

Active Queue Management
and Packet Scheduling (aqm)
Internet Draft
Intended status:
Expires: August 2014

Wolfram Lautenschlaeger
Alcatel-Lucent
Bell Labs

February 13, 2014

Global Synchronization Protection for Packet Queues
draft-lauten-aqm-gsp-00.txt

Abstract

Transmission capacity sharing TCP flows tend to synchronize among each other. This way rate variations of the individual flows, which are caused by the congestion control algorithms, do not even out. The effect is known as global synchronization. Large queuing buffer demand and large latency and jitter are the consequences. Global Synchronization Protection (GSP) is an extension of regular tail drop packet queuing schemes that prevents global synchronization. For large traffic aggregates the de-correlation between the individual flow variations reduces buffer demand and packet sojourn time by an order of magnitude and more. Even though quite simple, the solution has a theoretical rationale and is not heuristic, and it has been tested with a Linux implementation.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on August 13, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction.....	2
2. Conventions used in this document.....	3
3. Root cause of global synchronization.....	3
4. Protecting queues of global synchronization.....	4
4.1. Basic algorithm.....	4
4.2. Interval adaptation at large flow numbers.....	5
4.3. Interval adaptation at small RTT.....	6
4.4. Threshold adaptation.....	6
4.5. Sanity checks and special cases.....	6
5. Security Considerations.....	6
6. IANA Considerations.....	7
7. Conclusions.....	7
8. References.....	7
8.1. Normative References.....	7
8.2. Informative References.....	7
9. Acknowledgments.....	7

1. Introduction

The congestion window (CWND) of a particular TCP connection, in combination with the round trip time (RTT), limits the transmission rate of the flow, which enables adaptation of the sending rate to the actual network conditions, [1]. TCP uses a rather coarse congestion control feedback by halving the congestion window in response to packet loss. To fill a bottleneck link by 100% anyway, a packet buffer in front of the link is required. For a single TCP flow a buffer in the range of bottleneck capacity multiplied by the round trip time is required (bandwidth-delay product rule, BDP), [2]. For aggregated traffic of many flows the picture is not so clear.

Conservative estimations tend towards BDP of the whole aggregate, i.e. link capacity * RTT. At the other hand, rate reductions due to CWND halving are still only in the range of a particular flow rate. With the assumption of N sharing flows, this yields ideally a buffer size of only (link capacity/N)*RTT. Unfortunately this value cannot be reached in practice. It would require a uniform distribution of rate reductions by the different flows over time. In opposite, rate reductions of bottleneck sharing flows tend to synchronize among each other, which is called global synchronization. In worst case, with all flows synchronized, the buffer demand is back at BDP of the whole traffic, thus confirming the conservative estimation.

There are cases where global synchronization does not occur, in particular large number of flows ($N > 500$), large spread of RTT between the different flows, and high frequency of flow renewals. In these cases the buffer size can be reduced to BDP/\sqrt{N} , which lies between the conservative and overly optimistic estimations above, [3]. Nevertheless there are still doubts, whether the absence of global synchronization is a general reliable design assumption for high capacity links, [4].

Most Active Queue Management (AQM) algorithms like RED [5] are aiming at better control of the queue size, which implies control over global synchronization. Global Synchronization Protection (GSP) goes the other way round. It suppresses the root cause of global synchronization and de-correlates the CWND variations of the competing flows, but it does not impact the behavior of a particular flow. This way it moves the buffer size demand down from conservative BDP of the whole link into the direction towards the ideal BDP of a single flow.

2. Conventions used in this document

In this document, the term "packet drop" is used for congestion notification, silently assuming that congestion marking for ECN could be equally applied.

In this document, the term "queue size" is preferably applied in number of bytes, however, the algorithm could be also applied to the number of packets, or even to the queuing delay (milliseconds).

3. Root cause of global synchronization

Global synchronization occurs in cases where a number of greedy TCP flows with comparably uniform RTT cross a tail drop queue in front of a shared transmission link. Tail drop means, a newly arriving packet is placed at the end of the queue if buffer space permits. Otherwise

it is dropped. The queue is drained from front of the queue at the speed of the link as long as packets are available. Greedy TCP flows means, the applications try to send as fast as possible and the flows are not limited elsewhere in the network.

In congestion avoidance state, all senders gradually increase their sending rate, in total exceeding the transmission capacity so that the queue is filling up. At some point in time, a first packet is dropped due to lack of buffer space. Ideally, the TCP flow, where the dropped packet belongs to, reduces its sending rate, the queue relaxes, and subsequently arriving packets can be placed in the buffer. Senders are continuing to increase their sending rates until the next drop, and so on.

Unfortunately, the rate reduction due to the dropped packet takes at least one RTT to take effect at the queue entry. During that RTT interval all senders continue to gradually increase their sending rates, whereas the queue is still full. Further packets need to be dropped. It can be shown analytically that for N flows with NewReno and delayed ACK the number of drops is in the range of $N/2$. (Experiments confirm this and show an even higher number with CUBIC.) The outcome is that even though the rate reduction by one flow would suffice, not one but half of the flows are triggered within one RTT to reduce their sending rates - we have global synchronization.

4. Protecting queues of global synchronization

4.1. Basic algorithm

The basic algorithm is as follows: Set a threshold on queue size below the actual buffer size. If a new packet arrives and the queue size is above the threshold, then drop that packet. After that, ignore any further threshold violation for a timeout interval of 1 - 3 RTT. After expiry of the timeout proceed as above.

