ConEx
Internet-Draft
Intended status: Informational
Expires: January 10, 2013

B. Briscoe
BT
M. Sridharan
Microsoft

July 09, 2012

Network Performance Isolation in Data Centres using Congestion Exposure (ConEx)

draft-briscoe-conex-data-centre-00

Abstract

This document describes how a multi-tenant data centre operator can isolate tenants from network performance degradation due to each other's usage, but without losing the multiplexing benefits of a LAN-style network where anyone can use any amount of any resource. Zero per-tenant configuration and no implementation change is required on network equipment. Instead the solution is implemented with a simple change to the hypervisor (or container) on each physical server, beneath the tenant's virtual machines. These collectively enforce a very simple distributed contract - a single network allowance that each tenant can allocate among their virtual machines. The solution is simplest and most efficient using layer-3 switches that support explicit congestion notification (ECN) and if the sending operating system supports congestion exposure (ConEx). Nonetheless, an arrangement is described so that the operator can unilaterally deploy a complete solution while operating systems are being incrementally upgraded to support ConEx.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 10, 2013.

Copyright Notice

Table of Contents

1. Introduction

A number of companies offer hosting of virtual machines on their data centre infrastructure--so-called infrastructure as a service (IaaS). A set amount of processing power, memory, storage and network are offered. Although processing power, memory and storage are relatively simple to allocate on the 'pay as you go' basis that has become common, the network is less easy to allocate given it is a naturally distributed system.

This document describes how a data centre infrastructure provider can deploy congestion policing at every ingress to the data centre network, e.g. in all the hypervisors (or containers) in a data centre that provides virtualised 'cloud' computing facilities. These bulk congestion policers pick up congestion information in the data packets traversing the network, using one of two approaches: feedback tunnels or ConEx. Then, these policers at the ingress edge have sufficient information to limit the amount of congestion any tenant can cause anywhere in the data centre. This isolates the network performance experienced by each tenant from the behaviour of all the others, without any tenant-related configuration of any of the switches.

The key to the solution is the use of congestion-bit-rate rather than bit-rate as the policing metric. _How _this works is very simple and quick to describe (Section 3 outlines the design at the start andSection 5 gives details).

However, it is much more difficult to understand _why_ this approach provides performance isolation. In particular, why it provides performance isolation across a network of links, even though there is apparently no isolation mechanism in each link. Section 4 builds up an intuition for why the approach works, and why other approaches fall down in different ways. The explanation builds as follows:

- Starting with the simple case of long-running flows focused any one bottleneck link in the network, tenants get weighted shares of the link, much like weighted round robin, but with no mechanism in any of the links;

- In the more realistic case where flows are not all long-running but a mix of short to very long, it is explained that bit-rate is not a sufficient metric for isolating performance; how often a tenant is _not_ sending is the significant factor for performance isolation, not whether bit-rate is shared equally whenever it is sending;

- Although it might seem that data volume would be a good measure of how often a tenant does not send, we then show that a tenant can send a large volume of data but hardly affect the performance of others -- by being very responsive to congestion. Using congestion-volume (congestion-bit-rate over time) in a policer encourages large data senders to give other tenants much higher performance, whereas using straight volume as an allocation metric provides no isolation at all from tenants who send the same volume but are oblivious to its effect on others (the widespread behaviour today);

- We then show that a policer based on the congestion-bit-rate metric works across a network of links treating it as a pool of capacity, whereas other approaches treat each link independently, which is why the proposed approach requires none of the configuration complexity on switches that is involved in other approaches.

The solution would also be just as applicable to isolate the network performance of different departments within the data centre of an enterprise, which could be implemented without virtualisation. However, it will be described as a multi-tenant scenario, which is the more difficult case from a security point of view.

{ToDo: Meshed, pref multipath resource pool, not unnecessarily constrained paths.}

2. Design Features

The following goals are met by the design, each of which is explained subsequently:

- Performance isolation
- No loss of LAN-like openness and multiplexing benefits
- Zero tenant-related switch configuration
- No change to existing switch implementations
- Weighted performance differentiation
- Ultra-Simple contract--per-tenant network-wide allowance
- Sender constraint, but with transferrable allowance
- Transport-agnostic

- o Extensible to wide-area and inter-data-centre interconnection

Performance Isolation with Openness of a LAN: The primary goal is to ensure that each tenant of a data centre receives a minimum assured performance from the whole network resource pool, but without losing the efficiency savings from multiplexed use of shared infrastructure (work-conserving). There is no need for partitioning or reservation of network resources.

