Multipath TCP
Internet-Draft
Intended status: Standards Track
Expires: December 29, 2016

R. Barik
University of Oslo
S. Ferlin
Simula Research Laboratory
M. Welzl
University of Oslo

June 27, 2016

A Linked Slow-Start Algorithm for MPTCP
draft-barik-mptcp-lisa-01

Abstract

This document describes the LISA (Linked Slow-Start Algorithm) for Multipath TCP (MPTCP). Currently during slow-start, subflows behave like independent TCP flows making MPTCP unfair to cross-traffic and causing more congestion at the bottleneck. This also yields more losses among the MPTCP subflows. LISA couples the initial windows (IW) of MPTCP subflows during the initial slow-start phase to remove this adverse behavior.

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 December 29, 2016.

Copyright Notice

Table of Contents

1. Introduction

The current MPTCP implementation provides multiple congestion control algorithms, which aim to provide fairness to TCP flows at the shared bottlenecks. However, in RFC 6356 [RFC6356], the subflows' slow-start phase remains unchanged to RFC 5681 [RFC5681], and all the subflows at this stage behave like independent TCP flows. Following the development of IW as per [RFC6928], each MPTCP subflow can start with IW = 10. With an increasing number of subflows, the subflows' collective behavior during the initial slow-start phase can temporarily be very aggressive towards a concurrent regular TCP flow at the shared bottleneck.

According to [UIT02], most of the TCP sessions in the Internet consist of short flows, e.g., HTTP requests, where TCP will likely never leave slow-start. Therefore, the slow-start behavior becomes of critical importance for the overall performance.

To mitigate the adverse effect during initial slow-start, we introduce LISA, the "Linked Slow-Start Algorithm". LISA shares the congestion window MPTCP subflows in slow start whenever a new subflow joins.

1.1. Definitions

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].

Acronyms used in this document:

IW -- Initial Window

RTT -- Round Trip Time

CWND -- Congestion Window

Inflight -- MPTCP subflow's inflight data

old_subflow.CWND -- Congestion Window of the subflow having largest sending rate

new_subflow.CWND -- New incoming subflow's Congestion Window

Ignore_ACKs -- a boolean variable indicating whether ACKs should be ignored

ACKs_To_Ignore -- the number of ACKs for which old_subflow.CWND stops increasing during slow-start

compound CWND -- sum of CWND of the subflows in slow-start

2. MPTCP Slow-Start Problem Description

Since it takes 1 RTT for the sender to receive any feedback on a given TCP connection, sending an additional segment after every ACK is rather aggressive. Therefore, in slow-start, all subflows independently doubling their CWND as in regular TCP results in MPTCP also doubling its compound CWND. The MPTCP aggregate only diverges from this behavior when the number of subflows changes. Coupling of CWND is therefore not necessary in slow-start except when a new subflow joins.

2.1. Example of current MPTCP slow-start problem

We illustrate the problematic MPTCP slow-start behavior with an example: Consider an MPTCP connection consisting of 2 subflows. The first subflow starts with IW = 10, and after 2 RTTs the CWND becomes 40 and a new subflow joins, again with IW = 10. Then, the compound CWND becomes 40+10 = 50. With an increasing number of subflows, the compound CWND in MPTCP becomes larger than that of a concurrent TCP flow.

For example, MPTCP with eight subflows (as recommended in [DCMPTCP11] for datacenters) will have a compound CWND of 110 (40+7*10). As a result, MPTCP would behave unfairly to a concurrent TCP flow sharing the bottleneck. This aggressive behavior of MPTCP also affects the performance of MPTCP. If multiple subflows share a bottleneck, each of them doubling their rate every RTT, will cause excessive losses at the bottleneck. This makes MPTCP enter the congestion avoidance phase earlier and thereby increases the completion time of the transfer.

This problem, and the improvement attained with LISA, are documented in detail in [lisa].

3. Linked Slow-Start Algorithm

3.1. Description of LISA

The idea behind LISA is that each new subflow takes a ’packet credit’ from an existing subflow in slow-start for its own IW. We design the mechanism such that a new subflow has 10 segments as the upper limit

[RFC6928] and 3 segments as the lower limit [RFC3390]. This is based on [RFC6928], [RFC3390] and the main reason behind it is to let these subflows compete reasonably with other flows. We also divide the CWND fairly in order to give all subflows an equal chance when competing with each other.

LISA first finds the subflow with the largest sending rate measured over the last RTT. Depending on the subflow's CWND, between 3 and 10 segments are taken from it as packet credit and used for the new subflow's IW. The packet credit is realized by reducing the CWND from the old subflow and halting its increase for ACKs_To_Ignore number of ACKs.

We clarify LISA with the example given in Section 2.1. After 2 RTTs, the old_subflow.CWND = 40 and a new_subflow joins the connection. Since old_subflow.CWND >= 20 (refer to Section 3.2), 10 packets can be taken by the new_subflow.CWND, resulting in old_subflow.CWND = 30 and new_subflow.CWND = 10. Hence, MPTCP's compound CWND, whose current size is 40, should ideally become 60+20 = 80 after 1 RTT (assuming a receiver without delayed ACKs). However, if 40 segments from old_subflow.CWND are already in flight, the compound CWND becomes in fact 70+20 = 90. Here, LISA keeps old_subflow.CWND from increasing for the next 10 ACKs. In comparison, MPTCP without LISA would have a compound CWND of 80+20=100 after 1 RTT.

3.2. Algorithm

Below, we describe the LISA algorithm. LISA is invoked before a new subflow sends its IW.

1. Before computing the new_subflow.CWND, Ignore_ACKs = False and ACKs_To_Ignore = 0.

2. Then, ignoring the new_subflow, the subflow in slow-start with the largest sending rate (old_subflow.CWND, measured over the last RTT) is selected.

3. If there is no such subflow, the IW of the new_subflow.CWND = 10 Otherwise, the following steps are executed:

```
if old_subflow.CWND >= 20 // take IW(10) packets

   old_subflow.CWND -= 10

   new_subflow.CWND = 10
```

```
            Ignore_ACKs = True
         else if old_subflow.CWND >= 6 // take half the packets
            new_subflow.CWND -= old_subflow.CWND / 2
            old_subflow.CWND -= new_subflow.CWND
            Ignore_ACKs = True
         else
            new_subflow.CWND = 3 // can't take from old_subflow
   4.  if Ignore_ACKs and Inflight > old_subflow.CWND
         // do not increase CWND when ACKs arrive
         ACKs_To_Ignore = Inflight - old_subflow.CWND
```

4. Implementation Status

LISA is implemented as a patch to the Linux kernel 3.14.33+ and within MPTCP's v0.89.5. It is meant for research and provided by the Unviersity of Oslo and Simula Research Laboratory, and available for download from http://heim.ifi.uio.no/runabk/lisa This code was used to produce the test results that are reported in [lisa].

5. Acknowledgements

This work was part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The authors also would like to thank David Hayes (UiO) for his comments. The views expressed are solely those of the authors.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

8. Change History

Changes made to this document:

00->01 : Some minor text improvements and updated a reference.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3390] Allman, M., Floyd, S., and C. Partridge, "Increasing TCP's Initial Window", RFC 3390, DOI 10.17487/RFC3390, October 2002, <http://www.rfc-editor.org/info/rfc3390>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

[RFC6356] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, DOI 10.17487/RFC6356, October 2011, <http://www.rfc-editor.org/info/rfc6356>.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <http://www.rfc-editor.org/info/rfc6928>.

9.2. Informative References

[DCMPTCP11] Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., and M. Handley, "Improving datacenter performance and robustness with multipath TCP", ACM SIGCOMM p266-277, August 2011.

[UIT02] Brownlee, N. and K. Claffy, "Understanding internet traffic streams: Dragonflies and tortoises", IEEE Communications Magazine p110-117, 2002.

[lisa] Barik, R., Welzl, M., Ferlin, S., and O. Alay, "LISA: A Linked Slow-Start Algorithm for MPTCP", IEEE ICC 2016,

Kuala Lumpur, Malaysia , 2016.

Authors' Addresses

Runa Barik
University of Oslo
PO Box 1080 Blindern
Oslo N-0316
Norway

Email: runabk@ifi.uio.no

Simone Ferlin
Simula Research Laboratory
P.O.Box 134
Lysaker, 1325
Norway

Email: ferlin@simula.no

Michael Welzl
University of Oslo
PO Box 1080 Blindern
Oslo, N-0316
Norway

Phone: +47 2285 2420
Email: michawe@ifi.uio.no

Transport Area Working Group
Internet-Draft
Intended status: Experimental
Expires: April 27, 2017

J. Holland
Akamai Technologies, Inc.

October 24, 2016

Circuit Breaker Assisted Congestion Control (CBACC): Protocol Specification
draft-jholland-cb-assisted-cc-00

Abstract

This document specifies Circuit Breaker Assisted Congestion Control (CBACC), which provides bandwidth information from senders to intermediate network nodes to enable good decisions for fast-trip Network Transport Circuit Breaker activity ([I-D.ietf-tsvwg-circuit-breaker]) when necessary for network health. CBACC is specifically designed to support protocols using IP multicast, particularly as a supplement to receiver-driven congestion control protocols such as WEBRC [RFC3738], to help affected networks rapidly detect and mitigate the impact of scenarios in which a network is oversubscribed to flows which are not responsive to congestion.

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 April 27, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document specifies Circuit Breaker Assisted Congestion Control (CBACC).

CBACC is a congestion control building block designed for use with IP traffic that has a known maximum bandwidth, which does not reduce its sending rate in response to congestion. CBACC is specifically designed to supplement protocols using WEBRC [RFC3738] or other

receiver-driven multicast congestion control systems that rely on well-behaved receivers to achieve congestion control in a very highly scalable system (up to millions of receivers) without a feedback path that reduces sending rates by senders.

CBACC addresses a vulnerability to "overjoining", a condition in which receivers (particularly malicious receivers) subscribe to traffic which, from the sending side, is non-responsive to congestion. Overjoining attacks and the challenges they present are discussed in more detail in Appendix A.

A careful reading of the congestion control requirements of UDP Best Practices [I-D.ietf-tsvwg-rfc5405bis] suggests that a network that forwards multicast traffic is required to operate a circuit breaker to maintain network health under a persistent overjoining condition, at a cost of cutting off some or all multicast traffic across the network during high congestion.

CBACC provides a mechanism for networks to mitigate the impact of network overjoining by sender advertising of bandwidth information sufficient to implement a fast-trip circuit breaker [I-D.ietf-tsvwg-circuit-breaker] within a single network node which can specifically block the most problematic flows, and which can ensure the remaining flows fit within desired network parameters.

In conjunction with receiver counts (e.g. via [RFC6807]) such nodes can also provide much improved network fairness for circuit breaking decisions during an overjoining condition.

In addition to streams using WEBRC, CBACC may also be suitable for use with other traffic, both unicast and multicast, that does not respond to congestion by reducing sending rates, including certain profiles of RTP [RFC3550] over either unicast or multicast, as well as several tunneling protocols (e.g. AMT [RFC7450] and GRE [RFC2784]) when they are known to carry traffic that would be suitable for CBACC. A complete specification for use of CBACC with unicast protocols and with tunneling protocols is out of scope for this document, though the security issues section does mention a few special considerations for potential unicast usage.

CBACC-compliant senders transmit Bandwidth Advertisements through the same transport path as the data traffic, so that circuit breakers can make informed decisions about how flows should be prioritized for circuit breaking. Additionally, CBACC-compliant circuit breakers transmit information to receivers about flows which have been or might soon be circuit-broken, to encourage CBACC-aware applications to use alternate methods to retrieve equivalent (though probably lower-quality and possibly less efficient) data when possible.

This document describes a building block as defined in [RFC3048]. This document describes a congestion control building block that conforms to [RFC2357]. This document follows the general guidelines provided in [RFC3269], in addition to the requirements on RFCs from [RFC5226] and [RFC3552].

2. Terminology

Term	Definition
circuit breaker	See [I-D.ietf-tsvwg-circuit-breaker]
controlled environment	See [I-D.ietf-tsvwg-rfc5405bis] Section 3.6
general internet	See [I-D.ietf-tsvwg-rfc5405bis] Section 3.6
flow	traffic for a single (source,destination) IP pair, including destinations that are group addresses
upstream	along a network topology path in the direction of a flow's sender
downstream	along a network topology path in the direction of a flow's receiver
ingress interface	the (single) upstream interface for a flow in a circuit breaker
egress interface	a downstream interface for a flow in a circuit breaker

Table 1

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 [RFC2119].

3. Rationale

CBACC is defined as an independent congestion control building block instead of as an extension to WEBRC because it would be an equally useful supplement for other kinds of experimental receiver-driven multicast congestion control schemes, such as [PLM] or other methods based on receiver-driven conformance to a measurement of available network bandwidth or congestion.

CBACC is also potentially valuable, even without other congestion control systems, in controlled environments where congestion control may not be required (e.g. for certain profiles of RTP [RFC3550]),

since CBACC can provide protection for such a network against congestion due to sender or network mis-configuration.

CBACC provides a new form of communication between senders and network transit nodes to facilitate fast-trip circuit breakers as described in section 5.1 of [I-D.ietf-tsvwg-circuit-breaker] which are not available via previously existing methods. When used in conjunction with compatible circuit breakers, CBACC can greatly improve the safety of a network that accepts and delivers interdomain massively scalable multicast traffic to potentially untrusted receivers.

4. Applicability

CBACC relies on the presence of CBACC-aware circuit breakers on a flow's transit path in order to provide congestion control in a network. In the absence of any CBACC-aware circuit breakers on a network path, CBACC constitutes a small extra overhead to a flow that provides no additional value.

CBACC provides a form of congestion control for massively scalable protocols using the IP multicast service. CBACC is best used in conjunction with another receiver-driven multicast congestion control, but it is also suitable for use even without another congestion control mechanism, or when presence of another congestion control mechanism is unproven, such as when accepting multicast joins from untrusted receivers.

5. Protocol Specification

5.1. Overview

CBACC senders send Bandwidth Advertisement packets to advertise the maximum sending bandwidth along the data path for a flow through a network.

CBACC bandwidth information is monitored by CBACC circuit breakers along the network path, which may block the forwarding of traffic for some flows in order to maintain network health. When a flow is blocked, a CBACC circuit breaker sets a bit in Bandwidth Advertisement packets before they're forwarded downstream that indicates to subscribed receivers of that flow that traffic has been blocked.

An effort is also made to notify downstream receivers when a flow is in danger of being circuit broken in the near future. This gives applications an opportunity to gracefully shift to a lower-bandwidth

version of the same content, when possible, providing an early warning system against potential congestion.

A Bandwidth Advertisement packet constitutes an "ingress meter" as described in section 3.1 of [I-D.ietf-tsvwg-circuit-breaker]. The configured bandwidth caps of egress interfaces likewise constitute "egress meters". However, the diagram in the referenced document is simplified by running the ingress and egress on the same network node. At the CBACC-aware circuit breaker, that node has both pieces of information as soon as a Bandwidth Advertisement is received, and can trip the circuit breaker if the aggregate CBACC bandwidth exceeds the bandwidth available on any egress interfaces.

5.2. Packet Header Fields

5.2.1. Introduction

The initial draft of this document proposes 2 different methods of encapsulating the Bandwidth Advertisement with a discussion of the known pros and cons of each. The intent is to solicit feedback from the community and then settle on one encapsulation (possibly a different, as yet unconsidered one).

