

INTERNET-DRAFT  
Intended Status: Standard Track  
Expires: April 30, 2015

M. Ghobadi  
Microsoft Research  
H. Song  
R. Huang  
Huawei  
Y. Ganjali  
University of Toronto  
October 27, 2014

TCP Parameter Dynamic Control  
draft-song-dclc-tcpdc-04

Abstract

Congestion control has been extensively studied for many years. Today, the Transmission Control Protocol (TCP) is used in a wide range of networks (LAN, WAN, data center, campus network, enterprise network, etc.) as the de facto congestion control mechanism. Despite its common usage, TCP operates in these networks with little knowledge of the underlying network or traffic characteristics. As a result, it is deemed to continuously increase or decrease its congestion window size in order to handle changes in the network or traffic conditions. Thus, TCP frequently overshoots or undershoots the ideal rate making it a "Jack of all trades, master of none" congestion control protocol. In light of the emerging popularity of centrally controlled networks such as Software-Defined Networks (SDNs), we propose a framework that takes advantage of the information available at the central controller to improve TCP. Specifically, in this document, we propose OpenTCP as a dynamic and programmable TCP adaptation framework for centrally controlled networks. OpenTCP gathers global information about the status of the network and traffic conditions through the centralized controller, and uses this information to adapt TCP. OpenTCP periodically sends updates to end-hosts which, in turn, update their behaviour using a simple kernel module.

This document describes a framework and message flows for centralized congestion control parameter adaptation based on congestion control policies and network status measurements, so that each end host in a network can make better use of the network resource according to the available resources. In the rest of this document we use TCP as a standard congestion control mechanism, but the same idea can be applied to other congestion control protocols as well. A TCP Optimization Element and a TCP Optimization Agent are introduced. The message patterns include request response and subscription/notification. This mechanism can be used in network

service providers' networks, as well as in data center networks.

#### Status of this Memo

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

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

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

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

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

#### Copyright and License Notice

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

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

#### Table of Contents

1	Introduction . . . . .	4
2	Conventions Used in This Document . . . . .	6
3	TCP Parameter Control Architecture . . . . .	7
3.1	Guidance Level . . . . .	8

3.2	Subscription Mode . . . . .	8
3.3	Request/Response Mode . . . . .	8
4	Messages . . . . .	8
4.1	Explicit RR . . . . .	9
4.1.1	TcpParReq . . . . .	9
4.1.2	TcpParRes . . . . .	9
4.2	Subscription/Notification . . . . .	10
4.2.1	TcpParSub . . . . .	10
4.2.2	Notification . . . . .	11
4.3	Error Message . . . . .	11
5	Security Considerations . . . . .	12
5	Fairness and Stability . . . . .	12
7	IANA Considerations . . . . .	13
8	References . . . . .	13
8.1	Normative References . . . . .	13
	Authors' Addresses . . . . .	13

## 1 Introduction

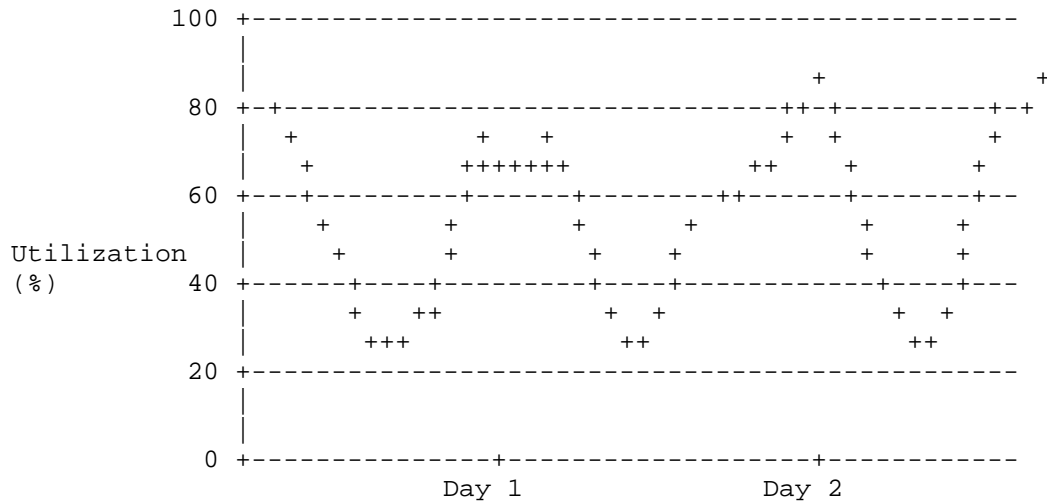


Figure 1 Link Utilization Rate during A Day

The Transmission Control Protocol (TCP) is used in a wide range of networks as the congestion control mechanism. Measurements reveal that 99.91% of traffic in Microsoft data centers is TCP, 10% of the aggregate North America Internet traffic is YouTube over TCP, and measurements from 10 major data centers including university, enterprise, and cloud data centers show TCP as the dominant congestion control protocol. TCP is a mature protocol and has been extensively studied over a number of years. Hence, network operators trust TCP as their congestion control mechanism to maximize the bandwidth utilization of their network while keeping the network stable.