Zero Tenant-Related Switch Configuration: Performance isolation is achieved with no per-tenant configuration of switches. All switch resources are potentially available to all tenants.

Separately, _forwarding_ isolation may (or may not) be configured to ensure one tenant cannot receive traffic from another's virtual network. However, _performance_ isolation is kept completely orthogonal, and adds nothing to the configuration complexity of the network.

No New Switch Implementation: Straightforward commodity switches (or routers) are sufficient. Bulk explicit congestion notification (ECN) is recommended, which is available in a large and growing range of layer-3 switches (a layer-3 switch does switching at layer-2, but it can use the Diffserv and ECN fields for traffic control if an IP header can be found). Once the network supports ECN, the performance isolation function is confined to the hypervisor (or container) and the operating systems on the hosts.

Weighted Performance Differentiation: A tenant gets network performance in proportion to their allowance when constrained by others, with no constraint otherwise. Importantly, the assurance is not just instantaneous, but over time. And the assurance is not just localised to each link but network-wide. This will be explained with numerical examples later.

Ultra-Simple Contract: The tenant needs to decide only two things: The peak bit-rate connecting each virtual machine to the network (as today) and an overall 'usage' allowance. This document focuses on the latter. A tenant just decides one number for her contracted allowance that can be shared over all her virtual machines (VMs). The 'usage' allowance is a measure of congestion-bit-rate, which will be explained later, but most tenants will just think of it as a number, where more is better. A tenant has no need to decide in advance which VMs will need more allowance and which less--an automated process allocates the allowance across the VMs, shifting more to those that need it most, as they use it. Therefore, performance cannot be constrained by poor choice of allocations between VMs, removing a whole dimension from

the problem that tenants face when choosing their traffic contract. The allocation process can be operated by the tenant, or provided by the data centre operator as part of an additional platform as a service (PaaS) offer.

Sender Constraint with transferrable allowance: By default, constraints are always placed on data senders, determined by the sending party's traffic contract. Nonetheless, if the receiving party (or any other party) wishes to enhance performance it can arrange this with the sender at the expense of its own allowance.

For instance, when a tenant's VM sends data to a storage facility the tenant that owns the VM consumes her allowance for enhanced sending performance. But by default when she later retrieves data from storage, the storage facility is the sender, so the storage facility consumes its allowance to determine performance in the reverse direction. Nonetheless, during the retrieval request, the storage facility can require that its sending 'costs' are covered by the receiving VM's allowance.

Transport-Agnostic: In a well-provisioned network, enforcement of performance isolation rarely introduces constraints on network behaviour. However, it continually counts how much each tenant is limiting the performance of others, and it will intervene to enforce performance isolation, but against only those customers who most persistently constrain others. This performance isolation is oblivious to flows and to the protocols and algorithms being used above the IP layer.

Interconnection: The solution is designed so that interconnected networks can ensure each is accountable for the performance degradation it contributes to in other networks. If necessary, one network has the information to intervene at its ingress to limit traffic from another network that is degrading performance. Alternatively, with the proposed protocols, networks can see sufficient information in traffic arriving at their borders to give their neighbours financial incentives to limit the traffic themselves.

The present document focuses on a single provider-scenario, but evolution to interconnection with other data centres over wide-area networks, and interconnection with access networks is briefly discussed in Section 7.2.

3. Outline Design

This section outlines the essential features of the design. Design details will be given in Section 5.

Edge policing: Traffic policing is located at the policy enforcement point where each sending host connects to the network, typically beneath the tenant's operating system in the hypervisor controlled by the infrastructure operator. In this respect, the approach has a similar arrangement to the Diffserv architecture with traffic policers forming a ring around the network [RFC2475].

