

Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 19, 2009

S. Bryant
B. Davie
L. Martini
E. Rosen
Cisco Systems, Inc.
June 17, 2009

Pseudowire Congestion Control Framework
draft-ietf-pwe3-congestion-frmwk-02.txt

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/ietf/lid-abstracts.txt>.

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

This Internet-Draft will expire on December 19, 2009.

Copyright Notice

Copyright (c) 2009 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 in effect on the date of publication of this document (<http://trustee.ietf.org/license-info>). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

Pseudowires are sometimes used to carry non-TCP data flows. In these

Internet-Draft

PWE3 Congestion

June 2009

circumstances the service payload will not react to network congestion by reducing its offered load. Pseudowires should therefore reduce their network bandwidth demands in the face of significant packet loss, including if necessary completely ceasing transmission. Since it is difficult to determine a priori the number of equivalent TCP flow that a pseudowire represents, a suitably "fair" rate of back-off cannot be pre-determined. This document describes pseudowire congestion problem and provides guidance on the development suitable solutions.

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 [RFC 2119](#) [[RFC2119](#)].

Internet-Draft

PWE3 Congestion

June 2009

Table of Contents

1.	Introduction	4
2.	PW Congestion Context	4
2.1.	Congestion in IP Networks	4
2.2.	A Short Tutorial on TCP Congestion Control	5
2.3.	Alternative Approaches to Congestion Control	6
2.4.	Pseudowires and TCP	7
2.5.	Arguments Against PW Congestion as a Practical Problem	8
2.6.	Goals and non-goals of this draft	9
3.	Challenges for PW Congestion Management	10
3.1.	Scale	10
3.2.	Interaction among control loops	10
3.3.	Determining the Appropriate Rate	11
3.4.	Constant Bit Rate PWs	11
3.5.	Valuable and Vulnerable PWs	12
4.	Congestion Control Mechanisms	12
5.	Detecting Congestion	13
5.1.	Using Sequence Numbers to Detect Congestion	14
5.2.	Using VCCV to Detect Congestion	15
5.3.	Explicit Congestion Notification	16
6.	Feedback from Receiver to Transmitter	16
6.1.	Control Plane Feedback	17
6.2.	Using Reverse Data Packets for Feedback	18
6.3.	Reverse VCCV Traffic	18
7.	Responding to Congestion	18
7.1.	Interaction with TCP	19
8.	Rate Control per Tunnel vs. per PW	20
9.	Constant Bit Rate Services	21
10.	Managed vs Unmanaged Deployment	21
11.	Related Work: Pre-Congestion Notification	22
12.	IANA Considerations	23
13.	Security Considerations	23
14.	Conclusion	23
15.	Informative References	23
	Authors' Addresses	25

1. Introduction

Pseudowires are sometimes used to emulate services network that carry non-TCP data flows. In these circumstances the service payload will not react to network congestion by reducing its offered load. In order to protect the network, pseudowires (PW) should therefore reduce their network bandwidth demands in the face of significant packet loss, including if necessary completely ceasing transmission.

Pseudowires are commonly deployed as part of the managed infrastructure deployed by a service provider. It is likely in these cases that the PW will be used to carry many TCP equivalent flows. In these environments the operator needs to make a judgement call on a fair estimate of the number of equivalent flows that the PW represents. The counterpoint to a well managed network is a plug and play network, of the type typically found in a consumer environment. Whilst PWs have been designed to provide an infrastructure tool to the service provider, we cannot be sure that they will not find some consumer or similar plug and play application. In this environment a conservative is needed, with PW congestion avoidance enabled by default, and the throughput set by default to the equivalent of a single TCP flow.

This framework first considers the context in which PWs operate, and hence the context in which PW congestion control needs to operate. It then explores the issues of detection, feedback and load control in PW context, and the special issues that apply to constant bit rate services. Finally it considers the applicability of congestion control in various network environments.

This document defines the framework to be used in designing a congestion control mechanism for PWs. It does not prescribe a solution for an particular PW type, and it is possible that more than one mechanism may be required, depending on the PW type and the environment in which it is to be deployed.

[2.](#) PW Congestion Context

[2.1.](#) Congestion in IP Networks

Congestion in an IP or MPLS enabled IP network occurs when the amount of traffic that needs to use a particular network resource exceeds the capacity of that resource. This results first in long queues within the network, and then in packet loss. If the amount of traffic is not then reduced, the packet loss rate will climb, potentially until it reaches 100%. If the situation is left unabated the network enters a state where it is no longer able to sustain any

productive information flows and the result is "congestive collapse"

To prevent this sort of "congestive collapse", there must be congestion control: a feedback loop by which the presence of congestion anywhere on the path from the transmitter to the receiver forces the transmitters to reduce the amount of traffic being sent. As a connectionless protocol, IP has no way to push back directly on the originator of the traffic. Procedures for (a) detecting congestion, (b) providing the necessary feedback to the transmitters, and (c) adjusting the transmission rates, are thus left to higher protocol layers such as TCP.

[2.2.](#) A Short Tutorial on TCP Congestion Control

TCP includes an elaborate congestion control mechanism that causes the end systems to reduce their transmission rates when congestion occurs. For those readers not intimately familiar with the details of TCP congestion control, we give below a brief summary, greatly simplified and not entirely accurate, of TCP's very complicated feedback mechanism. The details of TCP congestion control can be found in [\[RFC2581\]](#). [\[RFC2001\]](#) is an earlier but more accessible discussion. [\[RFC2914\]](#) articulates a number of general principles

governing congestion control in the Internet.