Algorithm:

```
interval = e.g. 2 * RTT
threshold = e.g. 1/2 * buffer size
next_possible_drop = now
at any packet enqueueing do:
    if queue_size > threshold && now > next_possible_drop:
        drop this packet
        next_possible_drop = now + interval
    else
        enqueue this packet
end
```

The first dropped packet is triggering the rate reduction. During the timeout the queue is growing further beyond the threshold until the rate reduction takes effect at queue entry. Afterwards the queue size should have dropped below the threshold, so that at expiry of the timeout the threshold is typically not violated anymore. No explicit action occurs at timeout expiry, which makes the parameter rather insensitive to the actual traffic characteristics. Even if the timeout interval is too short, the algorithm still reduces global synchronization.

4.2. Interval adaptation at large flow numbers

The basic algorithm works well for moderate numbers of flows N , i.e. in a range of $2 < N < 20$. More precisely, at flow numbers N smaller than the average CWND of one of the sharing flows. At larger numbers the total rate increase during the timeout interval is larger than the subsequent rate reduction by one of the flows. As consequence, after timeout expiry the threshold is still violated, the queue is growing further and further, and, eventually, reaches the buffer limit and enters tail drop operation. The performance is still better than plain tail drop and one could rely on the observation that at large flow numbers global synchronization disappears, anyway.

Alternatively the initial timeout interval can be reduced, depending on the actual traffic, in a way, where not just once, but twice, or even more times per RTT the timeout expires. The adaptation criterion is the proportion of time above and below threshold. In regular operation according to the basic algorithm, the queue is most of the time below the threshold. If, however, the queue is more frequently above than below threshold, the interval should be reduced until equilibrium is reached. In this condition the queue is oscillating

around the threshold, quite similar to other AQM schemes like e.g. PED.

Algorithm:

```
at any packet enqueueing do:
    theta = cumulative time {above - below} threshold
    k = max(1, gamma * theta)
    interval = initial_interval/k
end
```

Gamma is a scaling parameter that controls the sensitivity of adaptation.

4.3. Interval adaptation at small RTT

The RTT is not known exactly but there should be at least a rough idea on the range of RTT for setting up the timeout interval. If this estimation is much too large, a similar situation occurs like in the large flow numbers case. The total rate increase during the timeout interval (which turns out to be multiple RTTs) is larger than the subsequent rate reduction by one flow. The adaptation rule is the same as for large flow numbers.

4.4. Threshold adaptation

Tbc

4.5. Sanity checks and special cases

An additional rule can be introduced that prevents large packet bursts from immediately triggering the drop: Restart the timeout not only after a packet drop but also whenever a packet is arriving at an empty queue.

5. Security Considerations

Global synchronization is a particular problem of many elastic flows sharing a bottleneck. GSP is there to prevent this. But it does not protect of unresponsive flows. If the congestion notification according to section 4.1. randomly hits an unresponsive flow then the expected rate reduction within the timeout interval might simply not happen, which postpones the notification by one timeout interval. In extreme cases, with a large amount of unresponsive traffic, GSP behaves like plain tail drop.

6. IANA Considerations

There are no actions for IANA.

7. Conclusions

tbc

8. References

8.1. Normative References

8.2. Informative References

- [1] Van Jacobson, Congestion avoidance and control, Proc. SIGCOMM '88, 1988
- [2] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm, Comput. Commun. Rev., 27.3, 1997, pp. 67-82.
- [3] G. Appenzeller, I. Keslassy, and N. McKeown, Sizing router buffers, Proc. ACM SIGCOMM '04, 2004.
- [4] G. Vu-Brugier, R. S. Stanojevic, D. J. Leith, R. N. Shorten, A critique of recently proposed buffer-sizing strategies, ACM SIGCOMM Computer Communication Review, 37.1, 2007
- [5] S. Floyd, Van Jacobsen, Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Trans. Networking, 1.4, 1993
- [6] K. Nichols, Van Jacobson, "Controlling Queue Delay", ACM Queue - Networks, 2012
- [7] R. Pan, et al., PIE: A lightweight control scheme to address the bufferbloat problem, IETF, draft-pan-tsvwg-pie-00, 2012

9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Wolfram Lautenschlaeger
Alcatel-Lucent
Bell Labs
Lorenzstrasse 10
70435 Stuttgart
Germany

Email: Wolfram.Lautenschlaeger@alcatel-lucent.com

INTERNET-DRAFT
Intended Status: Proposed Standard
Expires: July 20, 2014

R.Sallantin
CNES/TAS/TESA
C.Baudoin
F.Arnal
Thales Alenia Space
E.Dubois
CNES
E.Chaput
A.Beylot
IRIT
January 16, 2014

Safe increase of the TCP's Initial Window
Using Initial Spreading
draft-sallantin-iccr-initial-spreading-00

Abstract

This document proposes a new fast start-up mechanism for TCP that can be used to speed the beginning of an Internet connection and then improved the short-lived TCP connections performance.

Initial Spreading allows to safely increase the Initial Window size in any cases, and notably in congested networks.