Congestion policing: However, unlike Diffserv, traffic policing limits congestion-bit-rate, not bit-rate. Congestion bit-rate is the product of congestion probability and bit-rate. For instance, if the instantaneous congestion probability (cf. loss probability) across a network path were 0.02% and a tenant's maximum contracted congestion-bit-rate was 600kb/s, then the policer would allow the tenant to send at a bit-rate of up to 3Gb/s (because 3Gb/s x 0.02% = 600kb/s). The detail design section describes how congestion policers at the network ingress know the congestion that each packet will encounter in the network. {ToDo: rewrite this section to describe how a congestion policer works, not to focus just on units.}

Hose model: The congestion policer controls all traffic from a particular sender without regard to destination, similar to the Diffserv 'hose' model. {ToDo: dual policer, and multiple hoses for long-term average.}

Flow policing unnecessary: A congestion policer could be designed to focus policing on the particular data flow(s) contributing most to the excess congestion-bit-rate. However we will explain why bulk policing should be sufficient.

FIFO forwarding: Each network queue only needs a first-in first-out discipline, with no need for any priority scheduling. If scheduling by traffic class is used (for whatever reason), congestion policing can be used to isolate tenants from each other within each class. {ToDo: Say this the other way round.}

ECN marking recommended: All queues that might become congested should support bulk ECN marking, but packets that do not support ECN marking can be accommodated.

In the proposed approach, the network operator deploys capacity as usual--using previous experience to determine a reasonable contention ratio at every tier of the network. Then, the tenant contracts with

the operator for an allowance that determines the rate at which the congestion policer allows each tenant to contribute to congestion {ToDo: Dual policer}. Section 6 discusses how the operator would determine this allowance. Each VM's congestion policer limits its peak congestion-bit-rate as well as limiting the overall average per tenant.

4. Performance Isolation: Intuition

Network performance isolation traditionally meant that each user could be sure of a minimum guaranteed bit-rate. Such assurances are useful if traffic from each tenant follows relatively predictable paths and is fairly constant. If traffic demand is more dynamic and unpredictable (both over time and across paths), minimum bit-rate assurances can still be given, but they have to be very small relative to the available capacity.

This either means the shared capacity has to be greatly overprovided so that the assured level is large enough, or the assured level has to be small. The former is unnecessarily expensive; the latter doesn't really give a sufficiently useful assurance.

Another form of isolation is to guarantee that each user will get 1/N of the capacity of each link, where N is the number of active users at each link. This is fine if the number of active users (N) sharing a link is fairly predictable. However, if large numbers of tenants do not typically share any one link but at any time they all could (as in a data centre), a 1/N assurance is fairly worthless. Again, given N is typically small but could be very large, either the shared capacity has to be expensively overprovided, or the assured bit-rate has to be worthlessly small.

Both these traditional forms of isolation try to give the tenant an assurance about instantaneous bit-rate by constraining the instantaneous bit-rate of everyone else. However, there are two mistakes in this approach. The amount of capacity left for a tenant to transfer data as quickly as possible depends on:

1. the load _over time_ of everyone else

2. how much everyone else yields to the increase in _congestion_ when someone else tries to transfer data

This is why limiting congestion-bit-rate over time is the key to network performance isolation. It focuses policing only on those tenants who go fast over congested path(s) excessively and persistently over time. This keeps congestion below a design threshold everywhere so that everyone else can go fast.

Congestion policing can and will enforce a congestion response if a particular tenant sends traffic that is completely unresponsive to congestion. However, the purpose of congestion policing is not to intervene in everyone's rate control all the time. Rather it is encourage each tenant to avoid being policed -- to keep the aggregate of all their flows' responses to congestion within an overall envelope. Nonetheless, the upper bound set by the congestion policer still ensures that each tenant's minimum performance is isolated from the combined effect of everyone else.

It has not been easy to find a way to give the intuition on why congesiton policing isolates performance, particularly across a networks of links not just on a single link. The approach used in this section, is to describe the system as if everyone is using the congestion response they would be forced to use if congestion policing had to intervene. We therefore call this the boundary model of congestion control. It is a very simple congestion response, so it is much easier to understand than if we introduced all the square root terms and other complexity of New Reno TCP's response. And it means we don't have to try to decribe a mix of responses.

We cannot emphasise enough that the intention is not to make individual flows conform to this boundary response to congestion. Indeed the intention is to allow a diverse evolving mix of congestion responses, but constrained in total within a simple overall envelope.

After describing and further justifying using the a simple boundary model of congestion control, we start by considering long-running flows sharing one link. Then we will consider on-off traffic, before widening the scope from one link to a network of links and to links of different sizes. Then we will depart from the initial simplified model of congestion control and consider diverse congestion control algorithms, including no end-system response at all.