Despite, and because of, its common usage, TCP operates in these networks with little knowledge of the underlying network or traffic characteristics. However, limiting TCP to a specific network and taking advantage of the local characteristics of that network can lead to major performance gains. For instance, DCTCP out-performs TCP in data center networks, even though the results might not be applicable in the Internet. With this mindset, one can adjust TCP (the protocol itself and its parameters) to gain better performance in specific networks (e.g. data centers). Moreover, even focusing on a particular network, the effect of dynamic congestion control adaptation to traffic patterns is not well understood in today's

networks. Such adaptation can potentially lead to major improvements, as it provides another dimension that today's TCP does not explore.

Figure 1 depicts aggregate link utilization of a core link in a backbone service provider in North America[Hotnets]. We can see that the link utilization is low for a significant period (below 50% for 6-8 hours). A pattern is seen on all the links in this network. In fact, the presented link has the highest utilization and is considered to be the bottleneck in this network. If the network operator aims at minimizing flow completion times in this network, it makes sense to increase TCP's initial congestion window size (`init_cwnd`) when the network is not highly utilized (we focus on internal traffic in this example). Ideally, the exact value of `init_cwnd` should be a function of the network-wide state (here, the number of flow initiations in the system) and how aggressively the operator wants the system to behave (congestion control policy). The operator can define a policy like the following: if link utilization is below 50%, `init_cwnd` should be increased to 20 segments instead of the default value of four segments. In other words, given the appropriate mechanisms the operator could choose the right value for the initial congestion window.

The forwarding capacity of the network is evolving very fast nowadays. When the TCP was designed, the routers and switches have low capacity, and the network was easy to be congested. So it was designed with a very small initial congestion window. But small initial congestion window size means more cycles during the slow start period. So for Linux 3.0, Google proposed to increase the `init_cwnd`. For example, when  $1095 < \text{MSS} \leq 2190$ , the original `init_cwnd` = 3, but in Linux 3.0, Google proposes to increase it to 10. However, that's still a fixed number without considerations of the network variations. In some areas of the world, the network condition is much better than that of other areas. That `init_cwnd` size should be even bigger to provide better performance for applications inside that area (when both sender and receiver are inside that area).

Currently, network operators use various ad-hoc solutions, as temporary adjustments of TCP to fit their network and traffic. These manual tweaks open the way for misconstruction, make debugging and troubleshooting difficult, and can result in substantial operational overhead. Moreover, making any changes to the underlying assumptions about the network or traffic requires rethinking the impact of various parameters and can result in ongoing efforts to manually adjust TCP because any proposed change should work under all conditions. Having a system that measures the state and dynamics of the network and adapts TCP's behaviour accordingly can address these problems.

This document addresses the need for a systematic way of adapting TCP to network and traffic conditions. We propose OpenTCP as a framework for dynamic adaptation of TCP based on network and traffic conditions in centrally controlled networks. Figure 2 provides a schematic view of how OpenTCP works. OpenTCP collects data on the underlying network state (e.g. topology and routing information) as well as statistics about network traffic (e.g. link utilization and traffic matrix). Then, using this aggregated information and based on congestion control policies defined by the network operator, OpenTCP determines a specific set of adaptations for TCP.

At a high level, congestion control policies define which statistics need to be collected, which high level performance metrics the operator would like to optimize (e.g. minimize drops, maximize utilization, or minimize flow completion times), and what the constraints of the system are. OpenTCP periodically sends Congestion Update Epistles or CUEs to the end-hosts which, in turn, update their behaviour using a simple kernel module that can adapt TCP.