Merging the increase in the IW with the spacing of the segments belonging to the Initial Window (IW), Initial Spreading is a very simple mechanism that improves short-lived TCP flows performance and do not deteriorate long-lived TCP flows performance.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	3
2	Terminology	4
3	Initial Spreading mechanism	4
4	Implementation considerations	4
4.1	Short Round Trip Time	5
4.2	Delayed Ack	5
4.3	TSO/GSO	5
4.4	RTT measure	6
5	Open discussions	6
5.1	Spacing Interval	6
5.2	Increasing the upper bound TCP's IW to more than 10 segments	7
5.3	Initial Spreading and LFN	7
6	Security Considerations	7
7	IANA Considerations	8
8	References	8
8.1	Normative References	8
8.2	Informative References	8
	Authors' Addresses	9

1 Introduction

The long Round Trip Time is probably the most detrimental constraint in Long Fat Networks (LFN), such as satellite networks, and notably for short-lived connections when the long delay significantly downgrades regular slow-start performance [FA11]. Several protocols and even new network architectures have been proposed to deal with this issue. The original idea of Initial Spreading [SB13] was to consider a long RTT as a resource to exploit, rather than as a constant to bypass. The long RTT can therefore be used as an opportunity to safely send a large amount of data during the first RTT after the connection has opened. Spacing the data along the whole RTT would in fact hopefully guarantee high independent probability that each segment is successfully received.

This approach resembles a combination of 2 TCP mechanisms: Pacing and Increase in the Initial Window. Both mechanisms have then been studied in depth to design Initial Spreading as an efficient fast start-up TCP mechanism, and notably avoid their respective flaws or weaknesses.

The original Pacing idea is to space the segments of a same window along an RTT to prevent generating bursts as far as possible. Hence, each segment arrives separately at the buffer and the impact on its queue is minimized. The bit rate can then reach its maximum. However, [AS00] has pointed out that this lack of bursts is responsible for poor performance. Pacing has a tendency to overload the network, and then cause a synchronization of the flows, that seriously damages both individual and global performance.

RFC 6928 [RFC6928] suggests to enlarge the IW size up to ten segments. Several articles and studies demonstrated that this would allow transmission of 90% of the connections in one RTT [DR10]. In most cases, and when the network is not congested in particular, this solution is probably the best one for dealing with short-lived TCP flows. However, in a congested environment, sending a large IW in one burst is likely to impact the buffers and then deteriorate the individual connection. Correlation between the segments of a same burst is responsible for major impairments when regarding the short-lived connections, and in particular for the connections that can be sent in one RTT (number of segments to be transmitted inferior to the upper bound value of the TCP's IW):

- o a decrease of the probability to successfully transmit the entire window.
- o an increase of the probability of successive segment losses.

- o a significant reduction of the number of potential Duplicated Acknowledgements that are necessary to trigger fast loss recovery mechanisms and avoid to wait for an Retransmission Time Out.

In favor of a conservative approach, [RFC3390] recommended the use of an IW equal to 3.

Both mechanisms therefore suffer from a burst-related phenomenon, but in opposite ways.

Initial Spreading has been designed to tackle previous burst issues. Simulations and experimentations show that Initial Spreading is not only efficient in case of LFNs but also for other networks with small RTT.

2 Terminology

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 [RFC2119].

3 Initial Spreading mechanism

Initial Spreading [SB13] mechanism uses the permitted upper bound value of the TCP's IW (e.g; RFC 6928 [RFC6928] suggests to use 10 for this value). Initial Spreading spaces out a number of segments inferior or equal to this value across the first RTT before letting the TCP algorithm continue conventionally:

- (1) The RTT is measured during the SYN-SYN/ACK exchange.
- (2) The first RTT is split into n spaces with n the permitted upper bound value of the TCP's IW. Depending on the number of segments to be sent, until n segments are sent every RTT/n.
- (3) After the transmission of the IW, the regular TCP algorithm is used.

Thus, bursts do not downgrade the transmission of short-lived connections, but continue to prevent an overload of the network in the case of long-lived connections.

4 Implementation considerations

In this section, we discuss a number of aspects surrounding the

Initial Spreading implementations.

4.1 Short Round Trip Time

Whatever the different timer implementations, 2 segments can not be spaced of less than 1 Kernel timer (e.g: jiffy for Linux kernel).

In case where the time resulting of the division of 1 RTT by the upper bound value of the TCP's IW is inferior to 1 Kernel timer, Initial Spreading is not activated and TCP uses a regular slow start with a large IW.

4.2 Delayed Ack

The use of Delayed Ack (Del Ack) does not downgrade Initial Spreading efficiency.

Regarding long-lived connections and notably TCP's steady state, the effects of Del Ack are lessened by new TCP's flavors (such as TCP Cubic or Compound TCP [HR08][TS06]) which tend to adapt their congestion algorithm to take into account whether the receiver uses the Del Ack option or not. In doing so, they can prevent the connection from being too slow, and still continue to reduce acknowledgments traffic. In the event of short-lived connections, the use of Del Ack does not modify the transmission of the IW. There is then no change in the burst propagation.

4.3 TSO/GSO

TSO/GSO is used to reduce the CPU overhead of TCP/IP on fast networks. Instead of doing the segmentation in the kernel, large packets are sent to the Network Interface Card (NIC). The segmentation is then achieved by the NIC or just before the entry into the driver's xmit routine.

In its current design, Initial Spreading is not working when TSO or GSO are activated, but using Initial Spreading with an inactive TSO/GSO still enables better performance.

Two options can be foreseen for the joint use of Initial Spreading and TSO/GSO:

- (1) disable TSO/GSO for the first RTT, with no impact on performance since the throughput is limited by the IW.
- (2) implement Initial Spreading using the TCP Offload Engine (TOE) [RFC5522].