Formal analysis to back-up the intuition provided by this section will be made available in a more extensive companion technical report [conex-dc_tr].

4.1. Simple Boundary Model of Congestion Control

The boundary model of congestion control ensures a flow's bit-rate is inversely proportional to the congestion level that it detects. For instance, if congestion probability doubles, the flow's bit-rate halves. This is called a scalable congestion control because it maintains the same rate of congestion signals (marked or dropped packets) no matter how fast it goes. Examples are Relentless TCP and Scalable TCP [ToDo: add refs].

New Reno-like TCP algorithms [RFC5681] have been widely replaced by alternatives closer to this scalable ideal (e.g. Cubic TCP, Compound TCP [ToDo: add refs]), because at high rates New Reno generated congestion signals too infrequently to track available capacity fast enough [RFC3649]. More recent TCP updates (e.g. data centre TCP) are becoming closer still to the scalable ideal.

It is necessary to carefully distinguish congestion bit-rate, which is an absolute measure of the rate of congested bits vs. congestion probability, which is a relative measure of the proportion of congested bits to all bits. For instance, consider a scenario where a flow with scalable congestion control is alone in a 1Gb/s link, then another similar flow from another tenant joins it. Both will push up the congestion probability, which will push down their rates until they together fit into the link. Because the flow's rate has to halve to accomodate the new flow, congestion probability will double (lets say from 0.002% to 0.004%), by our initial assumption of a scalable congestion control. When it is alone on the link, the congestion-bit-rate of the flow is 20kb/s (=1Gb/s * 0.002%), and when it shares the link it is still 20kb/s (= 500Mb/s * 0.04%).

In summary, a congestion control can be considered scalable if the bit-rate of packets carrying congestion signals (the congestion-bit-rate) always stays the same no matter how much capacity it finds available. This ensures there will always be enough signals in a round trip time to keep the dynamics under control.

Reminder: Making individual flows conform to this boundary or scalable response to congestion is a non-goal. Although we start this explanation with this specific simple end-system congestion response, this is just to aid intuition.

4.2. Long-Running Flows

Table 1 shows various scenarios where each of five tenants has contracted for 400kb/s of congestion-bit-rate in order to share a 1Gb/s link. In order to help intuition, we start with the (unlikely) scenario where all their flows are long-running. Long-running flows will try to use all the link capacity, so for simplicity we take utilisation as a round 100%.

In the case we have just described (scenario A) neither tenant's policer is intervening at all, because both their congestion allowances are 40kb/s and each sends only one flow that contributes 20kb/s of congestion -- half the allowance.

Tenant	contracted congestion-bit-rate kb/s	scenario A # : Mb/s	scenario B # : Mb/s	scenario C # : Mb/s	scenario D # : Mb/s
(a)	40	1 : 500	5 : 250	5 : 200	5 : 250
(b)	40	1 : 500	3 : 250	3 : 200	2 : 250
(c)	40	- : ---	3 : 250	3 : 200	2 : 250
(d)	40	- : ---	2 : 250	2 : 200	1 : 125
(e)	40	- : ---	- : ---	2 : 200	1 : 125
	Congestion probability	0.004%	0.016%	0.02%	0.016%

Table 1: Bit-rates that a congestion policer allocates to five tenants sharing a 1Gb/s link with various numbers (#) of long-running flows all using 'scalable congestion control'

Scenario B shows a case where four of the tenants all send 2 or more long-running flows. Recall that each flow always contributes 20kb/s no matter how fast it goes. Therefore the policers of tenants (a-c) limit them to two flows-worth of congestion (2 x 20kb/s = 40kb/s). Tenant (d) is only asking for 2 flows, so it gets them without being policed, and all four get the same quarter share of the link.

Scenario C is similar, except the fifth tenant (e) joins in, so they all get equal 1/5 shares of the link.

In Scenario D, only tenant (a) asks for more than two flows, so (a)'s policer limits it to two flows-worth of congestion, and everyone else gets the number of flows-worth that they ask for. This means that tenants (d&e) get less than everyone else, because they asked for less than they would have been allowed. (Similarly, in Scenarios A & B, some of the tenants are inactive, so they get zero, which is also less than they could have had if they had wanted.)