Consider the following simple example. Imagine a network where all links have very low utilization (say below 50%) at all times. If the network operator aims at minimizing flow completion times in this network, it makes sense to increase the TCP initial congestion window size, as suggested by Dukkipati et al. The exact value of the initial congestion window will be a function of the number of flow initiations in the system (network state), and how aggressively the operator wants the system to behave (congestion control policy). For a network where dropping a few packets is not a major problem, the operator can define a policy like the following: if all link utilizations are below 50%, the initial congestion window size can be increased to 20 segments instead of the default value of four. If the operator is more conservative, the window size can be set to a smaller value (e.g. 5 segments), improving flow completion times with smaller risk of causing packet drops. The operator can even leave it to OpenTCP to dynamically choose the right value for the initial congestion window size.

It is also possible to change the TCP timeout behaviors according to the network status. When the timeout happens during the period that relative network link utilization is under 50% (the cwnd size does not exceed the peak buffer size, and the rate does not exceed the subscription rate), the cwnd can be remained the same, without reducing it tremendously, if the sending rate does not exceed the subscription rate (upload rate of the sender and download rate of the receiver) nor overflow the receiver's receiving window.

## 2 Conventions Used in This Document

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 [KEYWORDS]. This document also uses the following conventions.

TOE: TCP Optimization Element, which accesses the network statistical information from network measurement entities, such as an OAM server, NMS, or a LMAP server and etc, and provides the TCP optimization service to the TCP Optimization Agent (TOA).

TOA: TCP Optimization Agent, which is deployed in the end host, and adjust the TCP stack behavior according to the guidance from the TOE. Note that one TOA can serve multiple applications.

### 3 TCP Parameter Control Architecture

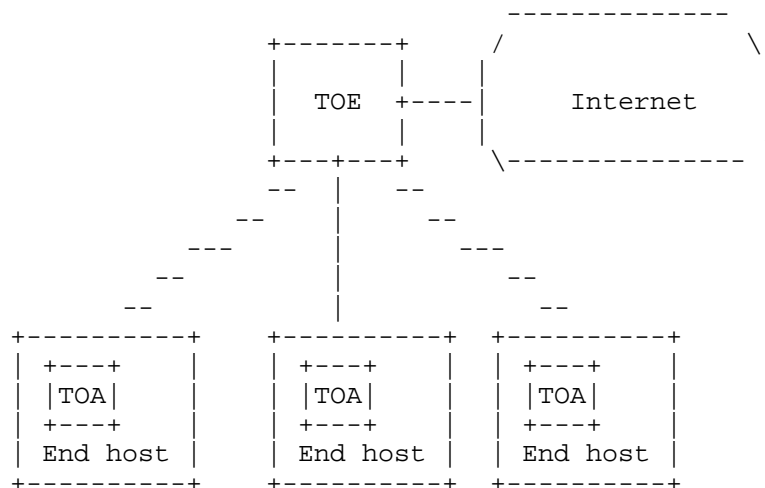


Figure 2 OpenTCP Architecture

It is assumed that there is existing method for the TOE to get the routing information and network status for each link in a network, for example, from a PCE server. Then the TOE knows the possible path for each communication, and it also knows about the link utilization rate, lost ratio, and the statistics information of the link and the network. The TOE contemplates the network utilization rate at different time during a day, and sets the TCP optimization parameters accordingly. For example, from the midnight to early morning, the network utilization is very low, end hosts can use larger init\_cwnd,

size and the window size degradation behavior can be much slower during time-out or receiving the same ACK event.

### 3.1 Guidance Level

There are different types of guidance from the TOE according to different network levels.

The normal type would be the TCP optimization parameter for the whole administrative network domain. When source end host and the destination end host are inside the same administrative network domain, they are suggested to use the parameters provided by the TOE to optimize the TCP transport. The domain can be an intra DC network, a LAN network or a NSP network.

Another type is TCP optimization parameter for a particular link, for example, TOE provides optimization parameters to end hosts in two data centers which share an inter-DC dedicated link. When the link is congested, the TOE suggests the end hosts to use smaller `init_cwnd` size and reduce the congestion window sharply during time-out or replicated ACKs. This type of service is only available when the source end host and the destination end host are deployed at two ends of a particular link.

When either one of the communication endpoints is out of the scope of the administrative boundaries, the recommendation TCP optimization parameters MUST NOT be used.

### 3.2 Subscription Mode

TOA can use subscription mode to communicate with the TOE to get updated TCP optimization parameters. This is very useful for long-lived traffic, as well as for end hosts which have frequent TCP connections. The guidance level can be either the network level or the link level.

### 3.3 Request/Response Mode

TOA can also use the request response mode to communicate with the TOE. With each TCP optimization request, the TOA lists the two communication end hosts IP address, and indicate the level of guidance. Then TOE gives the response of the current recommendation parameters for TCP transport.