5.2.2. Bandwidth Advertisement

5.2.2.1. As an IP header option

Bandwidth advertisement can appear as either an IPv4 header option (as in Section 3.1 of [RFC0791]) or as an IPv6 extension header option (as in section 4.2 of [RFC2460]). They have the same layout:

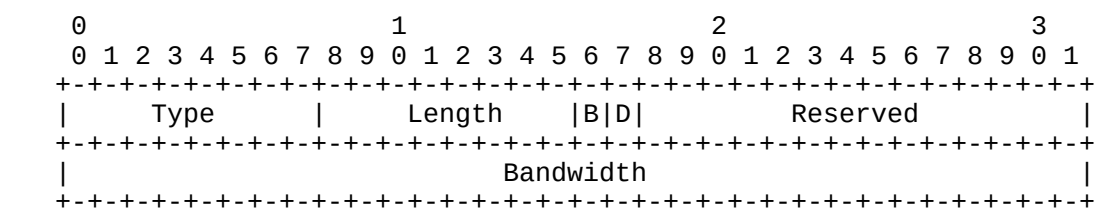


Figure 1

Bandwidth advertisements sent as IPv4 header options use option value [TBD], with the "copied" bit set and the option class "control", as specified in [RFC0791] section 3.1. Until and unless IANA assigns a value, this will be option number 158 as described in section 8 of [RFC4727] for experiments using IPv4 Option types. The length field is 8.

Bandwidth advertisements sent as IPv6 header options use option value [TBD], with the "action" bits set to "skip" and the "change" bit set to 1, as specified in [RFC2460] section 4.2. Until and unless IANA assigns a value, this will be option number 0x3e as described in section 8 of [RFC4727] for experiments using IPv6 Option Types. The length field is 6.

Using an IP header option has the benefit of exposing the bandwidth to all CBACC-compatible routers, in much the same way the IP Router Alert option would, but without being processed or causing undue load in non-CBACC routers.

The IP Header encapsulations DO work with IPSEC. As described in Appendix A of [RFC4302], the IP header fields are properly treated as mutable and zeroed for the IPSEC ICV calculations. CBACC circuit breakers MAY change bits in transit. The Bandwidth Advertisement header itself IS NOT protected by IPSEC security services, but protection of other parts of the packet remain unchanged.

5.2.2.2. As a UDP packet

Bandwidth advertisements can appear as a UDP packet with destination port [TBD]. Until and unless IANA assigns a value, this will be port number 1022, one of the experimental ports provided in Section 6 of [RFC4727]. The UDP packet contains a payload with the following packet format:

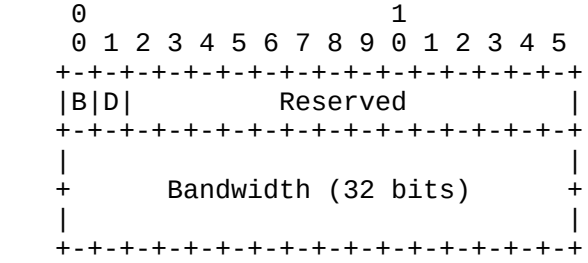


Figure 2

Using a dedicated UDP packet has the benefit of following the guidelines from section 2.2.5 of [RFC3269], that Reliable Multicast Building Blocks MUST initially define packet formats for use over UDP until such time as the protocol is sufficiently widely deployed and understood. However, since the Bandwidth Advertisement packets will be processed by intermediate network nodes while being transmitted along the same routing path as the corresponding data packets, it will be necessary to add an IP Router Alert option ([RFC2113] and [RFC2711]) to the appropriate packets to flag them for the circuit-

breaker's examination. It's important to note that [RFC6398] specifically recommends against using the Router Alert option in the end-to-end open Internet, which may impede experimentation and would very likely impede early adoption.

An additional disadvantage of UDP relative to the IP header option is that CBACC in UDP cannot be used for unicast tunnels, because the UDP port is the only discriminator for the Bandwidth Advertisement.

The UDP encapsulation DOES NOT work with IPSEC, because middle boxes can't unpack or modify the Bandwidth Advertisement packets. [TO BE REMOVED: Does this violate [RFC3269] section 2.2.5?]

5.2.2.3. Field definitions

5.2.2.3.1. Bandwidth

As in several other protocols sending bandwidth values such as OSPF-TE [RFC3630], the bandwidth is expressed in bytes per second (not bits), in IEEE floating point format. For quick reference, this format is as follows:

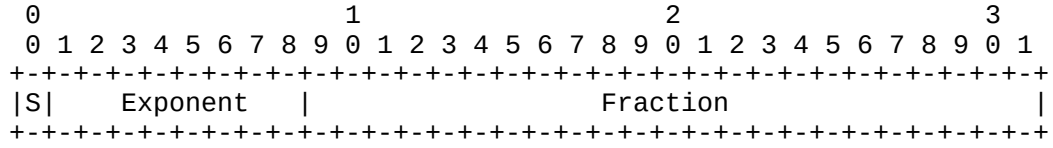


Figure 3

S is the sign, Exponent is the exponent base 2 in "excess 127" notation, and Fraction is the mantissa - 1, with an implied binary point in front of it. Thus, the above represents the value:

(-1)**(S) * 2**(Exponent-127) * (1 + Fraction)

For more details, refer to [IEEE.754.1985].

Figure 4

5.2.2.3.2. B (Blocked) bit

Indicates that the flow has been circuit-broken.

5.2.2.3.3. D (Danger) bit

Indicates that the flow is in danger of being circuit-broken.

5.2.2.3.4. Reserved bits

The sender MUST set all reserved bits to 0 when sending a CBACC control packet. Receivers MUST accept any value in the reserved bits.

5.3. States

5.3.1. Interface State

A CBACC circuit breaker holds the following state for each interface, for both the inbound and outbound directions on that interface:

aggregate bandwidth The sum of the bandwidths of all non-circuit-broken CBACC flows which transit this interface in this direction.

bandwidth limit The maximum aggregate CBACC bandwidth allowed, not including circuit-broken flows. This may depend on administrative configuration and congestion measurements for the network, whether from this node or other nodes. It's out of scope for this document to define such congestion measurements. Network operators should carefully consider that this bandwidth limit applies to flows that are unresponsive to congestion.

When reducing the bandwidth limit due to congestion, the circuit breaker MUST NOT reduce the limit by more than half its value in 10 seconds, and SHOULD use a smoothing function to reduce the limit gradually over time.

It is RECOMMENDED that no more than half the capacity for a link be allocated to CBACC flows if the link might be shared with TCP or other traffic that is responsive to congestion.

Depending on administrative configuration and the physical characteristics of the interface, the bandwidth limit may be either shared between upstream and downstream traffic, or it may be separate. Either a single shared value should be used, or two separate independent values should be used for the inbound and outbound directions for an interface.

CBACC bandwidth warning threshold A soft bandwidth threshold. When the aggregate CBACC bandwidth exceeds this threshold, flows that would have been circuit-broken with a bandwidth limit at this

threshold MUST have the Danger bit set in the Bandwidth Advertisement packets that are forwarded by this circuit breaker. This threshold SHOULD be configurable as a proportion of the bandwidth limit, and MUST remain at or below the bandwidth limit when the bandwidth limit changes. The recommended proportion value is .75, but specific networks may use a different value if deemed useful by the network operators.

5.3.2. Flow State

The following state is kept for flows that are joined from at least one downstream interface and for which at least one CBACC Bandwidth Advertisement packet has been received:

bandwidth The bandwidth from the most receintly received Bandwidth Advertisement.

ingress status One of the following values:

* ’subscribed’
Indicates that the circuit breaker is subscribed upstream to the flow and forwarding data and control packets through zero or more egress interfaces.

* ’pruned’
Indicates that the flow has been circuit-broken. A request to unsubscribe from the flow has been sent upstream, e.g. a PIM prune (section 3.5 of [RFC7761]) or a "leave" operation via IGMP, MLD, or another appropriate group membership protocol.

* ’probing’
Indicates that the flow was circuit-broken previously, and is currently joined upstream to refresh the most recent Bandwidth Advertisement in order to evaluate reinstating the flow.

probe timer Used to periodically probe a flow in the ’pruned’ state, to evaluate returning to ’forwarding’.

Flows additionally have a per-interface state for egress interfaces:

egress status One of the following values:

* ’forwarding’
Indicates that the flow is a non-circuit-broken flow in steady state, forwarding data and control packets downstream.

* ’blocked’

Indicates that data packets for this flow are NOT forwarded downstream via this interface. Bandwidth Advertisements are still forwarded, each with the 'Blocked' bit set to 1. All other flow traffic MUST be dropped.

5.4. Functionality

The CBACC building block on a sender MUST have access to the maximum bandwidth that may be sent at any time in the following 3 seconds. A CBACC sender MUST send this value in a Bandwidth Advertisement packet once per second. The end result of the traffic sent on the wire for a particular flow MUST honor this maximum bandwidth commitment, such that bandwidth measurements taken over any one-second period MUST NOT exceed any of prior 3 maximum Bandwidth Advertisements (or any of them, if fewer than 3 have been sent).

A CBACC circuit breaker MUST order its monitored flows based on per-flow estimates of network fairness and preferentially circuit break less fair flows when bandwidth limits are exceeded. A normative method to determine network fairness for a flow is out of scope for this document, but CBACC circuit breaker implementations SHOULD provide a capability for network operators to configure administrative biases for specific sets of flows, and network operators SHOULD consider fairness concerns as expressed in [RFC2914] section 3.2 and other relevant documents containing best practices.

In particular, fairness metrics SHOULD favor multicast flows with many receivers over multicast flows with few receivers and flows with low bandwidth over flows with high bandwidth. When receiver counts are known (for example via the experimental PIM extension specified in [RFC6807]) a RECOMMENDED metric is (bandwidth/receiver count), though other metrics MAY be used where deemed appropriate by network operators following internet best practices, or when receiver counts can't be determined.

Bandwidth Advertisement packets MUST NOT be sent by a sender more often than once per second.

If a circuit breaker receives more than 5 Bandwidth Advertisement packets for a flow in two seconds, the circuit breaker SHOULD set the flow to "pruned" and leave the upstream channel, and MUST drop Bandwidth Advertisement packets in excess of one per second.

Flows which are currently circuit-broken on an egress interface are set to "blocked". When an flow on an egress interface is in blocked state, Bandwidth Advertisement packets MUST be forwarded except as described in the preceding paragraph, the "Blocked" bit MUST be set

to 1 before forwarding, and other traffic for that flow MUST NOT be forwarded along that interface.

When a flow is blocked or pruned, the circuit breaker MAY truncate the Bandwidth Advertisement packet, keeping only the headers of the packet containing the Bandwidth Advertisement before forwarding.

When a flow is pruned, the circuit-breaker MUST generate and forward a Bandwidth Advertisement packet once per second with the "Blocked" bit set when there are still downstream receivers connected.

In flows which are not circuit-broken but which would be circuit-broken if the bandwidth warning threshold were the bandwidth limit, the Danger bit MUST be set to 1 before forwarding. Both data and control packets are forwarded for flows in this situation. The "Danger" bit MAY be used by receivers to take early action to avoid getting circuit-broken by shifting to a lower-bandwidth representation, if available.

When a flow is in the "blocked" state on every egress interface, the circuit breaker MAY set the flow to "pruned" on the ingress interface and leave the channel upstream.

In addition to monitoring the advertised bandwidth, a CBACC circuit breaker or other assisting nodes in the network SHOULD monitor the observed bandwidth per flow, and SHOULD circuit break "overactive" flows, defined as those which exceed their CBACC maximum bandwidth commitment. A circuit breaker MAY perform constant monitoring on all flows, or MAY use load sharing techniques such as random selection or round robin to monitor only a certain subset of flows at a time.

When detecting overactive flows, circuit breakers MUST use techniques to avoid false positives due to transient upstream network conditions such as packet compression or occasional packet duplication. For example, using an average of bandwidth measurements over the prior 3 seconds would qualify, where a half-second window would not. (A full listing of reasonable false-positive avoidance techniques is out of scope for this document.)

[TBD: examples with network diagrams and bandwidths?]

6. Requirements from other building blocks

The sender needs to know the bandwidth, including any upcoming changes, at least 3 seconds in advance. There is no requirement on how building blocks define this functionality except on the packets on the wire--the advance knowledge might, for example, be implemented by buffering and pacing on the sending machine. Specifics of the

sending bandwidth implementations are out of scope for this document, as it's intended to provide requirements that will be applicable to a broad range of possible implementations, including RTP and WEBRC.

7. IANA Considerations

This draft requests IANA to allocate an IPv6 packet header option number with the "action" bits set to "skip" and the "change" bit set to 1, as specified in [RFC2460] section 4.2. [TO BE REMOVED: This registration should take place at the following location: http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#extension-header.]

This draft also requests IANA to allocate an IPv4 packet header option number with the "copied" bit set and the option class "control", as specified in [RFC0791] section 3.1. [TO BE REMOVED: This registration should take place at the following location: http://www.iana.org/assignments/ip-parameters/ip-parameters.xhtml#ip-parameters-1.]

If those are deemed unacceptable, as an alternative with some compromises described in Section 5.2.2, this draft instead requests IANA to allocate a UDP destination port number. [TO BE REMOVED: This registration should take place at the following location: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.]

8. Security Considerations

8.1. Forged Packets

Forged Bandwidth Advertisement packets that get accepted by CBACC circuit breakers which dramatically over-report or under-report the correct bandwidth would present a potential DoS against a CBACC flow, by making the circuit breaker believe the flow exceeds the node's capacity when over-reporting, or by letting the node notice an apparent violation of the commitment to remain under the advertised bandwidth when under-reporting.

Similarly, it is possible to forge a CBACC Bandwidth Advertisement for a non-CBACC flow, which likewise may constitute a DoS against that flow.

For multicast, attacker would have to be on-path in order to deliver a forged packet to a CBACC circuit breaker, because the join's reverse path propagation will only reach the sender on a legitimate network path to its source address.

For unicast, it’s a bigger problem, because ANY sender along path that doesn’t have RPF check BCP 38 [RFC2827] permits attack on the flow via forged packet that substantially under-reports or over-reports bandwidth.

For AMT tunnels, when RPF checks along a path to the gateway are not present, nothing stops forged packets from being forwarded by the gateway. If these packets contain CBACC control packets, it’s possible to inject a forged packet into the network downstream from the gateway, combining the unicast hole with the multicast hole. This is a vulnerability that should probably be addressed by a new AMT version with some defense against forgery of data.

For IPSEC, since the Bandwidth Advertisement IP header option is mutable, it’s not protected by the IPSEC security services, so the Bandwidth Advertisement can be forged for consumption by the circuit breakers, even though the packet will be rejected by the end host with the security association. This could mount a DoS via the intermediate circuit-breakers by over-reporting or under-reporting flow bandwidth, when processing CBACC traffic through untrusted network paths.

The unicast vulnerabilities would be much mitigated by RPF checks as recommended by BCP 38 [RFC2827] at every hop, or otherwise maintained by the network. Absent such checks, cheap DoS vulnerabilities may be present from any permissive network locations.

8.2. Overloading of Slow Paths

CBACC control packets are sent as part of the data stream so that they traverse the same intermediate network nodes as the rest of the data, but they also carry control information that must be processed by certain nodes along that path.

This creates potential problems very similar to the problems with the Router Alert IP option discussed in Section 3 of [RFC6398], where a circuit-breaker might have a "fast path" for forwarding that can handle a much higher traffic volume than the "slow path" necessary to process CBACC control packets, which is potentially vulnerable to overloading.

If a CBACC-compatible circuit breaker receives a high rate of CBACC control packets, the circuit breaker MUST maintain network health for other flows. A circuit-breaker MAY drop all packets, including all CBACC control packets, for a flow in which more than 5 CBACC control packets were received in less than a second. (This number is intended to allow for moderate IP packet duplication and packet

compression by upstream routers, while still being slow enough for handling of packets on the slow path.)