In TCP congestion control, a lost packet is considered to be an indication of congestion. Roughly, TCP considers a given packet to be lost if that packet is not acknowledged within a specified time, or if three subsequent packets arrive at the receiver before the given packet. The latter condition manifests itself at the transmitter as the arrival of three duplicate acknowledgements in a row. The algorithm by which TCP detects congestion is thus highly dependent on the mechanisms used by TCP to ensure reliable and sequential delivery.

Once a TCP transmitter becomes aware of congestion, it halves its transmission rate. If congestion still occurs at the new rate, the rate is halved again. When a rate is found at which congestion no longer occurs, the rate is increased by one MSS ("Maximum Segment Size") per RTT ("Round Trip Time"). The rate is increased each RTT until congestion is encountered again, or until something else limits it (e.g., the flow control window reached, or the application is transmitting at its max desired rate, or at line rate).

This sort of mechanism is known as an "Additive Increase, Multiplicative Decrease" (AIMD) mechanism. Congestion causes relatively rapid decreases in the transmission rate, while the absence of congestion causes relatively slow increases in the allowed transmission rate.

[2.3.](#) Alternative Approaches to Congestion Control

[RFC2914] defines the notion of a "TCP-compatible flow":

"A TCP-compatible flow is responsive to congestion notification, and in steady state uses no more bandwidth than a conformant TCP running under comparable conditions (drop rate, RTT [round trip time], MTU [maximum transmission unit], etc.)"

TCP-compatible flows respond to congestion in much the way TCP does, so that they do not starve the TCP flows or otherwise obtain an unfair advantage. [\[RFC2914\]](#) further points out:

"any form of congestion control that successfully avoids a high sending rate in the presence of a high packet drop rate should be

sufficient to avoid congestion collapse from undelivered packets."

"This does not mean, however, that concerns about congestion collapse and fairness with TCP necessitate that all best-effort traffic deploy congestion control based on TCP's Additive-Increase Multiplicative-Decrease (AIMD) algorithm of reducing the sending rate in half in response to each packet drop."

"However, the list of TCP-compatible congestion control procedures is not limited to AIMD with the same increase/ decrease parameters as TCP. Other TCP-compatible congestion control procedures include rate-based variants of AIMD; AIMD with different sets of increase/ decrease parameters that give the same steady-state behavior; equation-based congestion control where the sender adjusts its sending rate in response to information about the long-term packet drop rate ... and possibly other forms that we have not yet begun to consider."

The AIMD procedures are not mandated for non-TCP traffic, and might not be optimal for non-TCP PW traffic. Choosing a proper set of procedures which are TCP-compatible while being optimized for a particular type of traffic is no simple task.

[[RFC3448](#)], "TCP Friendly Rate Control (TFRC)" provides an alternative: TFRC is designed to be reasonably fair when competing for bandwidth with TCP flows, where a flow is "reasonably fair" if its sending rate is generally within a factor of two of the sending rate of a TCP flow under the same conditions. However, TFRC has a much lower variation of throughput over time compared with TCP, which makes it more suitable for applications such as telephony or streaming media where a relatively smooth sending rate is of importance."

"For its congestion control mechanism, TFRC directly uses a throughput equation for the allowed sending rate as a function of the loss event rate and round-trip time. In order to compete fairly with TCP, TFRC uses the TCP throughput equation, which roughly describes TCP's sending rate as a function of the loss event rate, round-trip time, and packet size."

"Generally speaking, TFRC's congestion control mechanism works as

follows:

- o The receiver measures the loss event rate and feeds this information back to the sender.
- o The sender also uses these feedback messages to measure the round-trip time (RTT).
- o The loss event rate and RTT are then fed into TFRC's throughput equation, giving the acceptable transmit rate.
- o The sender then adjusts its transmit rate to match the calculated rate."

Note that the TFRC procedures require the transmitter to calculate a throughput equation. For these procedures to be feasible as a means of PW congestion control, they must be computationally efficient. [Section 8 of \[RFC3448\]](#) describes an implementation technique that appears to make it efficient to calculate the equation.

In the context of pseudowires, we note that TFRC aims to achieve comparable throughput in the long term to a single TCP connection experiencing similar loss and round trip time. As noted above, this may not be an appropriate rate for a PW that is carrying many flows.

[2.4.](#) Pseudowires and TCP

Currently, traffic in IP and MPLS networks is predominantly TCP traffic. Even the layer 2 tunnelled traffic (e.g., PPP frames tunnelled through L2TP) is predominantly TCP traffic from the end-users. If pseudowires (PWs) [\[RFC3985\]](#) were to be used only for carrying TCP flows, there would be no need for any PW-specific congestion mechanisms. The existing TCP congestion control mechanisms would be all that is needed, since any loss of packets on the PW would be detected as loss of packets on a TCP connection, and the TCP flow control mechanisms would ensure a reduction of transmission rate.

If a PW is carrying non-TCP traffic, then there is no feedback mechanism to cause the end-systems to reduce their transmission rates

in response to congestion. When congestion occurs, any TCP traffic

that is sharing the congested resource with the non-TCP traffic will be throttled, and the non-TCP traffic may "starve" the TCP traffic. If there is enough non-TCP traffic to congest the network all by itself, there is nothing to prevent congestive collapse.

The non-TCP traffic in a PW can belong to any higher layer whatsoever, and there is no way to ensure that a TCP-like congestion control mechanisms will be present to regulate the traffic rate. Hence it appears that there is a need for an edge-to-edge (i.e., PE-to-PE) feedback mechanism which forces a transmitting PE to reduce its transmission rate in the face of network congestion.