## 4 Messages

A TOA uses the HTTP protocol with an HTTP POST entity body of JSON



Objects, to request the TCP parameter guidance from a TOE server.

#### 4.1 Explicit RR

Explicit request and response mode is mainly used for the guidance of TCP parameters between two endpoints. If the path between two endpoints is a dedicated link, it is easier to give the guidance with considering the two endpoint properties and the link utilization status. When the path between two endpoints is within the administrative domain of the TOE, but subject to change (for example, the route may be changed through routers), then the TOE should give conservative guidance parameters.

##### 4.1.1 TcpParReq

```
object {  
  TypedEndpointAddr: source;  
  TypedEndpointAddr: destination;  
}TcpParReq;
```

Typed Endpoint Address: Typed Endpoint Addresses are encoded as strings of the format 'AddressType:EndpointAddr', with the ':' character as a separator. The type 'TypedEndpointAddr' is used to indicate a string of this format. This document defines two values for AddressType: 'ipv4' to refer to IPv4 addresses, and 'ipv6' to refer to IPv6 addresses. EndpointAddr component of TypedEndPointAddr is also encoded as a string. The exact characters and format depend on AddressType. This document defines EndpointAddr when AddressType is 'ipv4' or 'ipv6'. IPv4 Endpoint Addresses are encoded as specified by the 'IPv4address' rule in Section 3.2.2 of [RFC3986]. IPv6 Endpoint Addresses are encoded as specified in Section 4 of [RFC5952].

Upon receive this request, TOE should lookup the subscription rate, i.e. uplink rate quota of the source and the downlink rate quota of the destination, and then examine the current link utilization rate, then gives the appropriate TCP parameter guidance.

The media type for explicit request is "application/opentcp-rr+json".

##### 4.1.2 TcpParRes

```
object {  
  TcpPar: parameters<0...*>;  
}TcpParRes;  
  
object {  
  ParType -> ParValue;
```

```
}TcpPar;
```

ParType: A JSONString defined the TCP parameter type, this document defines the "initcwnd", "threshold", "timeOut", and "repeatedtimeouts". (It is open for discussion).

ParValue: A JSONValue defined the value for the relative parameter type.

The media type for explicit response is "application/opentcp-rrparameters+json".

## 4.2 Subscription/Notification

This method is mainly used for getting the guidance for the TCP parameters in the administrative domain, but can also be used for long-lived traffic flows. In the response, it has indications on when to change the TCP parameters.

### 4.2.1 TcpParSub

```
object {  
  JSONString: subscription_id;  
  JSONValue: request_type;  
  [TypedEndpointAddr: source;]  
  [TypedEndpointAddr: destination;]  
  GuidanceLevel: level;  
}TcpParSub
```

subscription\_id: a JSONString generated by the TOE to uniquely identify a subscription. If it is the first time for this TOA to send this particular subscription to the TOE, the subscription\_id must be "null". After the TOA gets the subscription\_id from the TOE, it has to insert the id for each following subscription message for the same link or network guidance information.

request\_type: this document defines the type "0" for unsubscription, and "1" for the first time subscription and the following polls to check if there is any update.

TypedEndpointAddr: the same as defined in previous sections.

GuidanceLevel: A JSONString which defines the level of guidance. This document defines the value of "link" and "AS".

Destination address is optional. When the source end host sends subscription for its TCP parameter guidance on the administrative domain, it does not need the destination address. However, when the

end host sends subscription for the link, it has to provide the destination address.

The media type for subscription is "application/opentcp-sub+json".

#### 4.2.2 Notification

```
object {  
  JSONString: subscription_id;  
  [ConditionedTcpPar: cparameters<0...*>;]  
}TcpParNotify  
  
object {  
  Condition conditions<0...*>;  
  TcpPar: parameters<0...*>;  
}ConditionedTcpPar;
```

subscription-id: a JSONString generated by the TOE to uniquely identify a subscription.

Condition: A condition contains three entities separated by whitespace: (1) a JSONString indicated the link or network status, or the subscriber property, this document defines "link-utilization-rate", "network-utilization-rate", "source-uplink-sub-rate", and "destination-download-sub-rate". (2) an operator, 'gt' for greater than, 'lt' for less than, 'ge' for greater than or equal to, 'le' for less than or equal to, or 'eq' for equal to; (3) a target JSONValue. The JSONValue is a number indicated to compare with the previous status.

The media type for subscription is "application/opentcp-notify+json".