4.4 RTT measure

Initial spreading uses the SYN-SYN/ACK exchange to calculate the space between two segments. This measurement may not be perfectly accurate in congested networks when the RTT varies. Two different scenarios can then occur:

- o The measured RTT is superior to the RTT of the first segment of the IW, and an ACK arrives before that all the segments of the IW have been sent. Then, Initial Spreading **MUST** be stopped and only the segments transmission that is triggered by the received ACK is done.
- o The measured RTT is inferior to the RTT of the first segment of the IW. Consequences are negligible.

5 Open discussions

In this section, we introduce possible improvements for Initial Spreading and new perspectives.

5.1 Spacing Interval

It has been observed that most of the savings enabled by the Initial Spreading in congested environments comes from the independence of the segments sent during the first RTT. Indeed, experimentations have shown that preventing the bursts, Initial Spreading enables each segment of the IW to have an independent loss probability.

Currently, Initial Spreading waits RTT/n seconds before transmitting two segments of the IW, with n the permitted upper bound value of the TCP's IW.

This simple mechanism offers very good results but has two minor drawbacks:

- (1) An inaccuracy of the space measure (cf section 4.4).
- (2) In uncongested networks, Initial Spreading adds an extra delay that is equal to $(IW-1) * RTT/n$, with IW the number of segments ($\leq n$) that can be sent during the first RTT.

A solution could be to set the space not as a ratio of the measured RTT but as a minimal space that preserves the independence between the sent segments. Preliminary results show that smaller spacing interval may allow to maintain the independence of the segment loss probability. This may provide the same performances in congested

networks and improve the average delay in uncongested networks.

5.2 Increasing the upper bound TCP's IW to more than 10 segments

[DR10] have shown that an IW of 10 segments enables to send more than 90% of the web objects in one RTT. So the authors recommend to use Initial Spreading as a complement to [RFC6928].

If the average size of the web objects continues to evolve, Initial Spreading can be used to raise the IW size. Simulations and experiments showed even better results with an IW equal to 12.

Thus, Initial Spreading paves the way for larger IW. Further studies are needed to assess the impact on the networks, notably in terms of individual performance, fairness, friendliness and global performance.

5.3 Initial Spreading and LFN

The space community designed middleboxes to mitigate poor TCP performance for network with large RTT [FA11]. Proxy Enhancement Performance (PEP) are generally used in LFN and in particular in satellite communication systems [RFC3135] and offer very good TCP performance.

Nevertheless, some recent studies have emphasized major impairments occasioned by the use of satellite-specific transport solutions, and notably TCP-PEPs, in a global context. The break of the end-to-end TCP semantic, which is required to isolate the satellite segment, is notably responsible for an increased complexity in case of mobility scenarios or security context. This strongly mitigates PEPs benefits and reopens the debate on their relevance[DC10].

Many researchers have outlined that new TCP releases perform well for long-lived TCP connections, even in satellite environment [SC12], but continue to suffer from very poor performance in case of short-lived TCP connections.

Initial Spreading enables to reduce the RTT consequences for short-lived TCP connections and could be an end-to-end alternative to PEP.

6 Security Considerations

The security considerations found in [RFC5681] apply to this document. No additional security problems have been identified with Initial Spreading at this time.

7 IANA Considerations

This document contains no IANA considerations.

8 References

8.1 Normative References

- [RFC3390] A. Allman and S. Floyd, "Increasing tcp's initial window," RFC 3390, IETF, Proposed Standard, 2002.
- [RFC5532] T. Talpey, C. Juszczak, "Network File System (NFS) Remote Direct Memory Access (RDMA) Problem Statement," RFC 5532, IETF, Informational, May 2009.
- [RFC6928] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, "Increasing tcp's initial window," RFC 6928, IETF, Experimental, Jan. 2013.
- [AH98] A. Allman, C. Hayes, and S. Ostermann, "An evaluation of TCP with Larger Initial Windows," ACM Computer Communication Review, 1998.
- [AS00] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in INFOCOM, vol. 3, mar 2000, pp. 1157-1165.
- [DR10] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin, "An Argument for Increasing TCP's Initial Congestion Window," SIGCOMM Comput. Commun. Rev., vol. 40, no. 3, pp. 26-33, Jun. 2010.
- [SB13] R. Sallantin, C. Baudoin, E. Chaput, E. Dubois, F. Arnal, and A. Beylot, "Initial spreading: a fast start-up tcp mechanism," proceedings of LCN, 2013.

8.2 Informative References

- [RFC3135] J. Border, M. Kojo, J. Griner, G. Montenegro, Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," RFC 3135, IETF, Informational, June 2001.
- [DF10] E. Dubois, J. Fasson, C. Donny, and E. Chaput, "Enhancing tcp based communications in mobile satellite scenarios: Tcp peps issues and solutions," in Proc. 5th Advanced

satellite multimedia systems conference (asma) and the 11th signal processing for space communications workshop (spsc), pages 476-483, 2010.

- [FA11] A. Fairhurst, G. Arjuna, H. Cruickshank, and C. Baudoin, "Transport challenges facing a next generation hybrid satellite internet," in International Journal of Satellite Communications and networking, 2011.
- [HR08] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," SIGOPS Oper. Syst. Rev., vol. 42, no. 5, pp. 64-74, Jul. 2008.
- [LC09] R. Lacamera, D. Caini, C. Firrincieli, "Comparative performance evaluation of tcp variants on satellite environments," in ICC'09 Proceedings of the 2009 IEEE international conference on Communications, pages Pages 5161-5165, 2009.
- [SC12] R. Sallantin, E. Chaput, E. P. Dubois, C. Baudoin, F. Arnal, and A.-L. Beylot, "On the sustainability of PEPs for satellite Internet access," in ICSSC. AIAA, 2012.
- [TS06] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "Compound TCP: A Scalable and TCP-friendly Congestion Control for High-speed Networks," in 4th International workshop on Protocols for Fast Long-Distance Networks (PFLDNet), 2006.