As TCP uses window-based flow control, controlling the rate is really a matter of limiting the amount of traffic which can be "in flight" (i.e., transmitted but not yet acknowledged) at any one time. Where a non-windowed protocol is used for transmitting data on a PW a different technique is obviously required to control the transmission rate.

[2.5.](#) Arguments Against PW Congestion as a Practical Problem

One may argue that congestion due to non-TCP PW traffic is only a theoretical problem and that no congestion control mechanism is needed. For example the following cases have been put forward:

- o "99.9% of all the traffic in PWs is really IP traffic"

If this is the case, then the traffic is either TCP traffic, which is already congestion-controlled, or "other" IP traffic. While the congestion control issue may exist for the "other" IP traffic, this is a general issue and hence is outside the scope of the pseudowire design.

Unfortunately, we cannot be sure that this is the case. It may well be the case for the PW offerings of certain providers, but perhaps not for others. It does appear that many providers want to be able to use PWs for transporting "legacy traffic" of various non-IP protocols. Constant bit-rate services are an example of this, and raise particular issues for congestion control (discussed below).

- o "PW traffic usually stays within one SP's network, and an SP always engineers its network carefully enough so that congestion is an impossibility"

Perhaps this will be true of "most" PWs, but inter-provider PWs are certainly expected to have a significant presence.

Even within a single provider's network, the provider might consider whether he is so confident of his network engineering that he does not need a feedback loop reducing the transmission rate in response to congestion.

There is also the issue of keeping the network running (i.e., out of congestive collapse) after an unexpected reduction of capacity.

Finally the PW may be deployed on the Internet rather than in an SP environment.

- o "If one provider accepts PW traffic from another, policing will be done at the entry point to the second provider's network, so that the second provider is sure that the first provider is not sending too much traffic. This policing, together with the second provider's careful network engineering, makes congestion an impossibility"

This could be the case given carefully controlled bilateral peering arrangements. Note though that if the second provider is merely providing transit services for a PW whose endpoints are in other providers, it may be difficult for the transit provider to tell which traffic is the PW traffic and which is "ordinary" IP traffic.

- o "The only time we really need a general congestion control mechanism is when traffic goes through the public Internet. Obviously this will never be the case for PW traffic."

It is not at all difficult to imagine someone using an IPsec tunnel across the public Internet to transport a PW from one private IP network to another.

Nor is it difficult to imagine some enterprise implementing a PW and transporting it across some SP's backbone, e.g., if that SP is providing VPN service to that enterprise.

The arguments that non-TCP traffic in PWs will never make any significant contribution to congestion thus do not seem to be totally compelling.

[2.6.](#) Goals and non-goals of this draft

As a framework, this draft aims to explore the issues surrounding PW congestion and to lay out some of the design trade offs that will need to be made if a solution is to be developed. It does not intend

to propose a particular solution, but it does point out some of the problems that will arise with certain solution approaches.

The over-riding technical goal of this work is to avoid scenarios in which the deployment of PW technology leads to congestive collapse of the PSN over which the PWs are tunnelled. It is a non-goal of this work to ensure that PWs receive a particular Quality of Service (QoS) particular level in order to protect them from congestion(([Section 3.5](#))). While such an outcome may be desirable, it is beyond the scope of this draft.

[3.](#) Challenges for PW Congestion Management

[3.1.](#) Scale

It might appear at first glance that an easy solution to PW congestion control would be to run the PWs through a TCP connection. This would provide congestion control automatically. However, the overhead is prohibitive for the PW application. The PWE3 data plane may be implemented in a micro-coded or hardware engine which needs to support thousands of PWs, and needs to do as little as possible for each data packet; running a TCP state machine, and implementing TCP's flow control procedures, would impose too high a processing and memory cost in this environment. Nor do we want to add the large overhead of TCP to the PWs -- the large headers, the plethora of small acknowledgements in the reverse direction, etc., etc. In fact, we need to avoid acknowledgements altogether. These same considerations lead us away from using e.g., DCCP [[RFC4340](#)], or SCTP [[RFC4960](#)], even in its partially reliable mode [[RFC3758](#)]. Therefore we will investigate some PW-specific solutions for congestion control.

The PW designed also needs to minimise the amount of interaction between the data processing path (which is likely to be distributed among a set of line cards) and the control path; be especially careful of interactions which might require atomic read/modify/write operations from the control path, or which might require atomic read/modify/write operations between different processors in a multiprocessing implementation, as such interactions can cause scaling problems.

Thus, feasible solutions for PW-specific congestion will require scalable means to detect congestion and to reduce the amount of traffic sent into the network when congestion is detected. These topics are discussed in more detail in subsequent sections.

[3.2.](#) Interaction among control loops

As noted above, much of the traffic that is carried on PWs is likely to be TCP traffic, and will therefore be subject the congestion

Bryant, et al.

Expires December 19, 2009

[Page 10]

Internet-Draft

PWE3 Congestion

June 2009

control mechanisms of TCP. It will typically be difficult for a PW endpoint to tell whether or not this is the case. Thus there is the risk that the PE-PE congestion control mechanisms applied over the PW may interact in undesirable ways with the end-to-end congestion control mechanisms of TCP. The PW-specific congestion control mechanisms should be designed to minimise the negative impact of such interaction.