The TCP parameter guidance will be sent to the IP address/port which subscribed earlier. When the template has changed, the TOE will send an immediate notification to relative TOAs.

Note that the guidance delivers the message such as when network utilization is between 50% to 80%, then the recommended parameters are given. So it means the TOA also has to get the change of the relative network status. Network or link status notification was assumed to be provided by other protocols, but if needed, this document can also be expanded to deliver the relative status. (Open issue)

#### 4.3 Error Message

TBD.

## 5 Security Considerations

Dynamic control of TCP parameters can be used for attacks and can cause serious problems to the network or to the applications.

If there are no proper mechanisms to monitor the network, it may be used to maliciously change the TCP parameters and cause network congestion. But in most environments it can be avoided as there are rate limitations.

It can also be used to attack the end hosts. So a mechanism to protect the illegal modification is needed.

## 5 Fairness and Stability

Fairness and stability are guaranteed as long as the changes in parameters are TCP-friendly. In other words, as long as the changes do not deviate the equilibrium formula of TCP protocol. It is possible that the network operator to defines a different metric for fairness such as weighted fairness. For example, the operator might want to give search queries a higher priority compared to background flows. In that sense, fairness between the two classes of flows is not meaningful. However, fairness among one set of flows is guaranteed as long as they are all using the same TCP parameters and follow TCP's algorithm to increase/decrease their congestion window sizes.

In practical settings, we assume that the network operator has expertise in defining the congestion control policies appropriate for the network. To achieve pragmatic stability and fairness, there can be a monitoring system in place which alerts the operator of churns and instabilities in the network. This monitoring component should alert the controller whenever there is an oscillation between states. For example, if the controller is making adjustments to TCP flows in time  $t_1$  and immediately in time  $t_1 + T$  those changes are reverted back, it is a good indication of a potential unstable condition. In this case, the operator should be notified by the monitoring system to adjust either the congestion control policies, the stability constraints, or the overall re- sources in the network. One simple stability metric is number of times a rule is applied and reverted. The monitoring system can measure such stability metrics and alert the operator.

## 6 Acknowledgement

Lingli Deng has provided many valuable comments to this document.

## 7 IANA Considerations

TBD.

## 8 References

### 8.1 Normative References

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

[Hotnets] Ghobadi, M., Yeganeh, S. H., and Y. Ganjali, "Rethinking End-to-End Congestion Control in Software-Defined Networks", Hotnets '12, October 29-30, 2012, Seattle, WA, USA.

### Authors' Addresses

Monia Ghobadi  
Email: monia@cs.toronto.edu

Haibin Song  
EMail: haibin.song@huawei.com

Rachel Huang  
Email: rachel.huang@huawei.com

Yashar Ganjali  
Email: yganjali@cs.toronto.edu



Internet Engineering Task Force  
INTERNET-DRAFT  
Intended Status: Standards Track  
Expires: January 2, 2016

X. Wei  
L.Zhu  
Huawei Technologies  
L.Deng  
China Mobile  
B.Briscoe  
July 1, 2015

Tunnel Congestion Feedback  
draft-wei-tsvwg-tunnel-congestion-feedback-04

## Abstract

This document describes a mechanism to calculate congestion of a tunnel segment based on RFC 6040 recommendations, and a feedback protocol by which to send the measured congestion of the tunnel from egress to ingress. A basic model for measuring tunnel congestion and feedback is described, and a protocol for carrying the feedback data is outlined.

## Status of this Memo

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

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

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

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

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

## Copyright and License Notice

Copyright (c) 2015 IETF Trust and the persons identified as the

document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
2. Conventions . . . . .	3
3. Congestion Information Feedback Models . . . . .	4
3.1 Direct Model . . . . .	4
3.2 Centralized Model . . . . .	4
4. Congestion Level Measurement . . . . .	5
5. Congestion Information Delivery . . . . .	7
5.1 IPFIX Extentions . . . . .	7
5.1.1 ce-cePacketTotalCount . . . . .	7
5.1.2 ect-nectPacketTotalCount . . . . .	8
5.1.3 ce-nectPacketTotalCount . . . . .	8
5.1.4 ce-ectPacketTotalCount . . . . .	8
5.1.5 ect-ectPacketTotalCount . . . . .	9
6. Congestion Management . . . . .	9
7. Security . . . . .	9
8. IANA Considerations . . . . .	10
9. References . . . . .	10
9.1 Normative References . . . . .	10
9.2 Informative References . . . . .	10
Authors' Addresses . . . . .	11



## 1. Introduction

In IP network, persistent congestion (or named congestion collapse) would cause transport throughput to drop down, lead to waste of network resource, so appropriate congestion control mechanisms are critical to make sure the network not fall into persistent congestion state. Currently, transport protocols such as TCP, SCTP, DCCP, has their built-in congestion control mechanism, and even for certain single transport protocol like TCP there could be a couple of different congestion control mechanism to choose. All these congestion control mechanisms are implemented on host side, and there are reasons that only host side congestion control is not sufficient for the whole network to keep away from persistent congestion, e.g., (1) some protocol's congestion control scheme might has internal design flaws; (2) improper software implementation of protocol; (3) some transport protocols even don't provide congestion control at all.