Authors' Addresses

Comments are solicited and should be addressed to the working group's mailing list at iccr@irtf.org and/or the authors:

Renaud Sallantin
CNES/TAS/TESA
IRIT/ENSEEIH 2, rue Charles Camichel BP 7122
31071 Toulouse Cedex 7
France
Phone: +33 6 48 07 86 44
Email: renaud.sallantin@gmail.com

Cedric Baudoin
Thales Alenia Space (TAS)
26 Avenue Jean Francois Champollion,
31100 Toulouse

France
Email: cedric.baudoin@thalesaleniaspace.com

Fabrice Arnal
Thales Alenia Space
Email: fabrice.arnal@thalesaleniaspace.com

Emmanuel Dubois
Centre National des Etudes Spatiales (CNES)
18 Avenue Edouard Belin
31400 Toulouse
France
Email: emmanuel.Dubois@cnes.Fr

Emmanuel Chaput
IRIT
IRIT / ENSEEIHT 2, rue Charles Camichel BP 7122
31071 Toulouse Cedex 7
France
Email: emmanuel.chaput@enseeiht.fr

Andre-Luc Beylot
IRIT
Email: andre-Luc.Beylot@enseeiht.fr

Active Queue Management and Packet Scheduling (aqm)
Internet-Draft
Intended status: Informational
Expires: August 18, 2014

G. White
CableLabs
R. Pan
Cisco Systems
February 14, 2014

A PIE-Based AQM for DOCSIS Cable Modems
draft-white-aqm-docsis-pie-00

Abstract

DOCSIS cable modems provide broadband Internet access to over one hundred million users worldwide. They are commonly positioned at the head of the bottleneck link for traffic in the upstream direction (from the customer), and as a result, the impact of buffering and bufferbloat in the cable modem can have a significant effect on user experience. The CableLabs DOCSIS 3.1 specification includes requirements for cable modems to support an Active Queue Management (AQM) algorithm that is intended to alleviate the impact that buffering has on latency sensitive traffic, while preserving bulk throughput performance. In addition, the CableLabs DOCSIS 3.0 specifications are being amended to contain similar requirements.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 18, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Overview of DOCSIS AQM Requirements	2
2. The DOCSIS MAC Layer and Service Flows	3
3. DOCSIS-PIE vs. PIE	4
3.1. Latency Target	4
3.2. Departure rate estimation	5
3.3. Packet drop de-randomization	6
3.4. Enhanced burst protection	6
3.5. Expanded auto-tuning range	6
3.6. Trigger for exponential decay	6
3.7. 16ms update interval	7
4. Implementation Guidance	7
5. References	7
Appendix A. DOCSIS-PIE Algorithm definition	8
A.1. DOCSIS-PIE AQM Constants and Variables	8
A.1.1. Configuration parameters	8
A.1.2. Constant values	8
A.1.3. Variables	9
A.1.4. Public/system functions:	9
A.2. DOCSIS-PIE AQM Control Path	10
A.3. DOCSIS-PIE AQM Data Path	12
Authors' Addresses	13

1. Overview of DOCSIS AQM Requirements

CableLabs' DOCSIS 3.1 specification [DOCSIS_3.1] mandates that cable modems implement a specific variant of the Proportional Integral controller Enhanced (PIE) [I-D.pan-aqm-pie] active queue management algorithm. This specific variant is provided for reference in Appendix A. CableLabs' DOCSIS 3.0 specification [DOCSIS_3.0] is being amended to recommend that cable modems implement the same algorithm. Both specifications allow that cable modems can optionally implement additional algorithms, that can then be selected for use by the operator via the modem's configuration file.

These requirements on the cable modem apply to upstream transmissions.

Both specifications also include requirements (mandatory in DOCSIS 3.1 and recommended in DOCSIS 3.0) that the Cable Modem Termination System (CMTS) implement active queue management for downstream traffic, however no specific algorithm is defined for downstream use.

2. The DOCSIS MAC Layer and Service Flows

The DOCSIS Media Access Control (sub-)layer provides tools for configuring differentiated Quality of Service for different applications by the use of Packet Classifiers and Service Flows.

Each cable modem can be configured with multiple Packet Classifiers and Service Flows. The maximum number of such entities that a cable modem supports is an implementation decision for the manufacturer, but modems typically support 16 or 32 Service Flows and at least that many Packet Classifiers.

Each Service Flow has an associated Quality of Service (QoS) parameter set that defines the treatment of the packets that traverse the Service Flow. These parameters include (for example) Minimum Reserved Traffic Rate, Maximum Sustained Traffic Rate, Peak Traffic Rate, Maximum Traffic Burst, Traffic Priority. Each upstream Service Flow corresponds to a queue in the cable modem, and each downstream Service Flow corresponds to a queue in the CMTS. The DOCSIS AQM requirements mandate that the CM and CMTS implement the AQM algorithm (and allow it to be disabled if need be) on each Service Flow queue independently.

Packet Classifiers can match packets based upon several fields in the packet/frame headers including the Ethernet header, IP header, and TCP/UDP header. Matched packets are then queued in the associated Service Flow queue.