[3.3.](#) Determining the Appropriate Rate

TCP tends to share the bandwidth of a bottleneck among flows somewhat evenly, although flows with shorter round-trip-times and larger MSS values will tend to get more throughput in the steady state than those with longer RTTs or smaller MSS. TFRC simply tries to deliver the same rate to a flow that a TCP flow would obtain in the steady state under similar conditions (loss rate, RTT and MSS). The challenge in the PW environment is to determine what constitutes a "flow". While it is tempting to consider a single PW to be equivalent to a "flow" for the purposes of fair rate estimation, it is likely in many cases that one PW will be carrying a large number of flows, in which case it would seem quite unfair to throttle that PW down to the same rate that a single TCP flow would obtain.

The issue of what constitutes fairness and the perils of using the TCP flow as the basic unit of fairness have been explored at some length in [[I-D.briscoe-tsvarea-fair](#)].

In the PW environment some estimate (measured or configured) of the number flows in the PW needs to be applied to the estimate of fair throughput as part of the process of determining appropriate congestion behavior.

[3.4.](#) Constant Bit Rate PWs

Some types of PW, for example SAToP (Structure Agnostic TDM over Packet) [[RFC4553](#)], CESoPSN (Circuit Emulation over Packet Switched Networks) [[RFC5086](#)], TDM over IP [[RFC5087](#)][[RFC4842](#)], SONET/SDH and Constant Bit Rate ATM PWs represent an inelastic constant bit-rate (CBR) flow. Such PWs cannot respond to congestion in a TCP-friendly manner prescribed by [[RFC2914](#)]. However this inability to respond well to congestion by reducing the amount of total bandwidth consumed is offset by the fact that such a PW maintains a constant bandwidth demand rather than being greedy (in the sense of trying to increase their share of a link, as TCP does). AIMD or even more gradual TFRC techniques are clearly not applicable to such services; it is not feasible to reduce the rate of a CBR service without violating the service definition. Such services are also frequently more sensitive to packet loss than connectionless packet PWs. Given that CBR

services are not greedy, there may be a case for allowing them greater latitude during congestion peaks. If some CBR PWs are not able to endure any significant packet loss or reduction in rate without compromising the transported service, such PWs must be shutdown when the level of congestion becomes excessive, but at suitably low levels of congestion they should be allowed to continue to offer traffic to the network.

Some CBR services may be carried over connectionless packet PWs. An example of such a case would be a CBR MPEG-2 video stream carried over an Ethernet PW. One could argue that such a service - provided the rate was policed at the ingress PE - should be offered the same latitude as a PW that explicitly provided a CBR service. Likewise, there may not be much value in trying to throttle such a service rather than cutting it off completely during severe congestion. However, this clearly raises the issue of how to know that a PW is indeed carrying a CBR service.

[3.5.](#) Valuable and Vulnerable PWs

Some PWs are a premium service for which the user has paid a premium to the provider for higher availability, lower packet loss rate etc. Some PWs, for example the CBR services described in [Section 3.4](#) are particularly vulnerable to packet loss. In both of these cases it may be tempting to relax the congestion considerations so that these

services can press on as best they can in the event of congestion. However a more appropriate strategy is to engineer the network paths, QoS parameters, queue sizes and scheduling parameters so that these services do not suffer congestion discard. If despite this network engineering these services still experience congestion, that is an indication that the network is having difficulty servicing their needs, and the services, like any other service should reduce their network load.

[4.](#) Congestion Control Mechanisms

There are three components of the congestion control mechanism that we need to consider:

1. Congestion state detection
2. Feedback from the receiver to the transmitter
3. Method used by the transmitter to respond to congestion state

Reference to congestion state apply to both the detection of the onset of the congestion state, and detection of the return of the

network to a state in which greater or normal bandwidth is available.

We discuss the design framework for each of these components in the sections below.

[5.](#) Detecting Congestion

In TCP, congestion is detected by the transmitter; the receipt of three successive duplicate TCP acknowledgements are taken to be indicative of congestion. What this actually means is that the several packets in a row were received at the remote end, such that none of those packets had the next expected sequence number. This is interpreted as meaning that the packet with the next expected sequence number was lost in the network, and the loss of a single packet in the network is taken as a sign of congestion. (Naturally, the presence of congestion is also inferred if TCP has to retransmit a packet.) Note that it is possible for misordered packets to be

misinterpreted as lost packets, if they do not arrive "soon enough".

In TCP, a time-out while awaiting an acknowledgement is also interpreted as a sign of congestion.

Since there are normally no acknowledgements on a PW (the only PW design that has so far attempted a sophisticated flow control is [[I-D.ietf-pwe3-fc-flow](#)]), the PW-specific congestion control mechanism cannot normally be based on either the presence of or the absence of acknowledgements. Some types of pseudowire (the CBR PWs) have a single bit that indicates that a preset amount of data has been lost, but this is a non-quantitative indicator. CBR PWs have the advantage that there is a constant two way data flow, while other PW types do not have the constant symmetric flow of payload on which to piggyback the congestion notification. Most PW types therefore provide no way for a transmitter to determine (or even to make an educated guess as to) whether any data has been lost.

Thus we need to add a mechanism for determining whether data packets on a PW have become lost. There are several possible methods for doing this:

- o Detect Congestion Using PW Sequence Numbers
- o Detect Congestion Using Modified VCCV Packets [[RFC5085](#)]
- o Rely on Explicit Congestion Notification (ECN) [[RFC3168](#)]

We discuss each option in turn in the following sections.