With lots of long-running flows, as in scenarios B & C, congestion policing seems to emulate round robin scheduling, equalising the bit-rate of each tenant, no matter how many flows they run. By configuring different contracted allowances for each tenant, it can easily be seen that congestion policing could emulate weighted round robin (WRR), with the relative sizes of the allowances acting as the weights.

Scenario D departs from round-robin. This is deliberate, the idea being that tenants are free to take less than their share in the short term, which allows them to take more at other times, as we will

see in Section 4.4. In Scenario D, policing focuses only on the tenant (a) that is continually exceeding its contract. This policer focuses discard solely on tenant a's traffic so that it cannot cause any more congestion at the shared link (shown as 0.016% in the last row).

To summarise so far, ingress congestion policers control congestion-bit-rate in order to indirectly assure a minimum bit-rate per tenant. With lots of long-running flows, the outcome is somewhat similar to WRR, but without the need for any mechanism in each queue.

4.3. On-Off Flows

Aiming to behave like round-robin (or weighted round-robin) is only useful when all flows are infinitely long. For transfers of finite size, congestion policing isolates one tenant's performance from the behaviour of others -- unlike WRR would, as will now be explained.

Figure 1 compares two example scenarios where tenant 'b' regularly sends small files in the top chart and the same size files but more often in the bottom chart (a higher 'on-off ratio'). This is the typical behaviour of a Web server when more clients request more files at peak time. Meanwhile, in this example, tenant c's behaviour doesn't change between the two scenarios -- it sends a couple of large files, each starting at the same time in both cases.

The capacity of the link that 'b' and 'c' share is shown as the full height of the plot. The files sent by 'b' are shown as little rectangles. 'b' can go at the full bit-rate of the link when 'c' is not sending, which is represented by the tall thin rectangles labelled 'b' near the middle. We assume for simplicity that 'b' and 'c' divide up the bit-rate equally. So, when both 'b' and 'c' are sending, the 'b' rectangles are half the height (bit-rate) and twice the duration relative to when 'b' sends alone. The area of a file to be transferred stays the same, whether tall and thin or short and fat, because the area represents the size of the file (bit-rate x duration = file size). The files from 'c' look like inverted castellations, because 'c' uses half the link rate while each file from 'b' completes, then 'c' can fill the link until 'b' starts the next file. The cross-hatched areas represent idle times when no-one is sending.

For this simple scenario we ignore start-up dynamics and just focus on the rate and duration of flows that are long enough to stabilise, which is why they can be represented as simple rectangles. We will introduce the effect of flow startups later.

In the bottom case, where 'b' sends more often, the gaps between b's

transfers are smaller, so 'c' has less opportunity to use the whole line rate. This squeezes out the time it takes for 'c' to complete its file transfers (recall a file will always have the same area which represents its size). Although 'c' finishes later, it still starts the next flow at the same time. In turn, this means 'c' is sending during a greater proprotion of b's transfers, which extends b's average completion time too.