8.3. Overloading of State

Since CBACC flows require state, it may be possible for a set of receivers and/or senders, possibly acting in concert, to generate many flows in an attempt to overflow the circuit breakers' state tables.

It is permissible for a network node to behave as a CBACC circuit breaker for some CBACC flows while treating other CBACC flows as non-CBACC, as part of a load balancing strategy for the network as a whole, or simply as defense against this concern when the number of monitored flows exceeds some threshold.

The same techniques described in section 3.1 of [RFC4609] can be used to help mitigate this attack, for much the same reasons. It is RECOMMENDED that network operators implement measures to mitigate such attacks.

9. Acknowledgements

Many thanks to Devin Anderson for a detailed review and many great suggestions. Thanks also to Cheng Jin and Scott Brown for early feedback.

10. References

10.1. Normative References

[IEEE.754.1985]
Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE Standard 754, August 1985.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, DOI 10.17487/RFC2113, February 1997, <http://www.rfc-editor.org/info/rfc2113>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, DOI 10.17487/RFC2711, October 1999, <http://www.rfc-editor.org/info/rfc2711>.

[RFC3048] Whetten, B., Vicisano, L., Kermode, R., Handley, M., Floyd, S., and M. Luby, "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", RFC 3048, DOI 10.17487/RFC3048, January 2001, <http://www.rfc-editor.org/info/rfc3048>.

[RFC3738] Luby, M. and V. Goyal, "Wave and Equation Based Rate Control (WEBRC) Building Block", RFC 3738, DOI 10.17487/RFC3738, April 2004, <http://www.rfc-editor.org/info/rfc3738>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <http://www.rfc-editor.org/info/rfc4302>.

[RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, DOI 10.17487/RFC4727, November 2006, <http://www.rfc-editor.org/info/rfc4727>.

[RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <http://www.rfc-editor.org/info/rfc7761>.

10.2. Informative References

[I-D.ietf-tsvwg-circuit-breaker] Fairhurst, G., "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-15 (work in progress), April 2016.

[I-D.ietf-tsvwg-rfc5405bis] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", draft-ietf-tsvwg-rfc5405bis-19 (work in progress), October 2016.

[PLM] A.Legout, E.W.Biersack, Institut EURECOM, "Fast Convergence for Cumulative Layered Multicast Transmission Schemes", 1999.

[RFC2357] Mankin, A., Romanow, A., Bradner, S., and V. Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols", RFC 2357, DOI 10.17487/RFC2357, June 1998, <http://www.rfc-editor.org/info/rfc2357>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <http://www.rfc-editor.org/info/rfc2784>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <http://www.rfc-editor.org/info/rfc2827>.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <http://www.rfc-editor.org/info/rfc2914>.

[RFC3269] Kermode, R. and L. Vicisano, "Author Guidelines for Reliable Multicast Transport (RMT) Building Blocks and Protocol Instantiation documents", RFC 3269, DOI 10.17487/RFC3269, April 2002, <http://www.rfc-editor.org/info/rfc3269>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <http://www.rfc-editor.org/info/rfc3552>.

[RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <http://www.rfc-editor.org/info/rfc3630>.

[RFC4609] Savola, P., Lehtonen, R., and D. Meyer, "Protocol Independent Multicast - Sparse Mode (PIM-SM) Multicast Routing Security Issues and Enhancements", RFC 4609, DOI 10.17487/RFC4609, October 2006, <http://www.rfc-editor.org/info/rfc4609>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6398] Le Faucheur, F., Ed., "IP Router Alert Considerations and Usage", BCP 168, RFC 6398, DOI 10.17487/RFC6398, October 2011, <http://www.rfc-editor.org/info/rfc6398>.

[RFC6807] Farinacci, D., Shepherd, G., Venaas, S., and Y. Cai, "Population Count Extensions to Protocol Independent Multicast (PIM)", RFC 6807, DOI 10.17487/RFC6807, December 2012, <http://www.rfc-editor.org/info/rfc6807>.

[RFC7450] Bumgardner, G., "Automatic Multicast Tunneling", RFC 7450, DOI 10.17487/RFC7450, February 2015, <http://www.rfc-editor.org/info/rfc7450>.

Appendix A. Overjoining

[I-D.ietf-tsvwg-rfc5405bis] describes several remedies for unicast congestion control under UDP, even though UDP does not itself provide congestion control. In general, any network node under congestion could in theory collect evidence that a unicast flow’s sending rate is not responding to congestion, and would then be justified in circuit-breaking it.

With multicast IP, the situation is different, especially in the presence of malicious receivers. A well-behaved sender using a receiver-controlled congestion scheme such as WEBRC does not reduce its send rate in response to congestion, instead relying on receivers to leave the appropriate multicast groups.

This leads to a situation where, when a network accepts inter-domain multicast traffic, as long as there are senders somewhere in the world with aggregate bandwidth that exceeds a network’s capacity, receivers in that network can join the flows and overflow the network capacity. A receiver controlled by an attacker could do this at the IGMP/MLD level without running the application layer protocol that participates in the receiver-controlled congestion control.

A network might be able to detect and defend against the most naive version of such an attack by blocking end users that try to join too many flows at once. However, an attacker can achieve the same effect by joining a few high-bandwidth flows, if those exist anywhere, and an attacker that controls a few machines in a network can coordinate the receivers so they join disjoint sets of non-responsive sending flows.

This scenario will produce congestion in a middle node in the network that can't be easily detected at the edge where the IGMP/MLD join is accepted. Thus, an attacker with a small set of machines in a target network can always trip a circuit breaker if present, or can induce excessive congestion among the bandwidth allocated to multicast. This problem gets worse as more multicast flows become available.

This is a significant barrier to multicast adoption because there is no present defense which does not itself constitute a denial of service attack.

Although the same can apply to non-responsive unicast traffic, network operators can assume that non-responsive sending flows are in violation of congestion control best practices, and can therefore cut off such flows. However, non-responsive multicast senders are likely to be well-behaved participants in receiver-controlled congestion control schemes.

However, receiver controlled congestion control schemes also show the most promise for efficient massive scale content distribution via multicast, provided network health can be ensured. Therefore, mechanisms to mitigate overjoining attacks while still permitting receiver-controlled congestion control are necessary.

TBD: network diagram

Figure 5

Author's Address

Jacob Holland
Akamai Technologies, Inc.
150 Broadway
Cambridge, Massachusetts 02142
USA

Phone: +1 617 444 3000
Email: jholland@akamai.com
URI: https://www.akamai.com/

Transport Area Working Group
Internet-Draft
Intended status: Experimental
Expires: October 23, 2017

J. Holland
Akamai Technologies, Inc.

April 21, 2017

Circuit Breaker Assisted Congestion Control (CBACC): Protocol Specification
draft-jholland-cb-assisted-cc-01

Abstract

This document specifies Circuit Breaker Assisted Congestion Control (CBACC), which provides bandwidth information from senders to intermediate network nodes to enable good decisions for fast-trip Network Transport Circuit Breaker activity ([I-D.ietf-tsvwg-circuit-breaker]) when necessary for network health. CBACC is specifically designed to support protocols using IP multicast, particularly as a supplement to receiver-driven congestion control protocols to help affected networks rapidly detect and mitigate the impact of scenarios in which a network is oversubscribed to flows which are not responsive to congestion.

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 October 23, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document specifies Circuit Breaker Assisted Congestion Control (CBACC).

CBACC is a congestion control building block designed for use with IP traffic that has a known maximum bandwidth, which does not reduce its sending rate in response to congestion. CBACC is specifically designed to supplement protocols using receiver-driven multicast congestion control systems that rely on well-behaved receivers to achieve congestion control in a very highly scalable system (up to millions of receivers) without a feedback path that reduces sending

rates by senders. Examples of congestion control systems fitting this description include PLM, RLM, RLC, FLID-DL, SMCC, ESMCC, QIRLM, and WEBRC [RFC3738].

CBACC addresses a vulnerability to "overjoining", a condition in which receivers (particularly malicious receivers) subscribe to traffic which, from the sending side, is non-responsive to congestion. Overjoining attacks and the challenges they present are discussed in more detail in Appendix A.

A careful reading of the congestion control requirements of UDP Best Practices [I-D.ietf-tsvwg-rfc5405bis] suggests that a network that forwards multicast traffic is required to operate a circuit breaker to maintain network health under a persistent overjoining condition, at a cost of cutting off some or all multicast traffic across the network during high congestion.

CBACC provides a mechanism for networks to mitigate the impact of overjoining within a network by introducing a mechanism for communicating the bandwidth of non-responsive flows from the sender of the flow to the transit nodes forwarding the flow. The bandwidth information is sufficient to implement a fast-trip circuit breaker [I-D.ietf-tsvwg-circuit-breaker] within a single network node which can specifically block or police flows when receivers have overjoined the network’s capacity.

In conjunction with receiver counts (e.g. via [RFC6807]) such nodes can also provide much improved network fairness for circuit breaking decisions during an overjoining condition.

In addition to streams using multicast receiver-driven congestion control, CBACC may also be suitable for use with other traffic, both unicast and multicast, that does not respond to congestion by reducing sending rates, including certain profiles of RTP [RFC3550] over either unicast or multicast, as well as several tunneling protocols (e.g. AMT [RFC7450] and GRE [RFC2784]) when they are known to carry traffic that would be suitable for CBACC. A complete specification for use of CBACC with unicast protocols and with tunneling protocols is out of scope for this document, though the security issues section does mention a few special considerations for potential unicast usage.

CBACC-compliant senders transmit Bandwidth Advertisements through the same transport path as the data traffic, so that circuit breakers can make informed decisions about how flows should be prioritized for circuit breaking. Additionally, CBACC-compliant circuit breakers transmit information to receivers about flows which have been or might soon be circuit-broken, to encourage CBACC-aware applications

to use alternate methods to retrieve equivalent (though probably lower-quality and possibly less efficient) data when possible.

This document describes a building block as defined in [RFC3048]. This document describes a congestion control building block that conforms to [RFC2357]. This document follows the general guidelines provided in [RFC3269], in addition to the requirements on RFCs from [RFC5226] and [RFC3552].

2. Terminology

Term	Definition
circuit breaker	See [I-D.ietf-tsvwg-circuit-breaker]
controlled environment	See [I-D.ietf-tsvwg-rfc5405bis] Section 3.6
general	See [I-D.ietf-tsvwg-rfc5405bis] Section 3.6
internet flow	traffic for a single (source,destination) IP pair, including destinations that are group addresses
upstream	along a network topology path in the direction of a flow's sender
downstream	along a network topology path in the direction of a flow's receiver
ingress interface	the (single) upstream interface for a flow in a circuit breaker
egress interface	a downstream interface for a flow in a circuit breaker

Table 1

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 [RFC2119].

3. Rationale

CBACC is defined as an independent congestion control building block because it would be a useful supplement a wide variety of receiver-driven multicast congestion control schemes, such as [PLM] or other methods based on receiver-driven conformance to a measurement of available network bandwidth or congestion.

CBACC is also potentially valuable, even without other congestion control systems, in controlled environments where congestion control

may not be required (e.g. for certain profiles of RTP [RFC3550]), since CBACC can provide protection for such a network against congestion due to sender or network mis-configuration.

CBACC provides a new form of communication between senders and network transit nodes to facilitate fast-trip circuit breakers as described in section 5.1 of [I-D.ietf-tsvwg-circuit-breaker] which are not available via previously existing methods. When used in conjunction with compatible circuit breakers, CBACC can greatly improve the safety of a network that accepts and delivers interdomain massively scalable multicast traffic to potentially untrusted receivers.

4. Applicability

CBACC relies on the presence of CBACC-aware circuit breakers on a flow's transit path in order to provide congestion control in a network. In the absence of any CBACC-aware circuit breakers on a network path, CBACC constitutes a small extra overhead to a flow without providing any additional value.

CBACC provides a form of congestion control for massively scalable protocols using the IP multicast service. CBACC is best used in conjunction with another receiver-driven multicast congestion control, but it is also suitable for use even without another congestion control mechanism, or when presence of another congestion control mechanism is unproven, such as when accepting multicast joins from untrusted receivers.

5. Protocol Specification

5.1. Overview

CBACC senders send Bandwidth Advertisement packets to advertise the maximum sending bandwidth along the data path for a flow through a network.

CBACC bandwidth information is monitored by CBACC circuit breakers along the network path, which may block the forwarding of traffic for some flows in order to maintain network health. When a flow is blocked, a CBACC circuit breaker sets a bit in Bandwidth Advertisement packets before they're forwarded downstream that indicates to subscribed receivers of that flow that traffic has been blocked.

The protocol also defines a way to notify downstream receivers when a flow is in danger of being circuit broken in the near future. A CBACC-capable transport node SHOULD send this information when it is

known, as described in section [TBD]. This gives applications an opportunity to gracefully shift to a lower-bandwidth version of the same content, when possible, providing an early warning system for avoiding congestion more smoothly.

A Bandwidth Advertisement packet constitutes an "ingress meter" as described in section 3.1 of [I-D.ietf-tsvwg-circuit-breaker]. The configured bandwidth caps of egress interfaces likewise constitute "egress meters". However, the diagram in the referenced document is simplified by running the ingress and egress on the same network node. At the CBACC-aware circuit breaker, the CBACC node has both pieces of information as soon as a Bandwidth Advertisement is received, and can trip the circuit breaker if the aggregate advertised CBACC bandwidth exceeds the actual bandwidth available on any egress interfaces.

5.2. Packet Header Fields

5.2.1. Bandwidth Advertisement

5.2.1.1. As an IP header option

Bandwidth advertisements can appear as either an IPv4 header option (as in Section 3.1 of [RFC0791]) or as an IPv6 extension header option (as in section 4.2 of [RFC2460]). They have the same layout:

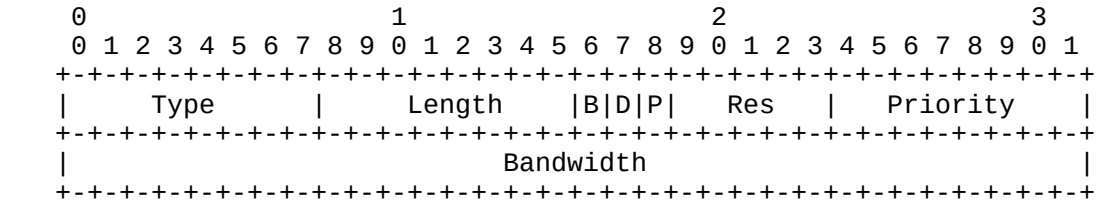


Figure 1

Bandwidth advertisements sent as IPv4 header options use option value [TBD], with the "copied" bit set and the option class "control", as specified in [RFC0791] section 3.1. Until and unless IANA assigns a value, this will be option number 158 as described in section 8 of [RFC4727] for experiments using IPv4 Option types. The length field is 8.

Bandwidth advertisements sent as IPv6 header options use option value [TBD], with the "action" bits set to "skip" and the "change" bit set to 1, as specified in [RFC2460] section 4.2. Until and unless IANA assigns a value, this will be option number 0x3e as described in

section 8 of [RFC4727] for experiments using IPv6 Option Types. The length field is 6.

Using an IP header option has the benefit of exposing the bandwidth to all CBACC-compatible routers, in much the same way the IP Router Alert option would, but without being processed or causing undue load in non-CBACC routers.