[5.1.](#) Using Sequence Numbers to Detect Congestion

When the optional sequencing feature is in use on a PW [[RFC4385](#)], it is necessary for the receiver to maintain a "next expected sequence number" for the PW. If a packet arrives with a sequence number that is earlier than the next expected (a "misordered packet"), the packet is discarded; if it arrives with a sequence number that is greater than or equal to the next expected, the packet is delivered, and the next expected sequence number becomes the sequence number of the current packet plus 1.

It is easy to tell when there is one or more missing packets (i.e., there is a "gap" in the sequence space) -- that is the case when a packet arrives whose sequence number is greater than the next expected. What is difficult to tell is whether any misordered packets that arrive after the gap are indeed the missing packets. One could imagine that the receiver remembers the sequence number of each missing packet for a period of time, and then checks off each such sequence number if a misordered packet carrying that sequence number later arrives. The difficulty is doing this in a manner which is efficient enough to be done by the microcoded hardware handling the PW data path. This approach does not really seem feasible.

One could make certain simplifying assumptions, such as assuming that the presence of any gaps at all indicates congestion. While this assumption makes it feasible to use the sequence numbers to "detect congestion", it also throttles the PW unnecessarily if there is really just misordering and no congestion. Such an approach would be considerably more likely to misinterpret misordering as congestion than would TCP's approach.

An intermediate approach would be to keep track of the number of missing packets and the number of misordered packets for each PW. One could "detect congestion" if the number of missing packets is significantly larger than the number of misordered packets over some sampling period. However, gaps occurring near the end of a sampling period would tend to result in false indications of congestion. To avoid this one might try to smooth the results over several sampling periods; While this would tend to decrease the responsiveness, it is inevitable that there will be a trade-off between the rapidity of responsiveness and the rate of false alarms.

One would not expect the hardware or microcode to keep track of the sampling period; presumably software would read the necessary counters from hardware at the necessary intervals.

Such a scheme would have the advantage of being based on existing PW mechanisms. However, it has the disadvantage of requiring

sequencing, which introduces a fairly complicated interaction between the control processing and the data path, and precludes the use of some pipelined forwarding designs.

[5.2.](#) Using VCCV to Detect Congestion

It is reasonable to suppose that the hardware keeps counts of the number of packets sent and received on each PW. Suppose that the PW periodically inserts VCCV packets into the PE data stream, where each VCCV packet carries:

- o A sequence number, increasing by 1 for each successive VCCV packet;
- o The current value of the transmission counter for the PW

We assume that the size of the counter is such that it cannot wrap during the interval between n VCCV packets, for some $n > 1$.

When the receiver gets one of these VCCV packets on a PW, it inserts into the packet, or packet metadata the count of received packets for that PW, and then delivers the VCCV packet to the software. The receiving software can now compute, for the inter-VCCV intervals, the count of packets transmitted and the count of packets received. The presence of congestion can be inferred if the count of packets transmitted is significantly greater than the count of packets received during the most recent interval. Even the loss rate could be calculated. The loss rate calculated in this way could be used as input to the TFRC rate equation.

VCCV messages would not need to be sent on a PW (for the purpose of detecting congestion) in the absence of traffic on that PW.

Of course, misordered packets that are sent during one interval but arrive during the next will throw off the loss rate calculation; hence the difference between sent traffic and received traffic should be "significant" before the presence of congestion is inferred. The value of "significance" can be made larger or smaller depending on the probability of misordering.

Note that congestion can cause a VCCV packet to go missing, and anything that misorders packets can disorder a VCCV packet as well as any other. One may not want to infer the presence of congestion if a single VCCV packet does not arrive when expected, as it may just be delayed in the network, even if it hasn't been misordered. However, failure to receive a VCCV packet after a certain amount of time has elapsed since the last VCCV was received (on a particular PW) may be taken as evidence of congestion. This scheme has the disadvantage of

requiring periodic VCCV packets, and it requires VCCV packet formats to be modified to include the necessary counts. However, the interaction between the control path and the data path is very simple, as there is no polling of counters, no need for timers in the data path, and no need for the control path to do read-modify-write operations on the data path hardware. A bigger disadvantage may arise from the possible inability to ensure that the transmit counts in the VCCVs are exactly correct. The transmitting hardware may not be able to insert a packet count in the VCCV IMMEDIATELY before transmission of the VCCV on the wire, and if it cannot, the count of transmit packets will only be approximate.

Neither scheme can provide the same type of continuous feedback that TCP gets. TCP gets a continuous stream of acknowledgments, whereas this type of PW congestion detection mechanism would only be able to say whether congestion occurred during a particular interval. If the interval is about 1 RTT, the PW congestion control would be approximately as responsive as TCP congestion control, and there does not seem to be any advantage to making it smaller. However, sampling at an interval of 1 RTT might generate excessive amounts of overhead. Sampling at longer intervals would reduce responsiveness to congestion but would not necessarily render the congestion control mechanism "TCP-unfriendly".

[5.3.](#) Explicit Congestion Notification

In networks that support explicit congestion notification (ECN) [[RFC3168](#)] the ECN notification provides congestion information to the PEs before the onset of congestion discard. This is particularly useful to PWs that are sensitive to packet loss, since it gives the PE the opportunity to intelligently reduce the offered load. ECN marking rates of packets received on a PW could be used to calculate the TFRC rate for a PW. However ECN is not widely deployed at the time of writing; hence it seems that PEs must also be capable of operating in a network where packet loss is the only indicator of congestion.