It is typical that upstream and downstream Service Flows used for broadband Internet access are configured with a Maximum Sustained Traffic Rate. This QoS parameter rate-shapes the traffic onto the DOCSIS link, and is the main parameter that defines the service offering. Additionally, it is common that upstream and downstream Service Flows are configured with a Maximum Traffic Burst and a Peak Traffic Rate. These parameters allow the service to burst at a higher (sometimes significantly higher) rate than is defined in the Maximum Sustained Traffic Rate for the amount of bytes configured in Maximum Traffic Burst, as long as the long-term average data rate remains at or below the Maximum Sustained Traffic Rate.

Mathematically, what is enforced is that the traffic placed on the DOCSIS link in the time interval $(t1, t2)$ complies with the following rate shaping equations:

$$\text{TxBytes}(t1,t2) \leq (t2-t1)*R/8 + B$$

$$\text{TxBytes}(t1,t2) \leq (t2-t1)*P/8 + 1522$$

for all values $t2 > t1$, where:

R = Maximum Sustained Traffic Rate (bps)

P = Peak Traffic Rate (bps)

B = Maximum Traffic Burst (bytes)

The result of this configuration is that the link rate available to the Service Flow varies based on the pattern of load. If the load that the Service Flow places on the link is less than the Maximum Sustained Traffic Rate, the Service Flow "earns" credit that it can then use (should the load increase) to burst at the Peak Traffic Rate. This dynamic is important since these rate changes (particularly the decrease in data rate once the traffic burst credit is exhausted) can induce a step increase in buffering latency.

3. DOCSIS-PIE vs. PIE

There are a number of differences between the version of the PIE algorithm that is mandated for cable modems in the DOCSIS specifications and the version described in [I-D.pan-aqm-pie].

- o 10 ms default latency target, configurable per service flow
- o departure rate estimation
- o packet drop de-randomization
- o enhanced burst-protection
- o expanded auto-tuning range
- o trigger for exponential decay
- o 16ms update interval

3.1. Latency Target

The latency target (aka delay reference) is a key parameter that affects, among other things, the tradeoff in performance between latency-sensitive applications and bulk TCP applications. Via simulation studies, a value of 10ms was identified as providing a good balance of performance. However, it is recognized that there

may be service offerings for which this value doesn't provide the best performance balance. As a result, this is provided as a configuration parameter that the operator can set independently on each upstream service flow. If not explicitly set by the operator, the modem will use 10 ms as the default value.

3.2. Departure rate estimation

The PIE algorithm utilizes a departure rate estimator to track fluctuations in the egress rate for the queue and to generate a smoothed estimate of this rate for use in the drop probability calculation. This estimator may be well suited to many link technologies, but is not ideal for DOCSIS upstream links for a number of reasons.

First, the bursty nature of the upstream transmissions, in which the queue drains at line rate (up to ~100 Mbps for DOCSIS 3.0 and ~1 Gbps for DOCSIS 3.1) and then is blocked until the next transmit opportunity, results in the potential for inaccuracy in measurement, given that the PIE departure rate estimator starts each measurement during a transmission burst and ends each measurement during a (possibly different) transmission burst. For example, in the case where the start and end of measurement occur within a single burst, the PIE estimator will calculate the egress rate to be equal to the line rate, rather than the average rate available to the modem.

Second, the latency introduced by the DOCSIS request-grant mechanism can result in some further inaccuracy. In typical conditions, the request-grant mechanism can add between ~4 ms and ~8 ms of latency to the forwarding of upstream traffic. Within that range, the amount of additional latency that affects any individual data burst is effectively random, being influenced by the arrival time of the burst relative to the next request transmit opportunity, among other factors.

Third, in the significant majority of cases, the departure rate, while variable, is controlled by the modem itself via the pair of token bucket rate shaping equations described in Section 2. Together, these two equations enforce a maximum sustained traffic rate, a peak traffic rate, and a maximum traffic burst size for the modem's requested bandwidth. The implication of this is that the modem, in the significant majority of cases, will know precisely what the departure rate will be, and can predict exactly when transitions between peak rate and maximum sustained traffic rate will occur. This, as compared to the PIE estimator, which would be simply reacting to (and smoothing its estimate of) those rate transitions after the fact.

Finally, since the modem is already implementing the dual token bucket traffic shaper, it contains enough internal state to calculate predicted queuing delay with a minimum of computations. Furthermore, these computations only need to be run every drop probability update interval, as opposed to the PIE estimator, which runs a similar number of computations on each packet dequeue event.

For these reasons, the DOCSIS-PIE algorithm utilizes the configuration and state of the dual token bucket traffic shaper to translate queue depth into predicted queuing delay, rather than implementing the departure rate estimator defined in PIE.

3.3. Packet drop de-randomization

Similar to PIE, the DOCSIS-PIE algorithm utilizes a randomized drop mechanism driven by a calculated drop probability. With an i.i.d. Bernoulli random number generator however, the localized probability of drop (over a small number of sequential packets) can vary radically from the desired drop probability. In order to avoid these extreme excursions from the desired drop probability (p), DOCSIS-PIE enforces bounds as follows:

minimum number of passed packets between drops = $\text{ceil}((0.85/p)-1)$

maximum number of passed packets between drops = $\text{floor}((8.5/p)+1)$

3.4. Enhanced burst protection

The PIE algorithm contains a burst-protection feature which allows relatively short-lived (i.e. up to 150 ms) bursts of traffic to be enqueued (and thus create queuing latency) without the AQM triggering a packet loss. In DOCSIS-PIE this is extended such that bursts that never occupy more than 1/3 of the buffer are protected even if they last longer than 150 ms.