The IP Header encapsulations DO work with IPSEC. As described in Appendix A of [RFC4302], the IP header fields are properly treated as mutable and zeroed for the IPSEC ICV calculations. CBACC circuit breakers MAY change bits in transit. The Bandwidth Advertisement header itself IS NOT protected by IPSEC security services, but protection of other parts of the packet remain unchanged.

5.2.1.2. Field definitions

5.2.1.2.1. Bandwidth

As in several other protocols sending bandwidth values such as OSPF-TE [RFC3630], the bandwidth is expressed in bytes per second (not bits), in IEEE floating point format. For quick reference, this format is as follows:

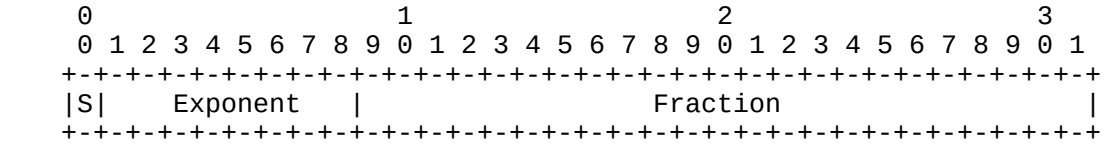


Figure 2

S is the sign, Exponent is the exponent base 2 in "excess 127" notation, and Fraction is the mantissa - 1, with an implied binary point in front of it. Thus, the above represents the value:

(-1)**(S) * 2**(Exponent-127) * (1 + Fraction)

For more details, refer to [IEEE.754.1985].

Figure 3

5.2.1.2.2. B (Blocked) bit

Indicates that the flow has been circuit-broken.

5.2.1.2.3. D (Danger) bit

Indicates that the flow is in danger of being circuit-broken.

5.2.1.2.4. P (Police) bit

Indicates that the flow should be policed instead of blocked. Flows marked for policing by the sender should have traffic proportionally dropped when bandwidth is needed, according to their priority. [TBD] Flesh this concept out, and decide whether it's actually viable. This was my attempt at addressing a suggestion from Bob Briscoe at IETF 97 in ICCRG at the mic, IIRC. It probably requires more state, such as total desired policable bandwidth, total current policed bandwidth, and current policing bandwidth per-flow, plus some definition of how to decide between cutting off some flows and policing others. This may not be worth the hassle, but there are some use cases such as FEC repair traffic which might actually be nicer this way. However, it might also be possible to get the same effect by assigning priority to those repair flows. Things like video enhancement layers of course are probably better done as a complete cutoff.

5.2.1.2.5. Res (Reserved bits)

The sender MUST set all reserved bits to 0 when sending a CBACC control packet. Receivers and CBACC-capable transit nodes MUST accept any value in the reserved bits.

5.2.1.2.6. Priority

The sender MAY indicate relative priorities of different streams from the same sender with this field. This is an 8-bit unsigned integer, and higher values are kept preferentially over other traffic from the same sender with lower priority values, so all flows with a lower priority value are circuit-broken before any flows with a higher priority value. Among multiple flows from the same sender with the same priority, the highest bandwidth flows are circuit- broken first.

5.3. States

5.3.1. Interface State

A CBACC circuit breaker holds the following state for each interface, for both the inbound and outbound directions on that interface:

- aggregate bandwidth: The sum of the bandwidths of all non-circuit-broken CBACC flows which transit this interface in this direction.

- bandwidth limit: The maximum aggregate CBACC advertised bandwidth allowed, not including circuit-broken flows. This may depend on administrative configuration and congestion measurements for the network, whether from this node or other nodes. It's out of scope for this document to define such congestion measurements. Network operators should carefully consider that this bandwidth limit applies to flows that are unresponsive to congestion.

  When reducing the bandwidth limit due to congestion, the circuit breaker MUST NOT reduce the limit by more than half its value in 10 seconds, and SHOULD use a smoothing function to reduce the limit gradually over time.

  It is RECOMMENDED that no more than half the capacity for a link be allocated to CBACC flows if the link might be shared with TCP or other traffic that is responsive to congestion.

  Depending on administrative configuration and the physical characteristics of the interface, the bandwidth limit may be either shared between upstream and downstream traffic, or it may be separate. Either a single shared value should be used, or two separate independent values should be used for the inbound and outbound directions for an interface.

- CBACC bandwidth warning threshold: A soft bandwidth threshold. When the aggregate CBACC advertised bandwidth exceeds this threshold, flows that would have been circuit-broken with a bandwidth limit at this threshold MUST have the Danger bit set in the Bandwidth Advertisement packets that are forwarded by this circuit breaker. This threshold SHOULD be configurable as a proportion of the bandwidth limit, and MUST remain at or below the bandwidth limit when the bandwidth limit changes. The recommended proportion value is .75, but specific networks may use a different value if deemed useful by the network operators.

5.3.2. Flow State

The following state is kept for flows that are joined from at least one downstream interface and for which at least one CBACC Bandwidth Advertisement packet has been received:

- bandwidth: The bandwidth from the most receintly received Bandwidth Advertisement.

- ingress status: One of the following values:

  * 'subscribed'

Indicates that the circuit breaker is subscribed upstream to the flow and forwarding data and control packets through zero or more egress interfaces.

* 'pruned'
Indicates that the flow has been circuit-broken. A request to unsubscribe from the flow has been sent upstream, e.g. a PIM prune (section 3.5 of [RFC7761]) or a "leave" operation via IGMP, MLD, or another appropriate group membership protocol.

* 'probing'
Indicates that the flow was circuit-broken previously, and is currently joined upstream to refresh the most recent Bandwidth Advertisement in order to evaluate reinstating the flow.

- probe timer: Used to periodically probe a flow in the 'pruned' state, to evaluate returning to 'forwarding'.

Flows additionally have a per-interface state for egress interfaces:

- egress status: One of the following values:

  * 'forwarding'
  Indicates that the flow is a non-circuit-broken flow in steady state, forwarding data and control packets downstream.

  * 'blocked'
  Indicates that data packets for this flow are NOT forwarded downstream via this interface. Bandwidth Advertisements are still forwarded, each with the 'Blocked' bit set to 1. All other flow traffic MUST be dropped.

5.4. Functionality

The CBACC building block on a sender MUST have access to the maximum bandwidth that may be sent at any time in the following 3 seconds. A CBACC sender MUST send this value in a Bandwidth Advertisement packet once per second. The end result of the traffic sent on the wire for a particular flow MUST honor this maximum bandwidth commitment, such that bandwidth measurements taken over any sliding window one-second period MUST NOT exceed any of prior 3 maximum Bandwidth Advertisements (or any of them, if fewer than 3 have been sent).

A CBACC circuit breaker MUST order its monitored flows based on per-flow estimates of network fairness and preferentially circuit break less fair flows when bandwidth limits are exceeded. A normative method to determine network fairness for a flow is out of scope for this document, but CBACC circuit breaker implementations SHOULD

provide a capability for network operators to configure administrative biases for specific sets of flows, and network operators SHOULD consider fairness concerns as expressed in [RFC2914] section 3.2 and other relevant documents describing best practices.

In particular, fairness metrics SHOULD favor multicast flows with many receivers over multicast flows with few receivers and flows with low bandwidth over flows with high bandwidth. When receiver counts are known (for example via the experimental PIM extension specified in [RFC6807]) a RECOMMENDED metric is (bandwidth/receiver count), though other metrics MAY be used where deemed appropriate by network operators following internet best practices, or when receiver counts can’t be determined.

A CBACC sender MUST send Bandwidth Advertisements once per second. (Implementation-specific jitter in timer implementations not exceeding .1s is acceptable.)

If a circuit breaker receives more than 5 Bandwidth Advertisement packets for a flow in two seconds, the circuit breaker SHOULD set the flow to "pruned" and leave the upstream channel, and MUST drop Bandwidth Advertisement packets in excess of one per second.

Flows which are currently circuit-broken on an egress interface are set to "blocked". When a flow on an egress interface is in blocked state, Bandwidth Advertisement packets MUST be forwarded except as described in the preceding paragraph, the "Blocked" bit MUST be set to 1 before forwarding, and other traffic for that flow MUST NOT be forwarded along that interface.

When a flow is blocked or pruned, the circuit breaker MAY truncate the Bandwidth Advertisement packet, keeping only the headers of the packet containing the Bandwidth Advertisement before forwarding.

When a flow is pruned, the circuit-breaker MUST generate and forward a Bandwidth Advertisement packet once per second with the "Blocked" bit set when there are still downstream receivers connected.

In flows which are not circuit-broken but which would be circuit-broken if the bandwidth warning threshold were the bandwidth limit, the Danger bit MUST be set to 1 before forwarding. Both data and control packets are forwarded for flows in this situation. The "Danger" bit MAY be used by receivers to take early action to avoid getting circuit-broken by shifting to a lower-bandwidth representation, if available.

When a flow is in the "blocked" state on every egress interface, the circuit breaker MAY set the flow to "pruned" on the ingress interface and leave the channel upstream.

In addition to monitoring the advertised bandwidth, a CBACC circuit breaker or other assisting nodes in the network SHOULD monitor the observed bandwidth per flow, and SHOULD circuit break "overactive" flows, defined as those which exceed their CBACC maximum bandwidth commitment. A circuit breaker MAY perform constant monitoring on all flows, or MAY use load sharing techniques such as random selection or round robin to monitor only a certain subset of flows at a time.

When detecting overactive flows, circuit breakers MUST use techniques to avoid false positives due to transient upstream network conditions such as packet compression or occasional packet duplication. For example, using an average of bandwidth measurements over the prior 3 seconds would qualify, where a half-second window would not. (A full listing of reasonable false-positive avoidance techniques is out of scope for this document.)

[TBD: examples with network diagrams and bandwidths?] [TBD: some internal structure on this section. "wall of text" was some feedback]

6. Requirements from other building blocks

The sender needs to know the bandwidth, including any upcoming changes, at least 3 seconds in advance. There is no requirement on how building blocks define this functionality except on the packets on the wire--the advance knowledge might, for example, be implemented by buffering and pacing on the sending machine. Specifics of the sending bandwidth implementations are out of scope for this document, as it’s intended to provide requirements that will be applicable to a broad range of possible implementations, including RTP and WEBRC.

7. IANA Considerations

This draft requests IANA to allocate an IPv6 packet header option number with the "action" bits set to "skip" and the "change" bit set to 1, as specified in [RFC2460] section 4.2. [TO BE REMOVED: This registration should take place at the following location: http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#extension-header.]

This draft also requests IANA to allocate an IPv4 packet header option number with the "copied" bit set and the option class "control", as specified in [RFC0791] section 3.1. [TO BE REMOVED: This registration should take place at the following location:

http://www.iana.org/assignments/ip-parameters/ip-parameters.xhtml#ip-parameters-1.]

If those are deemed unacceptable, as an alternative with some compromises described in Section 5.2.1, this draft instead requests IANA to allocate a UDP destination port number. [TO BE REMOVED: This registration should take place at the following location: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.]

8. Security Considerations

8.1. Forged Packets

Forged Bandwidth Advertisement packets that get accepted by CBACC circuit breakers which dramatically over-report or under-report the correct bandwidth would present a potential DoS against a CBACC flow, by making the circuit breaker believe the flow exceeds the node's capacity when over-reporting, or by letting the node notice an apparent violation of the commitment to remain under the advertised bandwidth when under-reporting.

Similarly, it is possible to forge a CBACC Bandwidth Advertisement for a non-CBACC flow, which likewise may constitute a DoS against that flow.

For multicast, attacker would have to be on-path in order to deliver a forged packet to a CBACC circuit breaker, because the join's reverse path propagation will only reach the sender on a legitimate network path to its source address.

For unicast, it's a bigger problem, because ANY sender along path that doesn't have RPF check BCP 38 [RFC2827] permits attack on the flow via forged packet that substantially under-reports or over-reports bandwidth.

For AMT tunnels, when RPF checks along a path to the gateway are not present, nothing stops forged packets from being forwarded by the gateway. If these packets contain CBACC control packets, it's possible to inject a forged packet into the network downstream from the gateway, combining the unicast hole with the multicast hole. This is a vulnerability that should probably be addressed by a new AMT version with some defense against forgery of data.

For IPSEC, since the Bandwidth Advertisement IP header option is mutable, it's not protected by the IPSEC security services, so the Bandwidth Advertisement can be forged for consumption by the circuit breakers, even though the packet will be rejected by the end host

with the security association. This could mount a DoS via the intermediate circuit-breakers by over-reporting or under-reporting flow bandwidth, when processing CBACC traffic through untrusted network paths.

The unicast vulnerabilities would be much mitigated by RPF checks as recommended by BCP 38 [RFC2827] at every hop, or otherwise maintained by the network. Absent such checks, cheap DoS vulnerabilities may be present from any permissive network locations.

8.2. Overloading of Slow Paths

CBACC control packets are sent as part of the data stream so that they traverse the same intermediate network nodes as the rest of the data, but they also carry control information that must be processed by certain nodes along that path.

This creates potential problems very similar to the problems with the Router Alert IP option discussed in Section 3 of [RFC6398], where a circuit-breaker might have a "fast path" for forwarding that can handle a much higher traffic volume than the "slow path" necessary to process CBACC control packets, which is potentially vulnerable to overloading.

If a CBACC-compatible circuit breaker receives a high rate of CBACC control packets, the circuit breaker MUST maintain network health for other flows. A circuit-breaker MAY drop all packets, including all CBACC control packets, for a flow in which more than 5 CBACC control packets were received in less than a second. (This number is intended to allow for moderate IP packet duplication and packet compression by upstream routers, while still being slow enough for handling of packets on the slow path.)

8.3. Overloading of State

Since CBACC flows require state, it may be possible for a set of receivers and/or senders, possibly acting in concert, to generate many flows in an attempt to overflow the circuit breakers' state tables.

It is permissible for a network node to behave as a CBACC circuit breaker for some CBACC flows while treating other CBACC flows as non-CBACC, as part of a load balancing strategy for the network as a whole, or simply as defense against this concern when the number of monitored flows exceeds some threshold.

The same techniques described in section 3.1 of [RFC4609] can be used to help mitigate this attack, for much the same reasons. It is

RECOMMENDED that network operators implement measures to mitigate such attacks.

9. Acknowledgements

Many thanks to Devin Anderson and Ben Kaduk for detailed reviews and many great suggestions. Thanks also to Cheng Jin, Scott Brown, Miroslav Kaduk, and Bob Briscoe for their thoughtful contributions.

10. References

10.1. Normative References

[IEEE.754.1985] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE Standard 754, August 1985.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC3048] Whetten, B., Vicisano, L., Kermode, R., Handley, M., Floyd, S., and M. Luby, "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", RFC 3048, DOI 10.17487/RFC3048, January 2001, <http://www.rfc-editor.org/info/rfc3048>.

[RFC3738] Luby, M. and V. Goyal, "Wave and Equation Based Rate Control (WEBRC) Building Block", RFC 3738, DOI 10.17487/RFC3738, April 2004, <http://www.rfc-editor.org/info/rfc3738>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <http://www.rfc-editor.org/info/rfc4302>.

[RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, DOI 10.17487/RFC4727, November 2006, <http://www.rfc-editor.org/info/rfc4727>.

[RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <http://www.rfc-editor.org/info/rfc7761>.

10.2. Informative References

[I-D.ietf-tsvwg-circuit-breaker] Fairhurst, G., "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-15 (work in progress), April 2016.

[I-D.ietf-tsvwg-rfc5405bis] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", draft-ietf-tsvwg-rfc5405bis-19 (work in progress), October 2016.

[PLM] A.Legout, E.W.Biersack, Institut EURECOM, "Fast Convergence for Cumulative Layered Multicast Transmission Schemes", 1999.