[6.](#) Feedback from Receiver to Transmitter

Given that the receiver can tell, for each sampling interval, whether or not a PW's traffic has encountered congestion, the receiver must provide this information as feedback to the transmitter, so that the transmitter can adjust its transmission rate appropriately. The feedback could be as simple as a bit stating whether or not there was any packet loss during the specified interval. Alternatively, the actual loss rate could be provided in the feedback, if that

information turns out to be useful to the transmitter (e.g. to enable

it to calculate an appropriate rate at which to send). There are a number of possible ways in which the feedback can be provided: control plane, reverse data traffic, or VCCV messages. We discuss each in turn below.

[6.1.](#) Control Plane Feedback

A control message can be sent periodically to indicate the presence or absence of congestion. For example, when LDP is the control protocol [[RFC4447](#)], the control message would of course be delivered reliably by TCP. (The same considerations apply for any protocol which has a reliable control channel.) When congestion is detected, a control message can be sent indicating that fact. No further congestion control messages would need to be sent until congestion is no longer detected. If the loss rate is being sent, changes in the loss rate would need to be sent as well. When there is no longer any congestion, a message indicating the absence of congestion would have to be sent.

Since congestion in the reverse direction can prevent the delivery of these control messages, periodic "no congestion detected" messages would need to be sent whenever there is no congestion. Failure to receive these in a timely manner would lead the control protocol peer to infer that there is congestion. (Actually, there might or might not be congestion in the transmitting direction, but in the absence of any feedback one cannot assume that everything is fine.) If control messages really cannot get through at all, control protocol keep-alives will fail and the control connection will go down anyway.

If the control messages simply say whether or not congestion was detected, then given a reliable control channel, periodic messages are not needed during periods of congestion. Of course, if the control messages carry more data, such as the loss rate, then they need to be sent whenever that data changes.

If it is desired to control congestion on a per-tunnel basis, these control messages will simply say that there was congestion on some PW (one or more) within the tunnel. If it is desired to control congestion on a per-PW basis, the control message can list the PWs which have experienced congestion, most likely by listing the

corresponding labels. If the VCCV method of detecting congestion is used, one could even include the sent/received statistics for particular VCCV intervals.

This method is very simple, as one does not have to worry about the congestion control messages themselves getting lost or out of sequence. Feedback traffic is minimized, as a single control message relays feedback about an entire tunnel.

[6.2.](#) Using Reverse Data Packets for Feedback

If a receiver detects congestion on a particular PW, it can set a bit in the data packets that are travelling on that PW in the reverse direction; when no congestion is detected, the bit would be clear. The bit would be ignored on any packet that is received out of sequence, of course. There are several disadvantages to this technique:

- o There may be no (or insufficient) data traffic in the reverse direction
- o Sequencing of the data stream is required
- o The transmission of the congestion indications is not reliable
- o The most one could hope to convey is one bit of information per PW (if there is even a bit available in the encapsulation).

[6.3.](#) Reverse VCCV Traffic

Congestion indications for a particular PW could be carried in VCCV packets travelling in the reverse direction on that PW. Of course, this would require that the VCCV packets be sent periodically in the reverse direction whether or not there is reverse direction traffic. For congestion feedback purposes they might need to be sent more frequently than they'd need to be sent for OAM purposes. It would also be necessary for the VCCVs to be sequenced (with respect to each other, not necessarily with respect to the data stream). Since VCCV transmission is unreliable, one would want to send multiple VCCVs within whatever period we want to be able to respond in. Further, this method provides no means of aggregating congestion information into information about the tunnel.

7. Responding to Congestion

In TCP, one tends to think of the transmission rate in terms of MTUs per RTT, which defines the maximum number of unacknowledged packets that TCP is allowed to maintain "in flight". Upon detection of a lost packet, this rate is halved ("multiplicative decrease"). It will be halved again approximately every RTT until the missing data gets through. Once all missing data has gotten through, the transmission rate is increased by one MTU per RTT. Every time a new acknowledgment (i.e., not a duplicate acknowledgment) is received, the rate is similarly increased (additive increase). Thus TCP can adjust its transmit rate very rapidly, i.e., it responds on the order of a RTT. By contrast, TCP-friendly rate control adjusts its rate

rather more gradually.

For simplicity, this discussion only covers the "congestion avoidance" phase of TCP congestion control. The analogy of TCP's "slow start phase" would also be needed.

TCP can easily estimate the RTT, since all its transmissions are acknowledged. In PWE3, the best way to estimate the RTT might be via the control protocol. In fact, if the control protocol is TCP-based, getting the RTT estimate from TCP might be a good option.

TCP's rate control is window-based, expressed as a number of bytes that can be in flight. PWE3's rate control would need to be rate-based. The TFRC specification [[RFC3448](#)] provides the equation for the TCP-friendly rate for a given loss rate, RTT, and MTU. Given some means of determining the loss rate, as described in [Section 5](#), the TCP friendly rate for a PW or a tunnel can be calculated at the ingress PE. However as we noted earlier, a PW may be carrying many flows, in which case the use of a single flow TCP rate would be a significant underestimate of the true fair rate application, and hence damaging to the operation of the PW.

If the congestion detection mechanism only produces an approximate result, the probability of a "false alarm" (thinking that there is congestion when there really is not) for some interval becomes significant. It would be better then to have some algorithm which

smoothes the result over several intervals. The TFRC procedures, which tend to generate a smoother and less abrupt change in the transmission rate than the AIMD procedures, may also be more appropriate in this case.

Once a PE has determined the appropriate rate at which to transmit traffic on a given PW or tunnel, it needs some means to enforce that rate via policing, shaping, or selective shutting down of PWs. There are tradeoffs to be made among these options, depending on various factors including the higher layer service that is carried. The effect of different mechanisms when the higher layer traffic is already using TCP is discussed below.