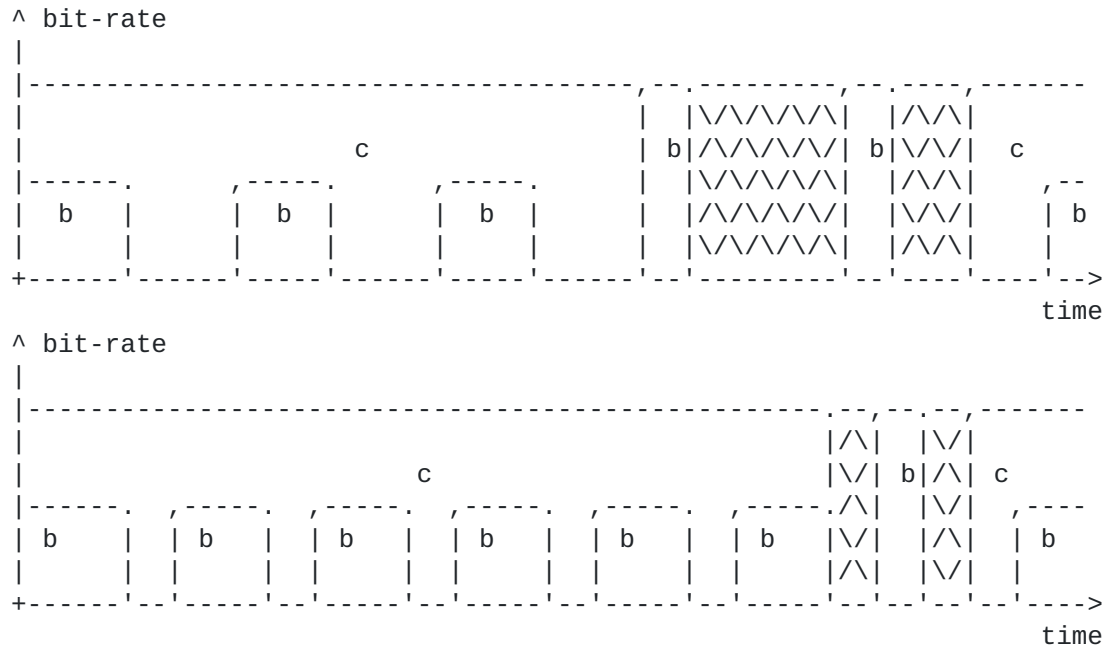

Figure 1: In the lower case, the on-off ratio of 'b' has increased, which extends all the completion times of 'c' and 'b'

Round-robin would do little if anything to isolate 'c' from the effect of 'b' sending files more often. Round-robin is designed to force 'b' and 'c' to share the capacity equally when they are both active. But in both scenarios they already share capacity equally when they are both active. The difference is in how often they are active. Round-robin and other traditional fair queuing techniques don't have any memory to sense that 'b' has been active more of the time.

In contrast, a congestion policer can tell when one tenant is sending files more frequently, by measuring the rate at which the tenant is contributing to congestion. Our aim is to show that policers will be able to isolate performance properly by using the right metric (congestion bit-rate), rather than using the wrong metric (bit-rate), which doesn't sense whether the load over time is large or small.

4.3.1. Numerical Examples Without Policing

The usefulness of the congestion bit-rate metric will now be illustrated with the numerical examples in Table 2. The scenarios illustrate what the congestion bit-rate would be without any policing or scheduling action in the network. Then this metric can be monitored and limited by a policer, to prevent one tenant from harming the performance of others.

The 2nd & 3rd columns (file-size and inter-arrival time) fully represent the behaviour of each tenant in each scenario. All the other columns merely characterise the outcome in various ways. The inter-arrival time (T) is the average time between starting one file and the next. For instance, tenant 'b' sends a 16Mb file every 200ms on average. The formula in the heading of some columns shows how the column was derived from other columns.

Scenario E is contrived so that the three tenants all offer the same load to the network, even though they send files of very different size (S). The files sent by tenant 'a' are 100 times smaller than those of tenant 'b', but 'a' sends them 100 times more often. In turn, b's files are 100 times smaller than c's, but 'b' in turn sends them 100 times more often. Graphicallyy, the scenario would look similar to Figure 1, except with three sizes of file, not just two. Scenarios E-G are designed to roughly represent various distributions of file sizes found in data centres, but still to be simple enough to facilitate intuition, even though each tenant would not normally send just one size file.

The average completion time (t) and the maximum were calculated from a fairly simple analytical model (documented in a campanion technical report [conex-dc_tr]). Using one data point as an example, it can be seen that a 1600Mb (200MB) file from tenant 'c' completes in 1905ms (about 1.9s). The files that are 100 times smaller complete 100 times more quickly on average. In fact, in this scenario with equal loads, each tenant perceives that their files are being transferred at the same rate of 840Mb/s on average (file-size divided by completion time, as shown in the apparent bit-rate column). Thus on average all three tenants perceive they are getting 84% of the 1Gb/s link on average (due to the benefit of multiplexing and utilisation being low at 240Mb/s / 1Gb/s = 24% in this case).

The completion times of the smaller files vary significantly, depending on whether a larger file transfer is proceeding at the same time. We have already seen this effect in Figure 1, where, when tenant b's files share with 'c', they take twice as long to complete as when they don't. This is why the maximum completion time is greater than the average for the small files, whereas there is

imperceptible variance for the largest files.

The final column shows how congestion bit-rate will be a useful metric to enforce performance isolation (the figures illustrate the situation before any enforcement mechanism is added). In the case of equal loads (scenario E), average congestion bit-rates are all equal. In scenarios F and G average congestion bit-rates are higher, because all tenants are placing much more load on the network over time, even though each still sends at equal rates to others when they are active together. Figure 1 illustrated a similar effect in the difference between the top and bottom scenarios.