[RFC2357] Mankin, A., Romanow, A., Bradner, S., and V. Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols", RFC 2357, DOI 10.17487/RFC2357, June 1998, <http://www.rfc-editor.org/info/rfc2357>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <http://www.rfc-editor.org/info/rfc2784>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <http://www.rfc-editor.org/info/rfc2827>.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <http://www.rfc-editor.org/info/rfc2914>.

[RFC3269] Kermode, R. and L. Vicisano, "Author Guidelines for Reliable Multicast Transport (RMT) Building Blocks and Protocol Instantiation documents", RFC 3269, DOI 10.17487/RFC3269, April 2002, <http://www.rfc-editor.org/info/rfc3269>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <http://www.rfc-editor.org/info/rfc3552>.

[RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <http://www.rfc-editor.org/info/rfc3630>.

[RFC4609] Savola, P., Lehtonen, R., and D. Meyer, "Protocol Independent Multicast - Sparse Mode (PIM-SM) Multicast Routing Security Issues and Enhancements", RFC 4609, DOI 10.17487/RFC4609, October 2006, <http://www.rfc-editor.org/info/rfc4609>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6398] Le Faucheur, F., Ed., "IP Router Alert Considerations and Usage", BCP 168, RFC 6398, DOI 10.17487/RFC6398, October 2011, <http://www.rfc-editor.org/info/rfc6398>.

[RFC6807] Farinacci, D., Shepherd, G., Venaas, S., and Y. Cai, "Population Count Extensions to Protocol Independent Multicast (PIM)", RFC 6807, DOI 10.17487/RFC6807, December 2012, <http://www.rfc-editor.org/info/rfc6807>.

[RFC7450] Bumgardner, G., "Automatic Multicast Tunneling", RFC 7450, DOI 10.17487/RFC7450, February 2015, <http://www.rfc-editor.org/info/rfc7450>.

Appendix A. Overjoining

[I-D.ietf-tsvwg-rfc5405bis] describes several remedies for unicast congestion control under UDP, even though UDP does not itself provide congestion control. In general, any network node under congestion could in theory collect evidence that a unicast flow's sending rate is not responding to congestion, and would then be justified in circuit-breaking it.

With multicast IP, the situation is different, especially in the presence of malicious receivers. A well-behaved sender using a receiver-controlled congestion scheme such as WEBRC does not reduce its send rate in response to congestion, instead relying on receivers to leave the appropriate multicast groups.

This leads to a situation where, when a network accepts inter-domain multicast traffic, as long as there are senders somewhere in the world with aggregate bandwidth that exceeds a network's capacity, receivers in that network can join the flows and overflow the network capacity. A receiver controlled by an attacker could do this at the IGMP/MLD level without running the application layer protocol that participates in the receiver-controlled congestion control.

A network might be able to detect and defend against the most naive version of such an attack by blocking end users that try to join too many flows at once. However, an attacker can achieve the same effect by joining a few high-bandwidth flows, if those exist anywhere, and an attacker that controls a few machines in a network can coordinate the receivers so they join disjoint sets of non-responsive sending flows.

This scenario will produce congestion in a middle node in the network that can't be easily detected at the edge where the IGMP/MLD join is accepted. Thus, an attacker with a small set of machines in a target network can always trip a circuit breaker if present, or can induce excessive congestion among the bandwidth allocated to multicast. This problem gets worse as more multicast flows become available.

This is a significant barrier to multicast adoption because there is no present defense which does not itself constitute a denial of service attack.

Although the same can apply to non-responsive unicast traffic, network operators can assume that non-responsive sending flows are in violation of congestion control best practices, and can therefore cut off such flows. However, non-responsive multicast senders are likely to be well-behaved participants in receiver-controlled congestion control schemes.

However, receiver controlled congestion control schemes also show the most promise for efficient massive scale content distribution via multicast, provided network health can be ensured. Therefore, mechanisms to mitigate overjoining attacks while still permitting receiver-controlled congestion control are necessary. [TBD: this whole section should be expanded and moved to a separate informational draft]

TBD: network diagram

Figure 4

Author's Address

Jacob Holland
Akamai Technologies, Inc.
150 Broadway
Cambridge, Massachusetts 02142
USA

Phone: +1 617 444 3000
Email: jholland@akamai.com
URI: https://www.akamai.com/

Internet Congestion Control Research Group
Internet-Draft
Intended status: Experimental
Expires: May 4, 2017

M. Welzl
S. Islam
K. Hiorth
University of Oslo
J. You
Huawei

October 31, 2016

TCP-CCC: single-path TCP congestion control coupling
draft-welzl-tcp-ccc-00

Abstract

This document specifies a method, TCP-CCC, to combine the congestion controls of multiple TCP connections between the same pair of hosts. This can have several performance benefits, and it makes it possible to precisely assign a share of the congestion window to the connections based on priorities. This document also addresses the problem that TCP connections between the same pair of hosts may not share the same path. We discuss methods to detect if, or enforce that connections traverse a common bottleneck.

Requirements Language

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 [RFC2119].

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 May 4, 2017.

Table of Contents

1. Introduction

When multiple TCP connections between the same host pair compete on the same bottleneck, they often incur more delay and losses than a single TCP connection. Moreover, it is often not possible to precisely divide the available capacity among the connections. To address this problem, this document presents TCP-CCC, a method to combine the congestion controls of multiple TCP connections between the same pair of hosts. This can have several performance benefits:

- Reduced average loss and queuing delay (because the competition between the encapsulated TCP connections is avoided)

- Assign a precise capacity share based on a priority.

- Even in the absence of prioritization, better fairness between the TCP connections.

- No need for new connections to slow start up to a reasonable cwnd value that ongoing connections already have: a connection can immediately be assigned its share of the aggregate's total cwnd. This can significantly reduce the completion time of short connections.

All of these benefits only play out when there are more than one TCP connections. Some of the benefits in the list above are more significant when some transfers are short. This makes the usage of TCP-CCC especially attractive in situations where some transfers are short.

We discuss methods to determine if connections traverse the same bottleneck as well as methods to ensure this. To this end, we propose a light-weight, dynamically configured TCP-in-UDP (TiU) encapsulation scheme. TiU is optional, as our coupled congestion control strategy is applicable wherever overlapping TCP flows must follow the same path (such as when routed over a VPN tunnel).

2. Coupled Congestion Control

For each TCP connection c, the algorithm described below receives cwnd and ssthresh as input and stores the following information:

- the Connection ID.

- a priority P(c) -- e.g., an integer value in the range from 1 (unimportant) to 10 (very important).

- The previously used cwnd used by the connection c, ccc_cwnd(c).

- The previously used ssthresh used by the connection c, ccc_ssthresh(c).

Three global variables sum_cwnd, sum_ssthresh and sum_p are used to represent the sum of all the ccc_cwnd values, ccc_sshtresh values and priorities of all TCP connections, respectively. sum_cwnd and sum_ssthresh are used to update the cwnd and ssthresh values for all connections.

This algorithm emulates the behavior of a single TCP connection by choosing one connection as the connection that dictates the increase / decrease behavior for the aggregate. We call it the "Coordinating Connection" (CoCo). The algorithm was designed to be as simple as possible. Below, abbreviations are used to refer to the phases of

TCP congestion control as defined in [RFC5681]: SS refers to Slow Start, CA refers to Congestion Avoidance and FR refers to Fast Recovery.

For simplicity, this algorithm refrains from changing cwnd when a connection is in FR. SS should not happen as long as ACKs arrive. Hence, the algorithm ensures that the aggregate's behavior is only dictated by SS when all connections are in the SS phase. We use a bit array, ssbits, with a bit for each connection in the group. We set the bit if the connection state is SS due to an RTO.

(1) When a connection c starts, it adds its priority P(c) to sum_p. If it is the very first connection, it sets sum_cwnd to its own cwnd. After that, the connection's globally known cwnd and ssthresh values (ccc_cwnd(c) and ccc_ssthresh(c)) are updated, and the connection updates its own cwnd and ssthresh values to be equal to ccc_cwnd(c) and ccc_ssthresh(c).

```
ccc_P(c) = P
sum_P = sum_P + P
sum_cwnd sum_cwnd + cwnd
ccc_cwnd(c) P = sum_cwnd / sum_P
ccc_ssthresh(c) = ssthresh
if sum_ssthresh > 0 then
    ccc_ssthresh(c) P = sum_ssthresh / sum_P
end if
// Update c's own cwnd and ssthresh for immediate use:
Send ccc_cwnd(c) and ccc_ssthresh(c) to c
```

(2) When a connection c stops, its entry is removed. sum_p is recalculated.

```
if c = CoCo then
    Coco = the next connection
end if
sum_p sum_p - ccc_P(c)
Remove ccc_P(c), ccc_cwnd(c), ccc_ssthresh(c)
```

(3) Every time the congestion controller of a connection c calculates a new cwnd, the connection calls UPDATE, which carries out the tasks listed below to derive the new cwnd and ssthresh values. Whenever the CoCo calls UPDATE, sum_cwnd and sum_ssthresh are additionally updated to reflect the current sum of all stored ccc_cwnd and ccc_ssthresh values. Initially,

there is only one connection and this connection automatically becomes the CoCo. It updates sum_cwnd to its own cwnd and sets sum_ssthresh to 0.

(4) WHEN a non-CoCo connection c CALLS UPDATE......

```
if(all of the connections including CoCo are in CA but c is in FR)
    c becomes the new CoCo.
else
    if(c is in CA or SS)
        c's cwnd is assigned its previously stored ccc_cwnd value.
```

(5) WHEN c(CoCo) CALLS UPDATE......

```
if CoCo == c then
    if state == CA and ssbits(c) == 0 then
        if cwnd >= ccc_cwnd(c) then // increased cwnd
            sum_cwnd = sum_cwnd + cwnd - ccc_cwnd(c)
        else
            sum_cwnd = sum_cwnd * cwnd / ccc_cwnd(c)
        end if
        ccc_cwnd(c) = ccc_P(c) * sum_cwnd / sum_p
        ccc_ssthresh(c) ssthresh
        if sum_ssthresh > 0 then
            ccc_ssthresh(c) ccc_P(c) * sum_ssthresh/sum_p
        end if
    else if state == FR then
        sum_ssthresh = sum_cwnd/2
    else if state == SS then
        if c experienced a timeout then
            ssbits(c) = 1
        end if
        if ssbits(x) == 1 for all x then
            ssbits(x) = 0 // for all x
            sum_cwnd = sum_cwnd * cwnd / ccc_cwnd(c)
            ccc_cwnd(c) = ccc_P(c) * sum_cwnd / sum_p
            sum_ssthresh = sum_cwnd/2
        else
            CoCo = first connection where ccc_state == SS
        end if
    end if
end if
```

(6) After that, if the ccc_state(c) is not equal to FR

```
if state != FR then
    Send ccc_cwnd(c) and ccc_ssthresh(c) to c
end if
```

When a flow gets a large share of the aggregate immediately after joining, it can potentially create a burst in the network. We propose a mechanism [anrw2016] to clock the packet transmission out by using the ack-clock of TCP. Our algorithm achieves a form of "pacing", but it does not rely on any timers.

When a connection c joins, it turns on the ack-clock feature and calculates the share of the aggregate, clocked_cwnd c. Below, we illustrate the ack-clock mechanism that is used to distribute the share of the cwnd based on the acknowledgements received from other flows.

```
if clocked_cwnd(c) <= 0 then
    return // alg. ends; other connections can increase cwnd again
end if
if number_of_acks c % N = 0 then
    send a new segment for connection c
    clocked_cwnd(c)= clocked_cwnd(c) - 1
end if
number_of_acks(c) = number_of_acks(c) + 1
```

3. Ensuring a Common Bottleneck

Our algorithm, as well as EFCM [EFCM], E-TCP [EFCM] and the CM [RFC3124] assume that multiple TCP connections between the same host pair traverse the same bottleneck. This is not always true: load-balancing mechanisms such as Link Aggregation Group (LAG) and Equal-Cost Multi-Path (ECMP) may force them to take different paths [RFC7424]. If this leads to the connections seeing different bottlenecks, combining the congestion controllers would incur wrong behavior. There are, however, several application scenarios where the single-bottleneck assumption is correct.

Sometimes, the network configuration is known, and it is known that mechanisms such as ECMP and LAG do not operate on the bottleneck or are simply not in use. Alternatively, measurements can infer whether flows traverse the same bottleneck [I-D.ietf-rmcat-sbd]. When IPv6 is available, the TCP connections could be assigned the same IPv6 flow label. According to [RFC6437], "The usage of the 3-tuple of the Flow Label, Source Address, and Destination Address fields enables efficient IPv6 flow classification, where only IPv6 main header

fields in fixed positions are used" - this would be favorable for TCP congestion control coupling. However, this [RFC6437] does not make a clear recommendation about either using the 3-tuple or 5-tuple (which includes the port numbers) - both methods are valid. Thus, whether it works to use the flow label as the sole means to put connections on the same path depends on router configuration. When it works, it is an attractive option because it does not require changing the receiver.

Finally, encapsulating packets with a header that ensures a common path is another possibility to make connections traverse the same bottleneck. We will discuss encapsulation in the next section.

3.1. Encapsulation

3.1.1. TCP in UDP

3.1.1.1. Introduction

We want to be able to ensure that TCP congestion control coupling can always work, provided that the required code is available at the receiver - and be able to efficiently fall back to the standard behaviour in case it is not. To achieve this, we present a method, TCP-in-UDP (TiU), to encapsulate multiple TCP connections using the same UDP port pair.

TCP-in-UDP (TiU) is based on [Che13]. It differs from it in that:

- Other than [Che13], TiU encapsulates multiple TCP connections using the same UDP port number pair. TCP port numbers are preserved; a single well-known UDP port is used for TiU. If TiU is implemented in the kernel, this allows using normal TCP sockets, where enabling the usage of TiU could be done via a socket option, for example.

- The header format is slightly different to allow representing a TCP connection with a few bits that are encoded across the original TCP header’s "Reserved" field and the URG (Urgent) flag to encode a Connection ID. With this encoding, similar to the encapsulation in [Che13], the total TiU header size does not exceed the original TCP header size.

- A (TiU-encapsulated) TCP SYN uses a newly defined TCP option to establish the mapping between a Connection ID and the original TCP port number pair.

TiU inherits all the benefits of [Che13] and a preceding similar proposal, [Den08]. It enables TCP-CCC coupled congestion control,

and it adds the potential disadvantage of not being able to benefit from ECMP. In short, the benefits and features of TiU that are already explained in detail in [Che13] and [Den08] are:

- To establish direct communication between two devices that are both behind NAT gateways, Interactive Connectivity Establishment (ICE) [RFC5245] is used to create the necessary mappings in both NAT gateways, and ICE can have higher success rates using UDP [RFC5128].

- TCP options, as required for Multipath TCP [RFC6824], for example, are expected to work more reliably because middleboxes will be less able to interfere with them.

- Because the packet format allows the first octet to be in the range 0x0-0x3 (as is the case for a STUN [RFC5389] packet, where the most significant two bits are always zero), the UDP port number pair used by TiU can be used to exchange STUN packets with a STUN server that is unaware of TiU.

- Following the method described in [Che13] and [Den08], other transport protocols than TCP (e.g., SCTP) could be UDP-encapsulated in a similar fashion. With TiU, the same outer UDP port number pair could be used for different encapsulated protocols at the same time.

[Che13] also lists a disadvantage of UDP-encapsulating TCP packets: because NAT gateways typically use shorter timeouts for UDP port mappings than they do for TCP port mappings, long-lived UDP-encapsulated TCP connections will need to send more frequent keepalive packets than native TCP connections. TiU inherits this problem too, although using a single five-tuple for multiple TCP connections alleviates it by reducing the chance of experiencing long periods of silence.