3.5. Expanded auto-tuning range

The PIE algorithm scales the PI coefficients based on the current drop probability. The DOCSIS-PIE algorithm extends this scaling to drop probabilities below $1e-4$.

3.6. Trigger for exponential decay

The PIE algorithm includes a mechanism by which the drop probability is allowed to decay exponentially (rather than linearly) when it is detected that the buffer is empty. In the DOCSIS case, recently arrived packets may reside in buffer due to the request-grant latency even if the link is effectively idle. As a result, the buffer may

not be identically empty in the situations for which the exponential decay is intended. To compensate for this, we trigger exponential decay when the buffer occupancy is less than $5\text{ms} * \text{Peak Traffic Rate}$.

3.7. 16ms update interval

PIE utilizes a 15ms update interval for the drop probability. In the case of DOCSIS-PIE, we utilize 16ms as it aligns with an integer number of nominal MAP scheduling intervals (which are typically 2ms).

4. Implementation Guidance

The AQM space is an evolving one, and it is expected that continued research in this field may in the future result in improved algorithms.

As part of defining the DOCSIS-PIE algorithm, we split the pseudocode definition into two components, a "data path" component and a "control path" component. The control path component contains the packet drop probability update functionality, whereas the data path component contains the per-packet operations, including the drop decision logic.

It is understood that some aspects of the cable modem implementation may be done in hardware, particularly functions that handle packet-processing.

While the DOCSIS specifications don't mandate the internal implementation details of the cable modem, modem implementers are strongly advised against implementing the control path functionality in hardware. The intent of this advice is to retain the possibility that future improvements in AQM algorithms can be accommodated via software updates to deployed devices.

5. References

[DOCSIS_3.0]

CableLabs, "DOCSIS 3.0 MAC and Upper Layer Protocols Specification", November 2013, <<http://www.cablelabs.com/wp-content/uploads/specdocs/CM-SP-MULPIv3.0-I23-131120.pdf>>.

[DOCSIS_3.1]

CableLabs, "DOCSIS 3.1 MAC and Upper Layer Protocols Specification", October 2013, <<http://www.cablelabs.com/wp-content/uploads/specdocs/CM-SP-MULPIv3.1-I01-131029.pdf>>.

[I-D.pan-aqm-pie]

Pan, R., Natarajan, P., Piglione, C., and M. Prabhu, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem", draft-pan-aqm-pie-01 (work in progress), February 2014.

Appendix A. DOCSIS-PIE Algorithm definition

PIE defines two functions organized here into two design blocks:

1. Control path block, a periodically running algorithm that calculates a drop probability based on the estimated queuing latency and queuing latency trend.
2. Data path block, a function that occurs on each packet enqueue: per-packet drop decision based on the drop probability.

It is desired to have the ability to update the Control path block based on operational experience with PIE deployments.

A.1. DOCSIS-PIE AQM Constants and Variables

A.1.1. Configuration parameters

- o LATENCY_TARGET. AQM Latency Target for this Service Flow
- o PEAK_RATE. Service Flow configured Peak Traffic Rate, expressed in Bytes/sec.
- o MSR. Service Flow configured Max. Sustained Traffic Rate, expressed in Bytes/sec.
- o BUFFER_SIZE. The size (in bytes) of the buffer for this Service Flow.

A.1.2. Constant values

- o A=0.25, B=2.5. Weights in the drop probability calculation
- o INTERVAL=16 ms. Update interval for drop probability.
- o DELAY_HIGH=200 ms.
- o BURST_RESET_TIMEOUT = 1 s.
- o MAX_BURST = 142 ms (150 ms-8 ms(update error))
- o MEAN_PKTSIZE = 1024 bytes

- o MIN_PKTSIZE = 64 bytes
- o PROB_LOW = 0.85
- o PROB_HIGH = 8.5
- o LATENCY_LOW = 5 ms

A.1.3. Variables

- o drop_prob_. The current packet drop probability.
- o accu_prob_. accumulated drop prob. since last drop
- o qdelay_old_. The previous queue delay estimate.
- o burst_allowance_. Countdown for burst protection, initialize to 0
- o burst_reset_. counter to reset burst
- o burst_state_. Burst protection state encoding 3 states:
 - NOBURST - no burst yet
 - FIRST_BURST - first burst detected, no protection yet
 - PROTECT_BURST - first burst detected, protecting burst if burst_allowance_ > 0
- o queue_. Holds the pending packets.

A.1.4. Public/system functions:

- o drop(packet). Drops/discards a packet
- o random(). Returns a uniform r.v. in the range 0 ~ 1
- o queue_.is_full(). Returns true if queue_ is full
- o queue_.byte_length(). Returns current queue_ length in bytes, including all MAC PDU bytes without DOCSIS MAC overhead
- o queue_.enqueue(packet). Adds packet to tail of queue_
- o msrtokens(). Returns current token credits (in bytes) from the Max Sust. Traffic Rate token bucket
- o packet.size(). Returns size of packet

A.2. DOCSIS-PIE AQM Control Path

The DOCSIS-PIE control path performs the following:

