	19	72	123
50000	32	122	208

Table 2 - Mean increase parameter (packets/RTT) vs congestion window size

4.3. Responsiveness

Responsiveness is qualitatively similar to that of the current AIMD congestion control algorithm, i.e. the convergence time of TCP flows using an AIMD backoff factor of 0.5 is approximately 4 congestion epochs, although the congestion epoch duration is significantly shorter on high bandwidth-delay product paths (see Table 1).

4.4. Efficiency

Link utilisation depends on queue provisioning in a similar manner to the current TCP congestion control algorithm. That is, for a single flow (or multiple synchronised flows) 100% link utilisation requires that the queue be sized as the bandwidth-delay product. Simulation and experimental tests indicate that statistical multiplexing between unsynchronised flows yields similar efficiency gains to standard TCP.

5. RTT Scaling

We note that the parameter alpha determines the AIMD increase rate in packets per RTT. Hence, flows with the same RTT have the same increase rate in packets per second, but flows with different RTTs have different increase rate in packets per second. It is this that primarily leads to unfairness between flows with different RTTs. Removing RTT unfairness is not one of our objectives here. However, we note that an AIMD flow generates roughly alpha packet drops per RTT as a result of its probing action. Hence, flows with short RTT are more aggressive than flows with long RTT in the sense that they generate more packet drops over intervals of time measured in seconds. We can reduce the aggressiveness of short RTT flows by scaling the increase parameter alpha with RTT. This need not compromise the responsiveness of TCP flows. As noted in [Sh04, Sh05, HTCP04], the convergence time of TCP flows using an AIMD backoff factor of 0.5 is approximately 4 congestion epochs. Scaling alpha by RTT leads to scaling of the congestion epoch duration to become effectively the same for both short and long RTT flows. The convergence time is therefore also scaled to be effectively the same for both short and long RTT flows.

Such RTT scaling can be readily implemented by modifying the increase rule to

```
if Delta <= Delta_L
   alpha  = 1
else
   alpha = K x f_alpha(Delta)
```

where K = RTT/RTT_ref. Note that RTT scaling is not applied in low-speed conditions in order to maintain backward compatibility with legacy TCP flows (ensuring adequate backward compatibility presented a major difficulty in previous studies on the use of RTT scaling). Note also that the scaling is proportional to RTT rather than RTT^2, as we do not seek to achieve throughput fairness here. RTT_ref is the reference RTT for which f_alpha is designed to ensure acceptable congestion epoch durations.

6. Security Considerations

Security implications are not discussed in this document.

7. Acknowledgements

This work was supported by Science Foundation Ireland grants 00/PI.1/C067 and 04/IN3/I460.

8. Informative References

[Jain89] D.M. Chiu, R. Jain, Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. Computer Networks and ISDN Systems, 1989.

[Flo03] S.Floyd, HighSpeed TCP for Large Congestion Windows . Sally Floyd. IETF RFC 3649, Experimental, Dec 2003.

[FAST04] C. Jin, D.X. Wei, S,H. Low, FAST TCP: motivation, architecture, algorithms, performance. Proc IEEE INFOCOM 2004.

[Kelly02] T. Kelly, On engineering a stable and scalable TCP variant, Cambridge University Engineering Department Technical Report CUED/F-INFENG/TR.435, June 2002.

[HTCP04] D.J.Leith, R.N.Shorten, H-TCP Protocol for High-Speed Long-Distance Networks. Proc. 2nd Workshop on Protocols for Fast Long Distance Networks. Argonne, USA, 2004. http://www.hamilton.ie/net/htcp3.pdf

[BIC04] L. Xu, K. Harfoush, I. Rhee, Binary Increase Congestion Control for Fast Long-Distance Networks. Proc. INFOCOM 2004.

[Sho04] R.N.Shorten, D.J.Leith,J.Foy, R.Kilduff, Analysis and design of congestion control in synchronised communication networks. Automatica, 2004. http://www.hamilton.ie/net/synchronised.pdf

[Sho05] R.N.Shorten, F. Wirth,F., D.J. Leith, A positive systems model of TCP-like congestion control: Asymptotic results. http://www.hamilton.ie/net/unsynchronised_final.pdf

[Yi05] Y.Li, D.J.Leith, R.N.Shorten, Experimental evaluation of TCP protocols of high-speed networks. http://www.hamilton.ie/net/eval/

[Cot05] R.L. Cottrell, S. Ansari, P. Khandpur, R. Gupta, R. Hughes-Jones, M. Chen, L. MacIntosh, F. Leers, Characterization and Evaluation of TCP and UDP-Based Transport On Real Networks. . Proc. 3rd Workshop on Protocols for Fast Long-distance Networks, Lyon, France, 2005.

[Hegde04] S. Hegde, D. Lapsley, B. Wydrowski, J. Lindheim, D.Wei, C. Jin, S. Low, H. Newman, FAST TCP in High Speed Networks: An Experimental Study. Proc. GridNets, San Jose, 2004.

Authors' Addresses

Doug Leith
Hamilton Institute
NUI Maynooth
Maynooth, Co. Kildare
Ireland

Email: doug.leith@nuim.ie

Robert Shorten
Hamilton Institute
NUI Maynooth
Maynooth, Co. KIldare
Ireland

Email: robert.shorten@nuim.ie

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.