In order to have a better control on network congestion status, it's necessary for the network side to do certain kind of traffic control. For example, ConEx [ConEx] provides a method for network operator to learn about traffic's congestion contribution information, and then congestion management action could be taken based on this information.

Tunnels are widely deployed in various networks including public Internet, datacenter network, and enterprise network etc, a tunnel consists of an ingress, an egress and a set of interior routers. For the tunnel scenario, a tunnel-based mechanism which is different from ConEx is introduced for network traffic control to keep network away from persistent congestion; in this case, tunnel ingress will implement congestion management function to control the traffic entering the tunnel.

In order to do congestion management at ingress, the ingress must first get the inner tunnel congestion level information. But the ingress cannot use the locally visible traffic rates, because it would require additional knowledge of downstream capacity and topology, as well as cross traffic that does not pass through this ingress.

This document provide a mechanism of feeding back inner tunnel congestion level to ingress, using this mechanism the egress could feed the tunnel congestion level information it collects back to ingress, after receiving the information ingress could do congestion management according to network management policy.

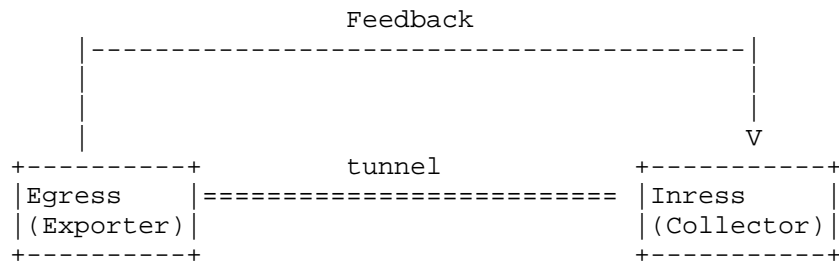
## 2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119]

### 3. Congestion Information Feedback Models

According to specific network deployment, there are two kinds of feedback model: direct model and centralized model.

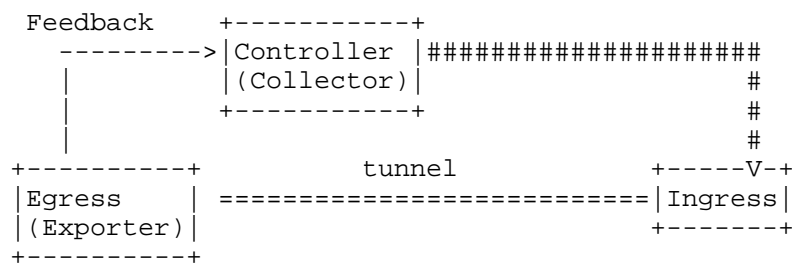
### 3.1 Direct Model



(a) Direct Feedback Model.

Direct model means egress feeds information directly to ingress. In this model, egress collects network congestion level information and feedback the information to ingress for congestion management. The ingress here will act as both decision point that decides how to do congestion management and action point that implements congestion management decision.

### 3.2 Centralized Model



(b) Centralized Feedback Model

There are scenarios that ingress only takes the role of action point, and it implements traffic control decision from another entity, named "controller" here.

In this model, after egress collects network congestion level information, it feeds back the information to controller instead of ingress, and then the controller makes congestion management decision and sends the decision to ingress.

#### 4. Congestion Level Measurement

This section describes how to measure congestion level in tunnel.