- o Calls `control_path_init()` at service flow creation
- o Calls `calculate_drop_prob()` at a regular INTERVAL (16ms)

```
=====
// Initialization function
control_path_init() {
    drop_prob_ = 0;
    qdelay_old_ = 0;
    burst_reset_ = 0;
    burst_state_ = NOBURST;
}

// Background update, occurs every INTERVAL
calculate_drop_prob() {

    if (queue_.byte_length() <= msrtokens()) {
        qdelay = queue_.byte_length() / PEAK_RATE;
    } else {
        qdelay = ((queue_.byte_length() - msrtokens()) / MSR \
                  + msrtokens() / PEAK_RATE);
    }

    if (burst_allowance_ > 0) {
        drop_prob_ = 0;
    } else {
        p = A * (qdelay - LATENCY_TARGET) + \
          B * (qdelay - qdelay_old_);
        // Since A=0.25 & B=2.5, can be implemented
        // with shift and add

        if (drop_prob_ < 0.000001) {
            p /= 2048;
        } else if (drop_prob_ < 0.00001) {
            p /= 512;
        } else if (drop_prob_ < 0.0001) {
            p /= 128;
        } else if (drop_prob_ < 0.001) {
            p /= 32;
        } else if (drop_prob_ < 0.01) {
            p /= 8;
        } else if (drop_prob_ < 0.1) {
            p /= 2;
        } else if (drop_prob_ < 1) {
```

```
        p /= 0.5;
    } else if (drop_prob_ < 10) {
        p /= 0.125;
    } else {
        p /= 0.03125;
    }

    if ((drop_prob_ >= 0.1) && (p > 0.02)) {
        p = 0.02;
    }
    drop_prob_ += p;

    /* for non-linear drop in prob */
    if (qdelay < LATENCY_LOW && qdelay_old_ < LATENCY_LOW) {
        drop_prob_ *= 0.98; // (1-1/64) is sufficient
    } else if (qdelay > DELAY_HIGH) {
        drop_prob_ += 0.02;
    }

    drop_prob_ = max(0, drop_prob_);
    drop_prob_ = min(drop_prob_, \
        PROB_LOW * MEAN_PKTSIZE/MIN_PKTSIZE);
}

if (burst_allowance_ < INTERVAL)
    burst_allowance_ = 0;
else
    burst_allowance_ = burst_allowance_ - INTERVAL;

// both old and new qdelay is well better than the
// target and drop_prob_ == 0, time to clear burst tolerance
if ((qdelay < 0.5 * LATENCY_TARGET)
    && (qdelay_old_ < 0.5 * LATENCY_TARGET)
    && (drop_prob_ == 0)
    && (burst_allowance_ == 0)){

    if (burst_state_ == PROTECT_BURST) {
        burst_state_ = FIRST_BURST;
        burst_reset_ = 0;

    } else if (burst_state_ == FIRST_BURST) {
        burst_reset_ += INTERVAL ;
        if (burst_reset_ > BURST_RESET_TIMEOUT) {
            burst_reset_ = 0;
            burst_state_ = NOBURST;
        }
    }
} else if (burst_state_ == FIRST_BURST) {
```

```

        burst_reset_ = 0;
    }

    qdelay_old_ = qdelay;
}

```

A.3. DOCSIS-PIE AQM Data Path

The DOCSIS-PIE data path performs the following:

- o Calls enqueue() in response to an incoming packet from the CMCI

```

=====
enqueue(packet) {
    if (queue_.is_full()) {
        drop(packet);
        accu_prob_ = 0;
    } else if (drop_early(packet, queue_.byte_length())) {
        drop(packet);
    } else {
        queue_.enqueue(packet);
    }
}

//////////
drop_early(packet, queue_length) {
    if (burst_allowance_ > 0) {
        return FALSE;
    }

    if (drop_prob_ == 0) {
        accu_prob_ = 0;
    }

    if (burst_state_ == NOBURST) { //first burst?
        if (queue_.byte_length() < BUFFER_SIZE/3) {
            return FALSE;
        } else {
            burst_state_ = FIRST_BURST; //burst detected
        }
    }

    //The CM can quantize packet.size to 64, 128, 256, 512, 768,
    // 1024, 1280, 1536, 2048 in the calculation below
    p1 = drop_prob_ * packet.size() / MEAN_PKTSIZE;
    p1 = min(p1, PROB_LOW);
}

```

```
    accu_prob_ += p1;

    // If latency is low, don't drop packets
    if ( (qdelay_old_ < 0.5 * LATENCY_TARGET && drop_prob_ < 0.2)
        || (queue_.byte_length() <= 2 * MEAN_PKTSIZE) ) {
        return FALSE;
    }

    drop = TRUE;
    if (accu_prob_ < PROB_LOW) { // avoid dropping too fast due
        drop = FALSE;           // to bad luck of coin tosses...
    } else if (accu_prob_ >= PROB_HIGH) { // ...and avoid dropping
        drop = TRUE;             // too slowly
    } else {                     //Random drop
        double u = random();      // 0 ~ 1
        if (u > p1) {
            drop = FALSE;
        }
    }

    if (drop == FALSE) return FALSE;

    // In case of packet drop:
    accu_prob_ = 0;

    // Not protecting burst yet? Start protecting burst.
    // This will set the burst_allowance_ value, and
    // calculate_drop_prob() will decrement it.
    // Could implement this as a 150ms timer instead.
    if (burst_state_ == FIRST_BURST) {
        burst_state_ = PROTECT_BURST;
        burst_allowance_ = MAX_BURST;
    }
    return TRUE;
}
```

Authors' Addresses

Greg White
CableLabs
858 Coal Creek Circle
Louisville, CO 80027-9750
USA

Email: g.white@cablelabs.com

Rong Pan
Cisco Systems
510 McCarthy Blvd
Milpitas, CA 95134
USA

Email: ropan@cisco.com