[7.1](#). Interaction with TCP

Ideally there should be no PW-specific congestion control mechanism used when the higher layer traffic is already running over TCP and is thus subject to TCP's existing congestion control. However it may be difficult to determine what the higher layer is on any given PW. Thus, interaction between PW-specific congestion control and TCP's congestion control needs to be considered.

As noted in [Section 3.2](#), a PW-specific congestion control mechanism may interact poorly with the "outer" control loop of TCP if the PW carries TCP traffic. A well-documented example of such poor interaction is a token bucket policer that drops packets outside the token bucket. TCP has difficulty finding the "bottleneck" bandwidth in such an environment and tends to overshoot, incurring heavy losses and consequent loss of throughput.

A shaper that queues packets at the PE and only injects them into the network at the appropriate rate may be a better choice, but may still interact unpredictably with the "outer control loop" of TCP flows that happen to traverse the PW. This issue warrants further study.

Another possibility is simply to shut down a PW when the rate of traffic on the PW significantly exceeds the appropriate rate that has been determined for the PW. While this might be viewed as draconian, it does ensure that any PW that is allowed to stay up will behave in a predictable manner. Note that this would also be the most likely choice of action for CBR PWs (as discussed in [Section 9](#)). Thus all

PWs would be treated alike and there would be no need to try to determine what sort of upper layer payload a PW is carrying.

8. Rate Control per Tunnel vs. per PW

Rate controls can be applied on a per-tunnel basis or on a per-PW basis. Applying them on a per-tunnel basis (and obtaining congestion feedback on a per-tunnel basis) would seem to provide the most efficient and most scalable system. Achieving fairness among the PWs then becomes a local issue for the transmitter. However, if the different PWs follow different paths through the network (e.g. because of ECMP over the tunnel), it is possible that some PWs will encounter congestion while some will not. If rate controls are applied on a per-tunnel basis, then if any PW in a tunnel is affected by congestion, all the PWs in the tunnel will be throttled. While this is sub-optimal, it is not clear that this would be a significant problem in practise, and it may still be the best trade-off.

Per-tunnel rate control also has some desirable properties if the action taken during congestion is to selectively shut down certain PWs. Since a tunnel will typically carry many PWs, it will be possible to make relatively small adjustments in the total bandwidth consumed by the tunnel by selectively shutting down or bringing up one or more PWs.

Note again the issue of estimating the correct rate. We know how many PWs there per tunnel, but we do not know how many flows there are per PW.

9. Constant Bit Rate Services

As noted above, some PW services may require a fixed rate of transmission, and it may be impossible to provide the service while throttling the transmission rate. To provide such services, the network paths must be engineered so that congestion is impossible; providing such services over the Internet is thus not very likely. In fact, as congestion control cannot be applied to such services, it may be necessary to prohibit these services from being provided in the Internet, except in the case where the payload is known to consist of TCP connections or other traffic that is congestion-controlled by the end-points. It is not clear how such a prohibition

could be enforced.

The only feasible mechanism for handling congestion affecting CBR services would appear to be to selectively turn off PWs, or channels with the PW, when congestion occurs. Clearly it is important to avoid "false alarms" in this case. It is also important to avoid bringing PWs (or channels within the PW) back up too quickly and re-introducing congestion.

The idea of controlling rate per tunnel rather than PW, discussed above, seems particularly attractive when some of the PWs are CBR. First, it provides the possibility that non-CBR PWs could be throttled before it is necessary to shut down the CBR PWs. Second, with the aggregation of multiple PWs on a single rate-controlled tunnel, it becomes possible to gradually increase or decrease the total offered load on the tunnel by selectively bringing up or shutting down PWs. As noted above, local policies at a PE could be used to determine which PWs to shut down or bring up first. Similar approaches would apply if the CBR PW offers a channelized service, with selected channels being shut down and brought up to control the total rate of the PW.

10. Managed vs Unmanaged Deployment

As discussed in [Section 2](#), there are a significant set of scenarios in which PW-specific congestion control may not be necessary. One might therefore argue that it doesn't seem to make sense to require PW-specific congestion control to be used on all PWs at all times. On the other hand, if the option of turning off PW-specific congestion control is available, there is nothing to stop a provider from turning it off in inappropriate situations. As this may contribute to congestive collapse outside the provider's own network, it may not be advisable to allow this.

The circumstance in which it is most vocally argued that congestion

control is not needed are where the PW is part of the service providers own network infrastructure. In cases where the PW is deployed in a managed, well-engineered network it is probably necessary to permit the operator to take the congestion risk upon themselves if they desire to do so by disabling any congestion

control mechanism. Ironically work in progress on the design of an MPLS Transport Profile indicates that some of these users require that an OAM is run (end-to-end, or over part of the tunnel (known as Tandem monitoring)), and when the OAM detects that the performance parameters are breached (delay, jitter or packet loss) this is used to trigger a failover to a backup path. When the backup path mechanism operates in 1 to 1 mode, this moves the congestive traffic to another network path, which is the characteristic we require. [**** add reference ****]

Further in such an environment it is likely that the PW will be used to carry many TCP equivalent flows. In these managed circumstances the operator will have to make a judgement call on a fair estimate of the number of equivalent flows that the PW represents and set the congestion parameters accordingly.

The counterpoint to a well managed network is a plug and play network, of the type typically found in a consumer environment. Whilst PWs have been designed to provide an infrastructure tool to the service provider, we cannot be sure that they will not find some consumer or similar plug and play application. In such an environment a conservative approach seems appropriate, and the default PW configuration **MUST** enable the congestion avoidance mechanism, with parameters set to the equivalent of a single TCP flow.