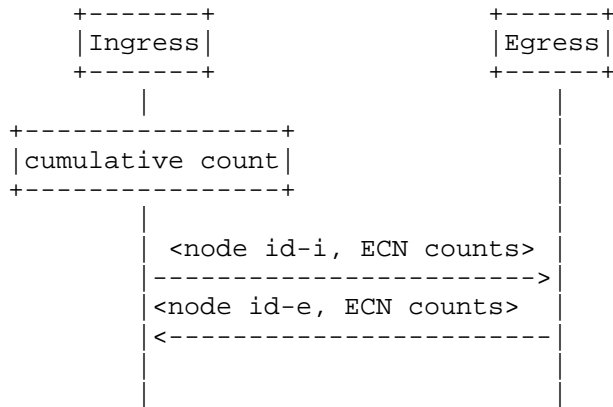
There may be different approaches of packet loss detection for different tunneling protocol scenarios, for instance, if there is a sequence field in tunneling protocol header, it will be easy for egress to detect packet loss through the gaps in sequence number space; another approach is to compare the number of packets entering ingress and the number of packets arriving at egress over the same span of packets. This document will focus on the latter one which is a more general approach.

If the routers support ECN, after router's queue length is over a predefined threshold, the routers will mark ECN packets as CE packets or drop not-ECN packets with the probability proportional to queue length, if the queue overflows all packets will be dropped; if the routers don't support ECN, after router's queue length is over a predefined threshold, the routers will drop both ECN packets and not-ECN packets with the probability proportional to queue length. It's assumed all routers in the tunnel support ECN.

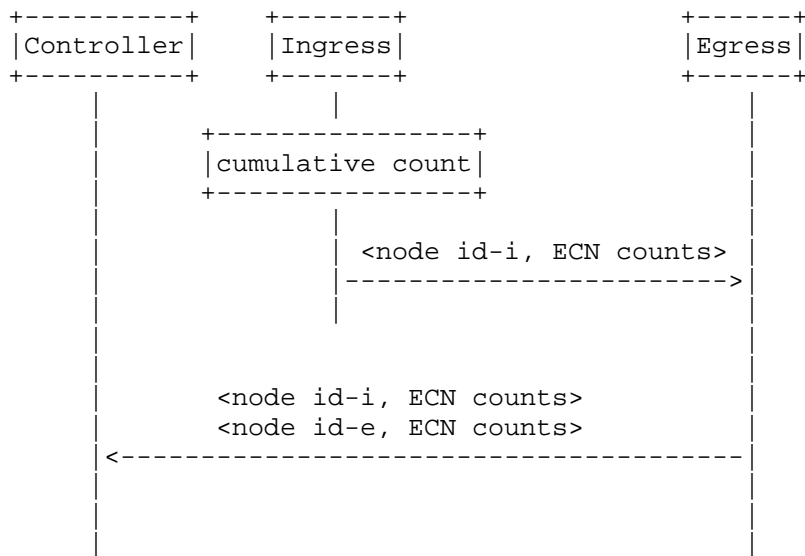
Faked ECT is used at ingress to defer packet loss to egress. The basic idea of faked ECT is that, when encapsulating packets, ingress first marks tunnel outer header according to RFC6040, and then remarks outer header of Not-ECT packet as ECT, there will be three kinds of combination of outer header ECN field and inner header ECN field: CE|CE, ECT|N-ECT, ECT|ECT (in the form of outer ECN| inner ECN).

In case all interior routers support ECN, the network congestion level could be indicated through the ratio of CE-marked packet and the ratio of packet drop, the relationship between these two kinds of indicator is complementary. If the congestion level in tunnel is not high enough, the packets would be marked as CE instead of being dropped, and then it is easy to calculate congestion level according to the ratio of CE-marked packets; if the congestion level is so high that ECT packet will be dropped, then the packet loss ratio could be calculated by comparing total packets entering ingress and total packets arriving at egress over the same span of packets, if packet loss is detected, it could be assumed that severe congestion has occurred in the tunnel, because loss is only ever a sign of serious congestion, so it doesn't need to measure loss ratio accurately.

The basic procedure of congestion level measurement is as follows:



(a) Direct model feedback procedure



(b) Centralized model feedback procedure

Ingress encapsulates packets and marks outer header according to faked ECT as described above. Ingress cumulatively counts packets for three types of ECN combination (CE|CE, ECT|N-ECT, ECT|ECT) and then the ingress regularly sends cumulative packet counts message of each type of ECN combination to the egress. When each message arrives, the

egress cumulatively counts packets coming from the ingress and adds its own packet counts of each type of ECN combination (CE|CE, ECT|N-ECT, CE|N-ECT, CE|ECT, ECT|ECT) to the message and either returns the whole message to the ingress, or to a central controller.

The counting of packets could be at the granularity of the all traffic from the ingress to the egress to learn about the overall congestion status of the path between the ingress and the egress; or at the granularity of individual customer's traffic or a specific set of flows to learn about their congestion contribution.