3.1.1.2. Specification

TiU uses a header that is very similar to the header format in [Den08] and [Che13], where it is explained in greater detail. It consists of a UDP header that is followed by a slightly altered TCP header. The UDP source and destination ports are semantically different from [Den08] and [Che13]: TiU uses a single well-known UDP port, and multiple TCP connections use the same UDP port number pair. The encapsulated TCP header is changed to fit into a UDP packet without increasing the MSS; this is achieved by removing the TCP source and destination ports, the Urgent Pointer and the (now unnecessary) TCP checksum. Moreover, the order of fields is changed to move the Data Offset field to the beginning of the UDP payload.

This allows using it to identify other encapsulated content such as a STUN packet: for TCP, the Data Offset must be at least 5, i.e. the most-significant four bits of the first octet of the UDP payload are in the range 0x5-0xF, whereas this is not the case for other protocols (e.g., STUN requires these bits to be 0). The altered TCP header for TiU is shown below:

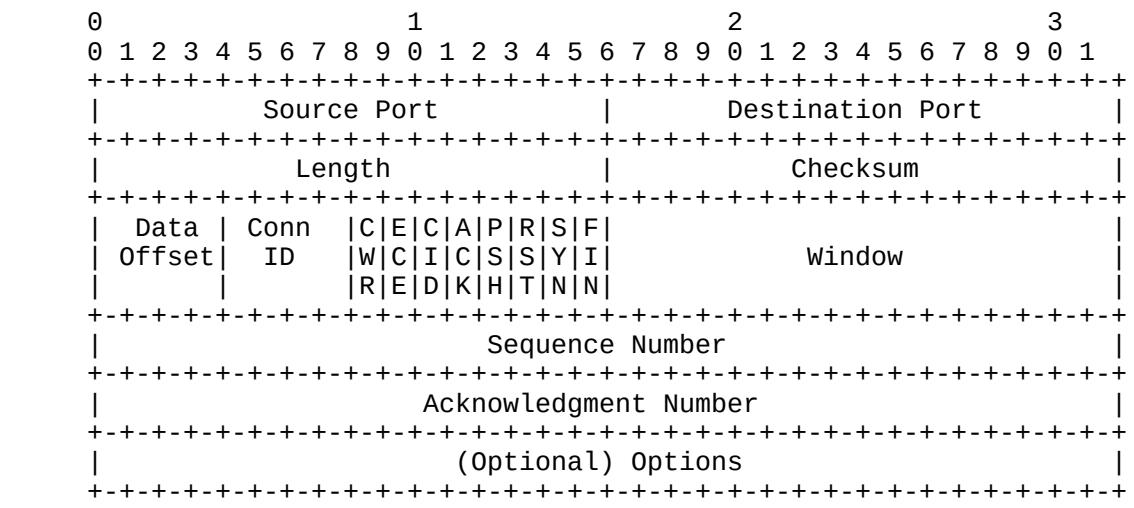


Figure 1: Encapsulated TCP-in-UDP Header Format (the first 8 bytes are the UDP header)

Different from [Den08] and [Che13], the least-significant four bits of the first octet and a bit that replaces the URG bit in the next octet together form a five-bit "Connection ID" (Conn ID). TiU maintains the port numbers of the TCP connections that it encapsulates; the Connection ID is a way to encode the port number information with a few unused header bits. It uniquely identifies a port number pair of a TCP connection that is encapsulated with TiU. Using these five bits, TiU can combine up to 32 TCP connections with one UDP port number pair.

The TiU-TCP SYN and SYN/ACK packets look slightly little different, because they need to establish the mapping between the Connection ID and the port numbers that are used by TiU-encapsulated TCP connections:

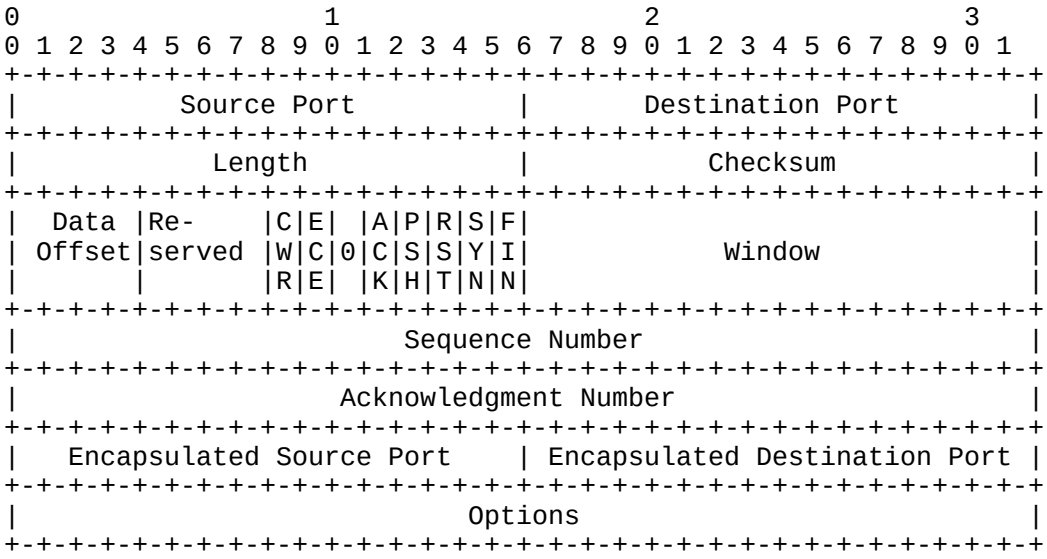


Figure 2: Encapsulated TCP-in-UDP SYN and SYN/ACK Packet Header Format

The Encapsulated Source Port and Encapsulated Destination Port are the port numbers of the TCP connection. To create this header, an implementation can simply swap the position of the original TCP header's port number fields with the position of the Data Offset / Reserved / Flags / Window fields.

Every TiU SYN or TiU SYN-ACK packet also carries at least the TiU-Setup TCP option. This option contains a Connection ID number. On a SYN packet, it is the Connection ID that the sender intends to use in future packets to represent the Encapsulated Source Port and Encapsulated Destination Port. On a SYN/ACK packet, it confirms that such usage is accepted by the recipient of the SYN. A special value of 255 is used to signify an error, upon which TiU will no longer be used (i.e., the next packet is expected to be a non-encapsulated TCP packet). The TiU-Setup TCP option is defined as follows:

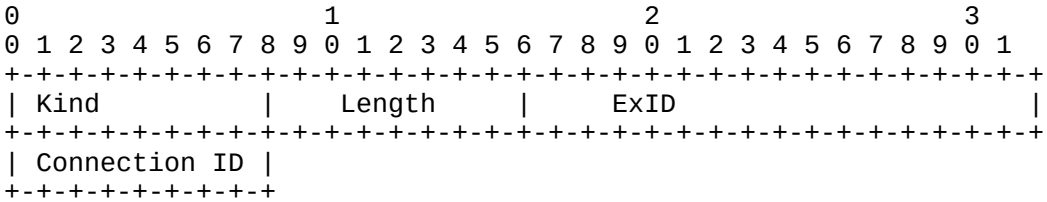


Figure 3: TiU Setup TCP Option

The option follows the format for Experimental TCP Options defined in [RFC6994]. It has Kind=253, Length=5, an ExID that is with value TBD (see Section 6) and the Connection ID. The Connection ID is an 8-bit field for easier parsing, but only values 0-31 are valid Connection IDs (because the Connection ID in non - SYN or SYN/ACK TiU packets is only 5 bit long).

3.1.1.3. Protocol Operation and Implementation Notes

There can be several ways to implement TCP-in-UDP. The following gives an overview of how a TiU implementation can operate. This description matches the implementation described in Section 5.

A goal of TiU is to achieve congestion control coupling with a simple implementation that minimizes changes to existing code. It is thus recommendable to implement TiU in the kernel, as a change to the existing kernel TCP code. The changes fall in two basic categories:

- Encapsulation and decapsulation: this is code that should, in the simplest case, operate just before a TCP segment is transmitted. Based on e.g. a socket option that enables/disables TiU, the TCP segment is changed into the TiU header format (Figure 1). In case it is a TCP SYN or TCP SYN/ACK packet, the header format is defined as in Figure 2, and the TiU-Setup TCP option is appended. This packet is then transmitted. For decapsulation, the reverse mechanism applies, upon reception of a UDP packet that uses destination port XXX (TBD, see Section 6). Both hosts keep a list of encapsulated TCP port numbers and their corresponding Connection IDs. In case a SYN packet requests using a Connection ID that is already reserved, an error (Connection ID value 255 in the TiU Setup TCP option) must be signified to the other end in a TiU-encapsulated TCP SYN/ACK, and encapsulation must be disabled on all further TCP packets. Similarly, when receiving a TiU SYN/ACK with an error, a TCP sender must stop encapsulating TCP packets.

The TCP port number space usage on the host is left unchanged: the original code can reserve TCP ports as it always did. Except for the TiU encapsulation compressing the port numbers into a Connection ID field, TCP ports should be used similar to normal TCP operation. A TCP port that is in use by a TiU-encapsulated TCP connection must therefore not be made available to non-encapsulated TCP connections, and vice versa.

For each TCP connection, two variables must be configured: 1) TiU-ENABLE, which is a boolean, deciding whether to use TiU or not, and 2) Priority, which is a value, e.g. from 1 to 10, that is used by the coupled congestion control algorithm to assign an appropriate share

of the total cwnd to the connection. Priority values are local and their range does not matter for this algorithm: the algorithm works with a flow's priority portion of the sum of all priority values. The configuration of the two per-connection variables can be implemented in various ways, e.g. through an API option.

With these code changes in place, TiU can operate as follows, assuming no previous TiU connections have been made between a specific host pair and a client tries to connect to a server:

- An application uses an API option to request TiU operation. The kernel then sends out a TiU TCP SYN that contains a TiU-Setup TCP option. This packet header contains the encapsulated TCP port numbers (source port A and destination port B) and the Connection ID X.

- The server listens on UDP port XXX (TBD, see Section 6). Upon receiving a packet on this port, it knows that it is a TiU packet and decodes it, handing the resulting TCP packet over to "normal" TCP processing. The TiU-Setup TCP option allows the server to associate future TiU packets containing Connection ID X with ports A and B. The server sends its response as a TiU SYN-ACK.

- TCP operates as normal from here on, but packets are TiU-encapsulated before sending them out and decapsulated upon reception, using Connection ID X. Both hosts associate TiU packets carrying Connection ID X with a local identifier that matches ports A and B, just like they would associate non-encapsulated TCP packets with the same local identifier when seeing ports A and B in the TCP header.

- If an application on either side of the TiU connection wants to connect to a destination host on the other side and requests TiU operation, the kernel sends out another TiU TCP SYN, this time containing a different TCP source port number and either the same or a different destination port number (C and D), and a TiU-Setup TCP option with Connection ID Y. From now on, packets carrying Connection ID Y will be associated with ports C and D on both hosts. Otherwise, TiU operation continues as described above.

- Now, because there are two or more connections available between the same host pair, coupled congestion control begins to operate for all outgoing TiU packets (see Section 2 for details). This is a local operation, applying the priority values that were configured to use for the TiU-encapsulated TCP connections.

Unless it is known that UDP packets with destination port number XXX (TBD, see Section 6) can be used without problems on the path between

two communicating hosts, it is advisable for TiU implementations to contain methods to fall back to non-encapsulated ("raw") TCP communication. Such fall-back must be supported for the case of Connection ID collisions anyway. Middleboxes have been known to track TCP connections [Honda11], and falling back to communication with raw TCP packets without ever using a raw TCP SYN - SYN/ACK handshake may lead to problems with such devices. The following method is recommended to efficiently fall back to raw TCP communication:

- After sending out a TiU SYN packet, additionally send a raw TCP SYN packet.
- After sending out a TiU SYN/ACK packet, additionally send a raw TCP SYN/ACK packet.
- Upon receiving a TiU SYN packet, after responding with a TiU SYN/ACK packet and raw TCP SYN/ACK packet, immediately store the encapsulated port numbers and Connection ID. As long as a TiU connection is ongoing, ignore any additional incoming TCP SYN or TCP SYN/ACK packets from the same host that carry port numbers matching the stored encapsulated port numbers. Otherwise, process TCP SYN or TCP SYN/ACK packets as normal.

This method ensures that the TCP SYN / SYN/ACK handshake is visible to middleboxes and allows to immediately switch back to raw TCP communication in case of failures. If implemented on both sides as described above and no TiU SYN or TiU SYN/ACK packet arrives, yet a TCP SYN or TCP SYN/ACK packet does, this can only mean that the other host does not support TiU, a UDP packet was dropped, or the UDP and TCP packets were reordered in transit. Reordering in the host (e.g., a server responding to a TCP SYN before it responds to a TiU SYN) can be a problem for similar methods (e.g. [RFC6555]), but it can be eliminated by prescribing the processing order as above.

Because TCP does not preserve message boundaries and the size of the TCP header can vary depending on the options that are used, it is also no problem to precede the TCP header in the UDP packet with a different header (e.g. PLUS or SPUD [I-D.hildebrand-spud-prototype]) without exceeding the known MTU limit. When creating a TCP segment, a TCP sender needs to consider the length of this header when calculating the segment size, just like it would consider the length of a TCP option. For this to work, the usage of other headers such as PLUS or SPUD in-between the UDP header and the TiU header must therefore be known to both the sender-side and receiver-side code that processes TiU.

3.1.1.4. Usage Considerations

TiU cannot work with applications that require the Urgent pointer (which is not recommended for use by new applications anyway [RFC6093], but should be consider if TiU is implemented in a way that allows it to be applied onto existing applications; telnet is a well-known example of an application that uses this functionality). It can also be used as a method to experimentally test new TCP functionality in the presence of middleboxes that would otherwise create problems (as some have been known to do [Honda11]).

Reasons to use TiU include the benefits of [Che13] and [Den08] that were discussed in Section 1. TiU has the disadvantage of disabling ECMP for the TCP connections that it encapsulates. This can reduce the capacity usage of these TCP connections. It has the advantage of being able to apply TCP-CCC coupled congestion control, which can provide precise congestion window assignment based on a priority.

3.1.2. Other Methods

There are many possible encapsulation schemes for various use cases. For example, Generic UDP Encapsulation (GUE) [I-D.draft-ietf-nvo3-gue] allows us to multiplex several TCP connections onto a same UDP port number pair. Several encapsulation methods transmit layer-2 frames over an IP network - e.g. VXLAN [RFC7348] (over UDP/IP) and NvGRE [RFC7637] (over GRE/IP). Because Layer-2 networks should be agnostic to the transport connections running over them, the path should not depend on the TCP port number pair and our algorithm should work. Some care must still be taken: for example, for NvGRE, [RFC7637] says: "If ECMP is used, it is RECOMMENDED that the ECMP hash is calculated either using the outer IP frame fields and entire Key field (32 bits) or the inner IP and transport frame fields". If routers do use the inner transport frame fields (typically, port numbers) for this hashing, we have the same problem even over NvGRE.

4. Related Work

The TCPMUX mechanism in [RFC1078] multiplexes TCP connections under the same outer transport port number; it does however not preserve the port numbers of the original TCP connections, and no method to couple congestion controls is described in [RFC1078].

Congestion control coupling follows the style of RTP application congestion control coupling in [I-D.ietf-rmcat-coupled-cc] which is designed to be easy to implement, and to minimize the number of changes that need to be made to the underlying congestion control mechanisms. This method was shown to yield several benefits in

[fse]. TCP-CCC requires slightly deeper changes to TCP's congestion control, making it harder to implement than [I-D.ietf-rmcat-coupled-cc], but it is still a much smaller code change than the Congestion Manager [RFC3124].