11. Related Work: Pre-Congestion Notification

It has been suggested that Pre-congestion Notification (PCN) [[I-D.ietf-pcn-architecture](#)] might provide a basis for addressing the PW congestion control problem. Using PCN, it would potentially be possible to determine if the level of congestion currently existing between an ingress and an egress PE was sufficiently low to safely allow a new PW to be established. PCN's pre-emption mechanisms could be used to notify a PE that one or more PWs need to be brought down, which again could be coupled with local policies to determine exactly which PWs should be shut down first. This approach certainly merits further examination, but we note that PCN is considerably further away from deployment in the Internet than ECN, and thus cannot be considered as a near-term solution to the problem of PW-induced congestion in the Internet.

[12.](#) IANA Considerations

This section may be removed by the RFC Editor.

There are no IANA actions required by this document.

[13.](#) Security Considerations

Security is a good thing (TM).

[14.](#) Conclusion

Pseudowires are sometimes used to emulate services network that carry non-TCP data flows. In these circumstances the service payload will not react to network congestion by reducing its offered load. Pseudowires SHOULD therefore reduces their network bandwidth demands in the face of significant packet loss, including if necessary completely ceasing transmission. In some service provider network environments the network operator may choose to not deploy congestion avoidance mechanisms, but they should make that choice in the full knowledge that they are avoiding a network safety valve. In an unmanaged environment a conservative approach to congestion is REQUIRED.

Since it is difficult to determine a priori the number of equivalent TCP flow that a pseudowire represents, a suitably "fair" rate of back-off cannot easily be pre-determined. In a managed environment setting the congestion parameters will require some level of informed judgement. In an unmanaged environment the equivalent TCP flow should be set to one.

The selection of the appropriate mechanism(s) to implement congestion avoidance is work in progress in the IETF PWE3 working group.

[15.](#) Informative References

[I-D.briscoe-tsvarea-fair]

Briscoe, B., "Flow Rate Fairness: Dismantling a Religion", [draft-briscoe-tsvarea-fair-02](#) (work in progress), July 2007.

[I-D.ietf-pcn-architecture]

Eardley, P., "Pre-Congestion Notification (PCN) Architecture", [draft-ietf-pcn-architecture-11](#) (work in progress), April 2009.

Internet-Draft

PWE3 Congestion

June 2009

[I-D.ietf-pwe3-fc-flow]

Roth, M., Solomon, R., and M. Tsurusawa, "Reliable Fibre Channel Transport Over MPLS Networks", [draft-ietf-pwe3-fc-flow-00](#) (work in progress), January 2009.

[RFC2001] Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", [RFC 2001](#), January 1997.

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

[RFC2581] Allman, M., Paxson, V., and W. Stevens, "TCP Congestion Control", [RFC 2581](#), April 1999.

[RFC2914] Floyd, S., "Congestion Control Principles", [BCP 41](#), [RFC 2914](#), September 2000.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", [RFC 3168](#), September 2001.

[RFC3448] Handley, M., Floyd, S., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", [RFC 3448](#), January 2003.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", [RFC 3758](#), May 2004.

[RFC3985] Bryant, S. and P. Pate, "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", [RFC 3985](#), March 2005.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", [RFC 4340](#), March 2006.

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", [RFC 4385](#), February 2006.

[RFC4447] Martini, L., Rosen, E., El-Aawar, N., Smith, T., and G.

Heron, "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", [RFC 4447](#), April 2006.

- [RFC4553] Vainshtein, A. and YJ. Stein, "Structure-Agnostic Time Division Multiplexing (TDM) over Packet (SAToP)", [RFC 4553](#), June 2006.

Bryant, et al.

Expires December 19, 2009

[Page 24]

Internet-Draft

PWE3 Congestion

June 2009

- [RFC4842] Malis, A., Pate, P., Cohen, R., and D. Zelig, "Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH) Circuit Emulation over Packet (CEP)", [RFC 4842](#), April 2007.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", [RFC 4960](#), September 2007.
- [RFC5085] Nadeau, T. and C. Pignataro, "Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires", [RFC 5085](#), December 2007.
- [RFC5086] Vainshtein, A., Sasson, I., Metz, E., Frost, T., and P. Pate, "Structure-Aware Time Division Multiplexed (TDM) Circuit Emulation Service over Packet Switched Network (CESoPSN)", [RFC 5086](#), December 2007.
- [RFC5087] Stein, Y(J)., Shashoua, R., Insler, R., and M. Anavi, "Time Division Multiplexing over IP (TDMoIP)", [RFC 5087](#), December 2007.

Authors' Addresses

Stewart Bryant
Cisco Systems, Inc.
250 Longwater
Green Park, Reading RG2 6GB
U.K.

Phone:
Fax:
Email: stbryant@cisco.com
URI:

Bruce Davie
Cisco Systems, Inc.
1414 Mass. Ave.
Boxborough, MA 01719
USA

Email: bsd@cisco.com

Bryant, et al.

Expires December 19, 2009

[Page 25]

Internet-Draft

PWE3 Congestion

June 2009

Luca Martini
Cisco Systems, Inc.
9155 East Nichols Avenue, Suite 400.
Englewood, CO 80112
USA

Email: lmartini@cisco.com

Eric Rosen
Cisco Systems, Inc.
1414 Mass. Ave.
Boxborough, MA 01719
USA

Email: erosen@cisco.com