The maximum instantaneous congestion bit-rate is nearly always 20kb/s. That is because, by definition, all the tenants are using scalable congestion controls with a constant congestion rate of 20kb/s. As we saw in Section 4.1, the congestion rate of a particular scalable congestion control is always the same, no matter how many other flows it competes with.

Once it is understood that the congestion bit-rate of one scalable flow is always 'w' and doesn't change whenever a flow is active, it becomes clear what the congestion bit-rate will be when averaged over time; it will simply be 'w' multiplied by the proportion of time that the tenant's file transfers are active. That is, w*t/T. For instance, in scenario E, on average tenant b's flows start 200ms apart, but they complete in 19ms. So they are active for 19/200 = 10% of the time (rounded). A tenant that causes a congestion bit-rate of 20kb/s for 10% of the time will have an average congestion-bit-rate of 2kb/s, as shown.

To summarise so far, no matter how many more files transfer at the same time, each scalable flow still contributes to congestion at the same rate, but it contributes for more of the time, because it squeezes out into the gap before its next flow starts.

Tenant	File size	Ave. inter-arr-ival	Ave. load	Completion time	Apparent bit-rate	Congest-ion bit-rate
	S	T	S/T	ave : max t	ave : min S/t	ave : max w*t/T
	Mb	ms	Mb/s	ms	Mb/s	kb/s
				Scenario E		
a	0.16	2	80	0.19 : 0.48	840 : 333	2 : 20
b	16	200	80	19 : 35	840 : 460	2 : 20
c	1600	20000	80	1905 : 1905	840 : 840	2 : 20
			240			
				Scenario F		
a	0.16	0.67	240	0.31 : 0.48	516 : 333	9 : 20
b	16	50	320	29 : 42	557 : 380	11 : 20
c	1600	10000	160	3636 : 3636	440 : 440	7 : 20
			720			
				Scenario G		
a	0.16	0.67	240	0.33 : 0.64	481 : 250	10 : 20
b	16	40	400	32 : 46	505 : 345	16 : 40
c	1600	10000	160	4543 : 4543	352 : 352	9 : 20
			800			

Single link of capacity 1Gb/s. Each tenant uses a scalable congestion control which contributes a congestion-bit-rate for each flow of w = 20kb/s.

Table 2: How the effect on others of various file-transfer behaviours can be measured by the resulting congestion-bit-rate

In scenario F, clients have increased the rate they request files from tenants a, b and c respectively by 3x, 4x and 2x relative to scenario E. The tenants send the same size files but 3x, 4x and 2x more often. For instance tenant 'b' is sending 16Mb files four times as often as before, and they now take longer as well -- nearly 29ms rather than 19ms -- because the other tenants are active more often too, so completion gets squeezed to later. Consequently, tenant 'b' is now sending 57% of the time, so its congestion-bit-rate is 20kb/s * 57% = 11kb/s. This is nearly 6x higher than in scenario E, reflecting both b's own increase by 4x and that this increase coincides with everyone else increasing their load.

In scenario G, tenant 'b' increases even more, to 5x the load it offered in scenario E. This results in average utilisation of 800Mb/s / 1Gb/s = 80%, compared to 72% in scenario F and only 24% in scenario E. 'b' sends the same files but 5x more often, so its load rises 5x.

Completion times rise for everyone due to the overall rise in load, but the congestion rates of 'a' and 'c' don't rise anything like as much as that of 'b', because they still leave large gaps between files. For instance, tenant 'c' completes each large file transfer in 4.5s (compared to 1.9s in scenario E), but it still only sends files every 10s. So 'c' only sends 45% of the time, which is reflected in its congestion bit-rate of 20kb/s * 45% = 9kb/s.

In contrast, on average tenant 'b' can only complete each medium-sized file transfer in 32ms (compared to 19ms in scenario E), but on average it starts sending another file after 40ms. So 'b' sends 79% of the time, which is reflected in its congestion bit-rate of 20kb/s * 79% = 16kb/s (rounded).

However, during the 45% of the time that 'c' sends a large file, b's completion time is higher than average (as shown in Figure 1). In fact, as shown in the maximum completion time column, 'b' completes in 46ms, but it starts sending a new file after 40ms, which is before the previous one has completed. Therefore, during each of c's large files, 'b' sends 46/40 = 116% of the time on average.