Combining congestion controls as TCP-CCC does it has some similarities with Ensemble Sharing in [RFC2140], which however only concerns initial values of variables used by new connections and does not share the congestion window (cwnd). The cwnd variable is shared across ongoing connections in [ETCP] and [EFCM], and the mechanism described in Section 2 resembles the mechanisms in these works, but neither [ETCP] nor [EFCM] address the problem of ECMP.

Coupled congestion control has also been specified for Multipath TCP [RFC6356]. MPTCP's coupled congestion control combines the congestion controls of subflows that may traverse different paths, whereas we propose congestion control coupling for flows sharing a single-path. TCP-CCC builds on the assumption that all its encapsulated TCP connections traverse the same path. This makes the two methods for coupled congestion control very different, even though they both aim at emulating the behavior of a single TCP connection in the case where all flows traverse the same network bottleneck. For example, a new flow obtaining a a larger-than-IW share of the aggregate cwnd would be inappropriate for an MPTCP subflow.

5. Implementation Status

We have implemented TCP-CCC and TiU encapsulation for both the sender and receiver in the FreeBSD kernel, as a simple add-on to the TCP implementation that is controlled via a socket option.

6. IANA Considerations

This document specifies a new TCP option that uses the shared experimental options format [RFC6994]. No value has yet been assigned for ExID.

This document requires a well-known UDP port (referred to as port XXX in this document). Due to the highly experimental nature of TiU, this document is being shared with the community to solicit comments before requesting such a port number.

7. Security Considerations

TBD

8. Acknowledgements

This work has received funding from Huawei Technologies Co., Ltd., and the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644334 (NEAT). The views expressed are solely those of the author(s).

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

9.2. Informative References

[anrw2016] Islam, S. and M. Welzl, "Start Me Up:Determining and Sharing TCP's Initial Congestion Window", ACM, IRTF, ISOC Applied Networking Research Workshop 2016 (ANRW 2016) , 2016.

[Che13] Cheshire, S., Graessley, J., and R. McGuire, "Encapsulation of TCP and other Transport Protocols over UDP", Internet-draft draft-cheshire-tcp-over-udp-00, June 2013.

[Den08] Denis-Courmont, R., "UDP-Encapsulated Transport Protocols", Internet-draft draft-denis-udp-transport-00, July 2008.

[EFCM] Savoric, M., Karl, H., Schlager, M., Poschwatta, T., and A. Wolisz, "Analysis and performance evaluation of the EFCM common congestion controller for TCP connections", Computer Networks (2005) , 2005.

[ETCP] Eggert, L., Heidemann, J., and J. Joe, "Effects of ensemble-TCP", ACM SIGCOMM Computer Communication Review (2000) , 2000.

[fse] Islam, S., Welzl, M., Gjessing, S., and N. Khademi, "Coupled Congestion Control for RTP Media", ACM SIGCOMM Capacity Sharing Workshop (CSWS 2014) and ACM SIGCOMM CCR 44(4) 2014; extended version available as a technical report from http://safiquli.at.ifi.uio.no/paper/fse-tech-report.pdf , 2014.

[Honda11] Honda, M., Nishida, Y., Raiciu, C., Greenhalgh, A., Handley, M., and H. Tokuda, "Is it still possible to extend TCP?", Proc. of ACM Internet Measurement Conference (IMC) '11, November 2011.

[I-D.draft-ietf-nvo3-gue] Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", Internet-draft draft-ietf-nvo3-gue-05, October 2016.

[I-D.hildebrand-spud-prototype] Hildebrand, J. and B. Trammell, "Substrate Protocol for User Datagrams (SPUD) Prototype", draft-hildebrand-spud-prototype-03 (work in progress), March 2015.

[I-D.ietf-rmcat-coupled-cc] Islam, S., Welzl, M., and S. Gjessing, "Coupled congestion control for RTP media", draft-ietf-rmcat-coupled-cc-03 (work in progress), July 2016.

[I-D.ietf-rmcat-sbd] Hayes, D., Ferlin, S., Welzl, M., and K. Hiorth, "Shared Bottleneck Detection for Coupled Congestion Control for RTP Media.", draft-ietf-rmcat-sbd-04 (work in progress), March 2016.

[RFC1078] Lottor, M., "TCP port service Multiplexer (TCPMUX)", RFC 1078, DOI 10.17487/RFC1078, November 1988, <http://www.rfc-editor.org/info/rfc1078>.

[RFC2140] Touch, J., "TCP Control Block Interdependence", RFC 2140, DOI 10.17487/RFC2140, April 1997, <http://www.rfc-editor.org/info/rfc2140>.

[RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, DOI 10.17487/RFC3124, June 2001, <http://www.rfc-editor.org/info/rfc3124>.

[RFC5128] Srisuresh, P., Ford, B., and D. Kegel, "State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)", RFC 5128, DOI 10.17487/RFC5128, March 2008, <http://www.rfc-editor.org/info/rfc5128>.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <http://www.rfc-editor.org/info/rfc5245>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <http://www.rfc-editor.org/info/rfc5389>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

[RFC6093] Gont, F. and A. Yourtchenko, "On the Implementation of the TCP Urgent Mechanism", RFC 6093, DOI 10.17487/RFC6093, January 2011, <http://www.rfc-editor.org/info/rfc6093>.

[RFC6356] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, DOI 10.17487/RFC6356, October 2011, <http://www.rfc-editor.org/info/rfc6356>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <http://www.rfc-editor.org/info/rfc6437>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

[RFC6994] Touch, J., "Shared Use of Experimental TCP Options", RFC 6994, DOI 10.17487/RFC6994, August 2013, <http://www.rfc-editor.org/info/rfc6994>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <http://www.rfc-editor.org/info/rfc7348>.

[RFC7424] Krishnan, R., Yong, L., Ghanwani, A., So, N., and B. Khasnabish, "Mechanisms for Optimizing Link Aggregation Group (LAG) and Equal-Cost Multipath (ECMP) Component Link Utilization in Networks", RFC 7424, DOI 10.17487/RFC7424, January 2015, <http://www.rfc-editor.org/info/rfc7424>.

[RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <http://www.rfc-editor.org/info/rfc7637>.

Authors' Addresses

Michael Welzl
University of Oslo
PO Box 1080 Blindern
Oslo N-0316
Norway

Email: michawe@ifi.uio.no

Safiqul Islam
University of Oslo
PO Box 1080 Blindern
Oslo N-0316
Norway

Phone: +47 22 84 08 37
Email: safiquli@ifi.uio.no

Kristian Hiorth
University of Oslo
PO Box 1080 Blindern
Oslo N-0316
Norway

Email: kristahi@ifi.uio.no

Jianjie You
Huawei
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: youjianjie@huawei.com

INTERNET-DRAFT
Intended Status: Standard Track
Expires: May 4, 2017

Y.Yu
HUAWEI Technologies
October 31, 2016

Layer 3 Quantized Congestion Notification(L3QCN)in the Converged Network
draft-yu-tsvwg-l3qcn-00

Abstract

The more demands for the lossless and low latency network in the modern datacenter appear because the proliferation of demanding applications. Some congestion control schemes such as CN, PFC, ETS which is introduced by IEEE 802.1 focus on the L2 network domain. While current TCP/IP stacks can’t meet these requirement on L3 or above networks. This draft introduces the L3QCN(Layer 3 Quantized Congestion Notification), an end to end congestion control scheme which adopt QCN and DCQCN on L2 network. It specifies protocols, procedures, and managed objects to support congestion control on the datacenter network.

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). 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 May 4, 2017.

Copyright and License Notice

Table of Contents

1 Introduction

Currently, there are 3 classes of streams in the DC network:
1)Storage Traffic (Lossless)
2)High Compute Traffic(Low latency)
3)Ethernet Traffic (Certain packet loss& latency tolerance)
Traditional DC network treat different traffic with different network bearer which exist in the small scale DC. While with the expand of the DC scale, there is an available method which use the Ethernet to bear the streams by applying the congestion control method. IEEE has introduced the following specifications:

1. Enhanced Transmission Selection (ETS) [1] When the offered load in a traffic class doesn’t use its allocated bandwidth, enhanced transmission selection will allow other traffic classes to use the available bandwidth. This avoid the burst of one class traffic to influence other classes which provide the minimum guaranteed bandwidth to all traffic classes. This also facilitate the multiple classes exist in one network.

2.Priority-based Flow Control(PFC) [2] Data Center Bridging networks (bridges and end nodes) are characterized by limited bandwidth-delay product and limited hop-count. Traffic class is identified by the VLAN tag priority values. Priority-based flow control is intended to eliminate frame loss due to congestion. This realized the lossless of storage stream and no impact to other 2 traffic classes when all the 3 traffic classes coexist in the Ethernet.

3.Quantized Congestion Notification (QCN) [3] This mechanism enable bridges to signal congestion information to end stations capable of transmission rate limiting to avoid frame loss. Resolve the latency increase caused by flow control or packet retransmission to achieve the higher network throughput.

This draft introduce a L3QCN method to resolve the congestion problem under the converged network in the datacenter. Different classes of traffic will be configured with corresponding priorities. Bridge will apply the policies of congestion control according to the traffic of congested traffic which is defined by the priority.

1.1 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].

2 Current Congestion Control method

2.1 QCN Introduction

2.1.1 QCN Technical Solution

QCN is defined in IEEE 802.1Qau, there are 2 types of Ethernet frame: One data frame with CN-TAG as Figure 1 shown. The Converged Network Adapters (CNA) which support QCN function will send out the CN-TAG frame when connecting network domain. The difference from the normal frame is the CN-TAG field in the head of Ethernet frame which includes RPID (also known as FLOW-ID). RPID will uniquely identifies every stream sent by the adaptor. When the congestion appeared, bridge will send out CNM frame(introduced in second clause) to notify the source node to stop sending this stream. The FLOW-ID of source frame will be encapsulated in the CNM frame. When the adaptor receives the CNM frame, it will reduce the transmission rate of the identified flow in order to control the specific traffic precisely.

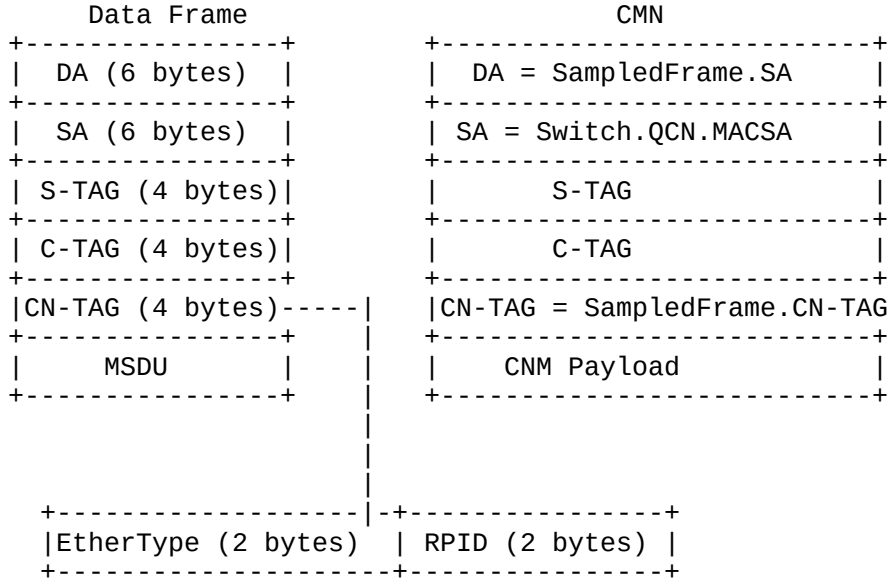


Figure 1. QCN data frame and CMN

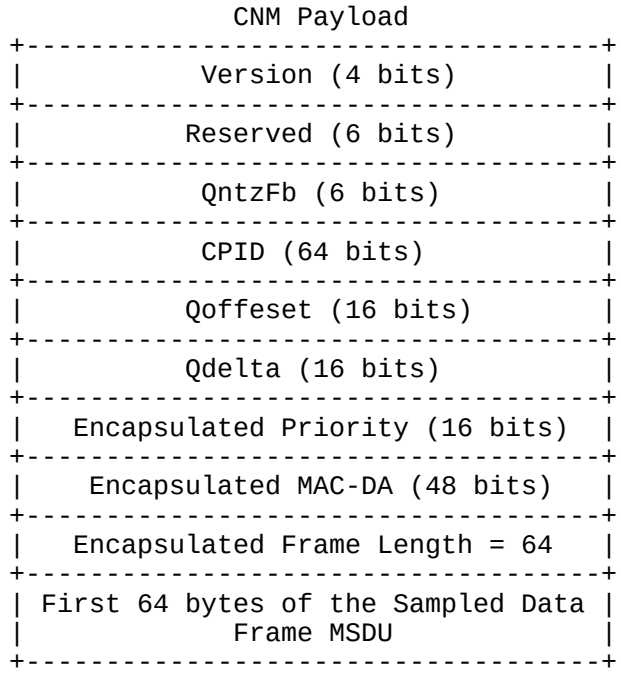


Figure 2. CMN payload

The CNM frame is shown as Figure 2:

Field 1: Version of CNM message (4 bits)
Field 2: Reserved (6 bits)
Field 3: QntzFB, Quantized feedback of CNM message (6 bits)
Field 4: Congestion Point Identifier (CPID, 8 bytes). In order to assure the uniqueness of the identifier, use the MAC address as the upper 6 bytes. Lower 2 bytes identify the different ports or different priority classes in the same device.
Field 5: QOffset (2 bytes). Current number of available bytes in the sending queue of the congested point (CP)
Field 6: QDelta (2 bytes), the difference of available bytes of CP at 2 time point.
Field 7: Encapsulated priority (2 bytes). Use upper 3 bits of the 1st byte to fill the priority of the CNM frame. Else is 0.
Field 8: Encapsulated destination MAC address (6 bytes). Fill the destination MAC address which trigger the CNM frame.
Field 9: Encapsulated MSDU length (2 bytes). The length of the Encapsulated MSDU.
Field 10: Encapsulated MSDU (64 bytes). Fill in the payload of the CNM.

2.1.2 The limitation of QCN

During the congestion, bridge need to encapsulate the FLOW-ID(in the head of Ethernet frame) in the CNM. Then replace the destination MAC address of CNM with the source MAC address of the congested frame in order to ensure CNM could be send back the sending server. Sending server reduce the flow according to the FLOW-ID carried in the CNM. This characteristic limit QCN only in Ethernet(Level 2 in ISO). Since the head of Ethernet frame will be changed during every packet routing in the IP network, the FLOW-ID and MAC address of sending server will be lost. So the downstream bridge could not create the CNM and send back to the sending server. QCN couldn't support the Layer 3 networking.

2.2 Introduction of DCQCN
2.2.1 DCQCN technical solution

DCQCN[4] is a kind of congestion control solution proposed by Microsoft for the DC network domain. DCQCN is mainly deployed in the RoCEv2 scene. CP (Congestion Point, bridge) set the CN(congestion notification) for the datagram with probability according to the degree of the congestion. After the datagram sent to NP(Notification Point, receiving server) , NP construct CNP (Congestion Notification Packet) to RP(Reaction Point, sending server). RP reduce or increase the transmission rate according to the dedicated algorithm which is similar to QCN.

2.2.2 The limitation of DCQCN

DCN construct ECN(explicit congestion notification)[5] tag during the congestion and forward to NP. NP construct CNP to notify RP. The reaction is not quite timely(Control Loop Delay is big). If the congestion appeared on the upper jump, for example on the TOR, there is more delay of 9 jumps than the direct response.