## 5. Congestion Information Delivery

As described above, the tunnel ingress needs to convey message of cumulative packet counts of each type of ECN combination to tunnel egress, and the tunnel egress also needs to feed the message of cumulative packet counts of each type of ECN combination to the ingress or central collector. This section describes how the messages could be conveyed.

The message could be along the same path with network data traffic, referred as in band signal; or go through a different path with network data traffic, referred as out of band signal. Because out of band scheme needs additional separate path which might limit its actual deployment, so the in band scheme will be discussed here.

Because the message is transmitted in band, so the message packet might get lost in case of network congestion. To cope with the situation that message packet gets lost, the packet counts values are sent as cumulative counters, so if a message is lost the next message will recover the missing information.

IPFIX [RFC7011] is selected as a choice of candidate protocol. IPFIX is preferred to use SCTP as transport, and because SCTP allows partially reliable delivery [RFC3758], which makes sure the feedback message will not be blocked to be sent in case of SCTP packets lost due to network congestion.

When sending message from ingress to egress, the ingress acts as IPFIX exporter and egress acts as IPFIX collector; when sending message from egress to ingress or controller, the egress acts as IPFIX exporter and ingress or controller acts as IPFIX collector.

### 5.1 IPFIX Extensions

#### 5.1.1 ce-cePacketTotalCount

Description: The total number of incoming packets with CE|CE ECN

marking combination for this Flow at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD1

Statuses: current

Units: packets

#### 5.1.2 ect-nectPacketTotalCount

Description: The total number of incoming packets with ECT|N-ECT ECN marking combination for this Flow at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD2

Statuses: current

Units: packets

#### 5.1.3 ce-nectPacketTotalCount

Description: The total number of incoming packets with CE|N-ECT ECN marking combination for this Flow at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD3

Statuses: current

Units: packets

#### 5.1.4 ce-ectPacketTotalCount

Description: The total number of incoming packets with CE|ECT ECN

marking combination for this Flow at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD4

Statuses: current

Units: packets

#### 5.1.5 ect-ectPacketTotalCount

Description: The total number of incoming packets with ECT|ECT ECN marking combination for this Flow at the Observation Point since the Metering Process (re-)initialization for this Observation Point.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD5

Statuses: current

Units: packets

### 6. Congestion Management

After tunnel ingress (or controller) receives congestion level information, then congestion management actions could be taken based on the information, e.g. if the congestion level is higher than a predefined threshold, then action could be taken to reduce the congestion level.

Congestion management action must be delayed by more than a worst-case global RTT, otherwise tunnel traffic management will not give normal e2e congestion control enough time to do its job, and the system could go unstable. The detailed description of congestion management is out of scope of this document, as examples, congestion management such as circuit breaker [CB] and congestion policing [CP] could be applied.

### 7. Security

This document describes the tunnel congestion calculation and

feedback. For feeding back congestion, security mechanisms of IPFIX are expected to be sufficient. No additional security concerns are expected.

## 8. IANA Considerations

This document defines a set of new IPFIX Information Elements (IE). New registry for these IE identifiers is needed.

TBD1~TBD5.

## 9. References

### 9.1 Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001, <<http://www.rfc-editor.org/info/rfc3168>>.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004, <<http://www.rfc-editor.org/info/rfc3758>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, September 2013, <<http://www.rfc-editor.org/info/rfc7011>>.

### 9.2 Informative References

- [CONEX] Matt Mathis, Bob Briscoe. "Congestion Exposure (ConEx) Concepts, Abstract Mechanism and Requirements", draft-ietf-conex-abstract-mech-13, October 24, 2014
- [CB] G. Fairhurst. "Network Transport Circuit Breakers", draft-ietf-tsvwg-circuit-breaker-01, April 02, 2015
- [CP] Bob Briscoe, Murari Sridharan. "Network Performance Isolation in Data Centres using Congestion Policing", draft-briscoe-



conex-data-centre-02, February 14, 2014

Authors' Addresses

Xinpeng Wei  
Beiqing Rd. Z-park No.156, Haidian District,  
Beijing, 100095, P. R. China  
E-mail: weixinpeng@huawei.com

Zhu Lei  
Beiqing Rd. Z-park No.156, Haidian District,  
Beijing, 100095, P. R. China  
E-mail: lei.zhu@huawei.com

Lingli Deng  
Beijing, 100095, P. R. China  
E-mail: denglingli@gmail.com

Bob Briscoe  
B54/77, Adastral Park  
Martlesham Heath  
Ipswich IP5 3RE  
UK