3. Layer3 QCN
3.1 L3QCN Introduction

L3QCN is a technical solution to resolve the congestion problem under the converged network in the datacenter. Different class of traffic will be configured with corresponding priorities. Bridge will deploy the policies of congestion control according to the class of congested traffic which is defined by the priority.

3.2 Use case of L3QCN
3.2.1 A hybrid method with QCN
Deploy priority 5 to the traffic sent out by QCN server. When the queue buffer for the priority 5 exceed the defined threshold, the bridge will back-haul the congestion information to the accessing TOR. TOR and HOST can reach to each other on the Ethernet which is similar to a L2 domain. In this situation, standard QCN is performed. Accessing TOR transform the congestion information to standard CNM frame and send to QCN server

which realize the congestion control.

Deploy priority 7 to the traffic sent out by RoCEv2 server. When the queue buffer for the priority 7 exceed the defined threshold, CP judges the key flow causing the congestion. Then CP construct the standard CNP. The RoCEv2 server reduce the transform rate according to the probability of CNP reception.

3.2.2 L3QCN in CLOS fat-tree

L3QCN control steps are as follows:

1)Datagram sent out from QCN server enters the accessing TOR. Firstly, accessing TOR will save the source MAC address, FLOW-ID, VLAN-TAG and IP 5-tuple to the local table, shown in Table 1. Then TOR perform the normal routing.

Src IP	Dst IP	Src Port	Dst Port	Proto -col	MAC SA	Flow ID	VLAN TAG
192.168.2.100	192.168.3.30	5678	21	6	0x01a4f5aefe	0xa878	100
10.1.10.2	10.2.20.1	8957	21	6	0xfd16783acd	0xc9a0	1024
192.168.2.100	10.3.50.1	2345	80	6	0x0a25364101	0x0ac9	3
200.1.2.3	100.2.3.4	2567	47	17	0xed16d8ea0a	0x37a0	90

Table 1. FLOW-ID Mapping Table

2)Shown in Figure 3. Congestion caused by Incast flow, T4 detect the congestion in a certain queue and exceed the threshold. Distinguish the flow model according to the priority of the queue.

Please view in a fixed-width font such as Courier.

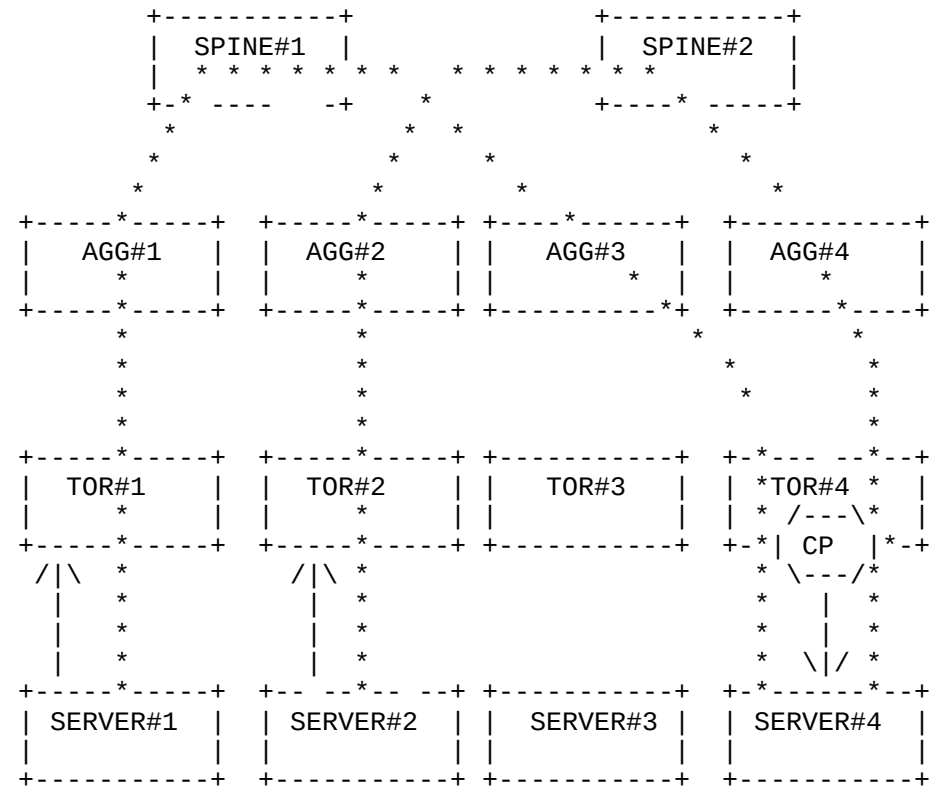


Figure 3. Incast flow model

3)If it is the flow from QCN server, conduct self-defined CNM which include the 5-tuple, congestion indications (defined in QCN specification, such as QntzFb, CPID, Qoffset,QDelta), encapsulate IP +UDP. UDP need to use a specific port No. which is used to recognize the QCN frame in TOR. Or use a bit in the IP head(reserved bit) to indicate the type of the frame. The dedicate IP is set to the source IP which assure the CNM could be routed to the accessing TOR. It’s better to construct the self-defined CNM based on the standard CNM to reduce the writing times which might increase the performance.

4)As shown in the Figure 4&5 , T2 recognize the self-defined CNM according to the destination UDP port. T2 map the self-defined CNM to the standard CNM and send to H2. The QCN is performed in L2 domain because the adaptor of H2 support the standard QCN.

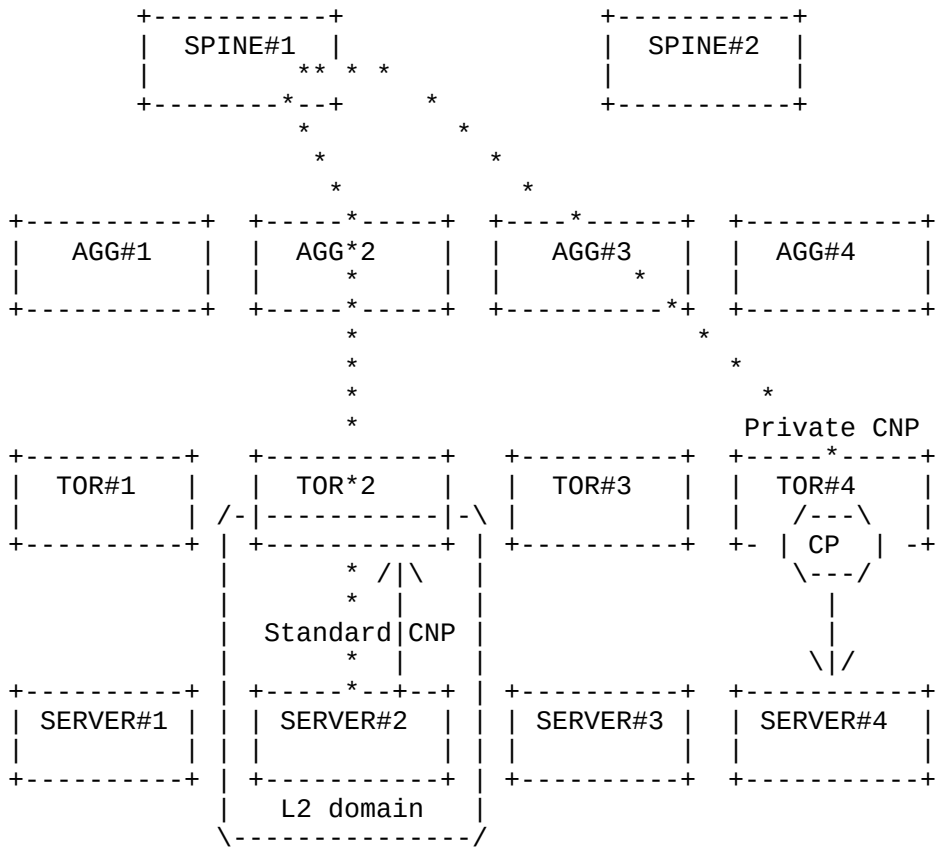
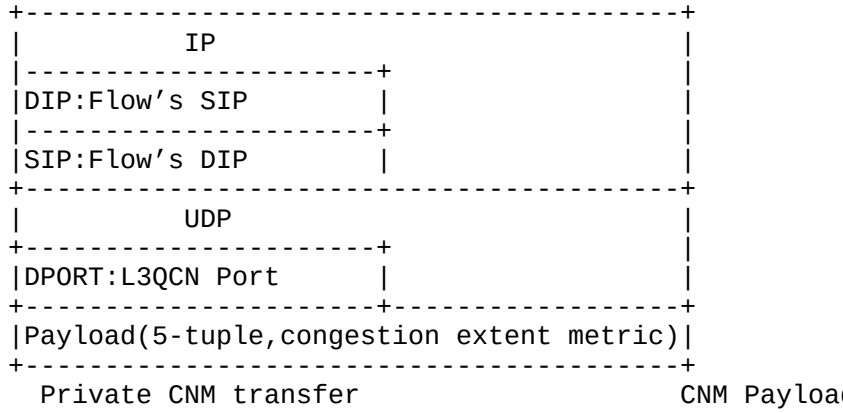


Figure 4. Construct the self-defined CNM

Please view in a fixed-width font such as Courier.



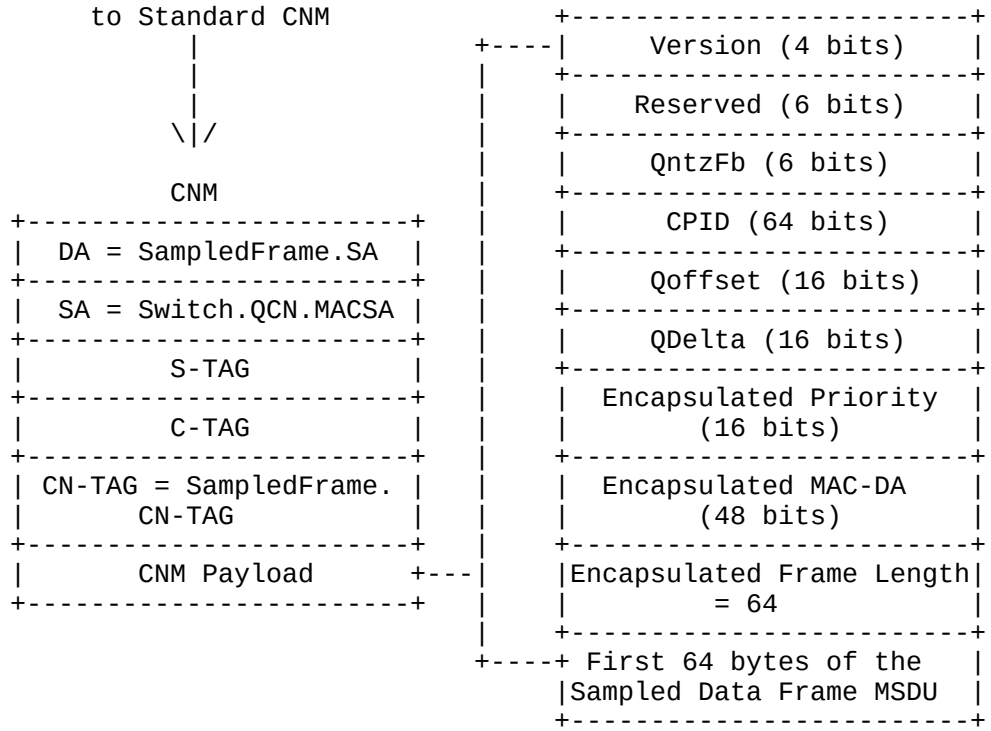
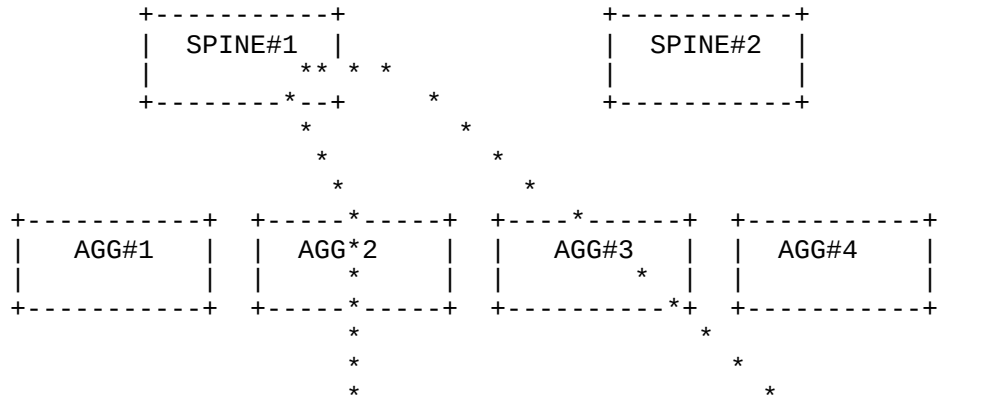


Figure 5. Transfer Private CNM to Standard CNM

5)As shown in the Figure 6 , T4 recognize which flow causes the congestion. CP construct the standard CNP. The adaptor of RoCEv2 server support CNP and reduce the transmission rate according to the probability of CNP reception.

Please view in a fixed-width font such as Courier.



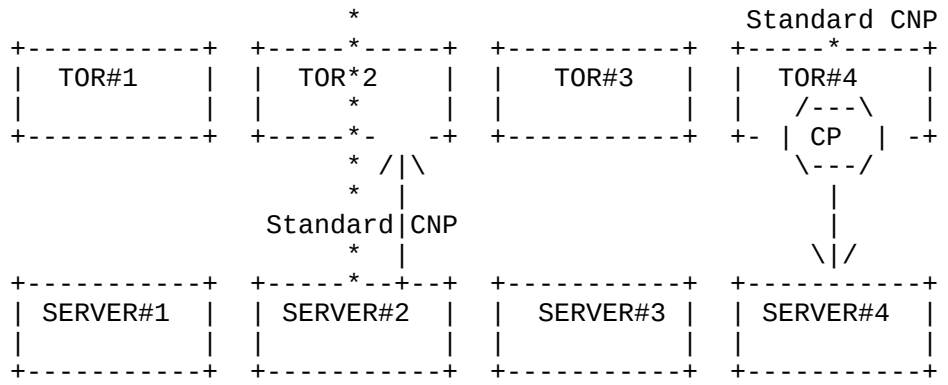


Figure 6. CP construct the standard CNP based on RoCEv2

4. Conclusion

L3QCN resolve the problem that QCN could not support L3 network. L3QCN realize the QCN mechanism across the L3 network. There is no modification on the QCN servers.
For the RoCEv2 traffic, since the CP send the CNP when reach the congestion threshold, it reduce the Control Loop Delay dramatically which could reduce the depth of the queue buffer and the datagram delay. The performance of the network is improved.

5 Security Considerations

N/A

6 IANA Considerations

Will apply the specific UDP port No. if required.

7 References

7.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

7.2 Informative References

[1] IEEE 802.1: 802.1Qaz Draft 2.5- Enhanced Transmission Selection

[2] IEEE 802.1: 802.1Qbb Draft 2.3- Priority-based Flow Control

[3] IEEE 802.1: 802.1Qau Draft 2.4- Congestion Notification

[4] Yibo Zhu et al., SIGCOMM 2015, Congestion Control for Large-Scale RDMA Deployments

[5] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN). RFC 3168

Authors' Addresses

Yolanda Yu
101 SOFTWARE AV., YUHUATAI DIST., NANJING,
JIANGSU,210012,CHINA
EMail: yolanda.yu@huawei.com