This actually means 'b' is overlapping two files for 16% of the time on average and sending one file for the remaining 84%. Whenever two file transfers overlap, 'b' will be causing 2 x 20kb/s = 40kb/s of congestion, which explains why tenant b in scenario G is the only case with a maximum congestion rate of 40kb/s rather than 20kb/s as in every other case. Over the duration of c's large files, 'b' would therefore cause congestion at an average rate of 20kb/s * 84% + 40kb/s * 16% = 23kb/s (or more simply 10kb/s * 116% = 23kb/s). Of course, when 'c' is not sending a large file, 'b' will contribute less to congestion, which is why its average congestion rate is 16kb/s overall, as discussed earlier.

4.3.2. Congestion Policing of On-Off Flows

Still referring to the numerical examples in Table 2, we will now discuss the effect of limiting each tenant with a congestion policer.

The network operator might have deployed congestion policers to cap each tenant's average congestion rate to 16kb/s. None of the tenants are exceeding this limit in any of the scenarios, but tenant 'b' is just shy of it in scenario G. Therefore all the tenants would be free to behave in all sorts of ways like those of scenarios E-G, but they

would be prevented from degrading the performance of the other tenants beyond the point reached by tenant 'b' in scenario G. If tenant 'b' added more load, the policer would prevent the extra load entering the network by focusing drop solely on tenant 'b', preventing the other tenants from experiencing any more congestion due to tenant 'b'. Then tenants 'a' and 'c' would be assured the (average) apparent bit-rates shown, whatever the behaviour of 'b'.

If 'a' added more load, 'c' would not suffer. Instead 'b' would go over limit and its rate would be trimmed during congestion peaks, sacrificing some of its lead to 'a'. Similarly, if 'c' added more load, 'b' would be made to sacrifice some of its performance, so that 'a' would not suffer. Further, if more tenants arrived to share the same link, the policer would force 'b' to sacrifice performance in favour of the additional tenants.

There is nothing special about a policer limit of 16kb/s. The example when discussing infinite flows used a limit of 40kb/s per tenant. And some tenants can be given higher limits than others (e.g. at an additional charge). If the operator gives out congestion limits that together add up to a higher amount but it doesn't increase the link capacity, it merely allows the tenants to apply more load (e.g. more files of the same size in the same time), but each with lower bit-rate.

{ToDo: Discuss min bit-rates}

{ToDo: discuss instantaneous limits and how they protect the minimum bit-rate of other tenants}

4.4. Weighted Congestion Controls

At high speed, congestion controls such as Cubic TCP, Data Centre TCP, Compound TCP etc all contribute to congestion at widely differing rates, which is called their 'aggressiveness' or 'weight'. So far, we have made the simplifying assumption of a scalable congestion control algorithm that contributes to congestion at a constant rate of w = 20kb/s. We now assume tenant 'c' uses a similar congestion control to before, but with different parameters in the algorithm so that its weight is still constant, but at w = 2.2kb/s.

Tenant 'b' still uses w = 20kb/s for its smaller files, so when the two compete for the 1Gb/s link, they will share it in proportion to their weights, 20:2.2 (or 90%:10%). That is, 'b' and 'c' will respectively get (20/22.2)*1Gb/s = 900Mb/s and (2.2/22.2)*1Gb/s = 100Mb/s of the 1Gb/s link. Figure 2 shows the situation before (upper) and after (lower) this change.

When the two compete, 'b' transfers each file 9/5 faster than before (900Mb/s rather than 500Mb/s), so it completes them in 5/9 of the time. 'b' still contributes congestion at the same rate of 20kb/s, but for 5/9 less time than before. Therefore, relative to before, 'b' uses up its allowance 5/9 as quickly.

Tenant 'c' contributes congestion at 2.2/22.2 of its previous rate, that is 2kb/s rather than 20kb/s. Although tenant 'b' goes faster, as each file finishes, it gets out of the way sooner, so 'c' can catch up to where it got to before after each 'b' file and should complete hardly any later than before. Tenant 'c' will probably lose some completion time because it has to accelerate and decelerate more. But, whenever it is sending a file, 'c' gains (20kb/s -2kb/s) = 18kb of allowance every second, which it can use for other transfers.

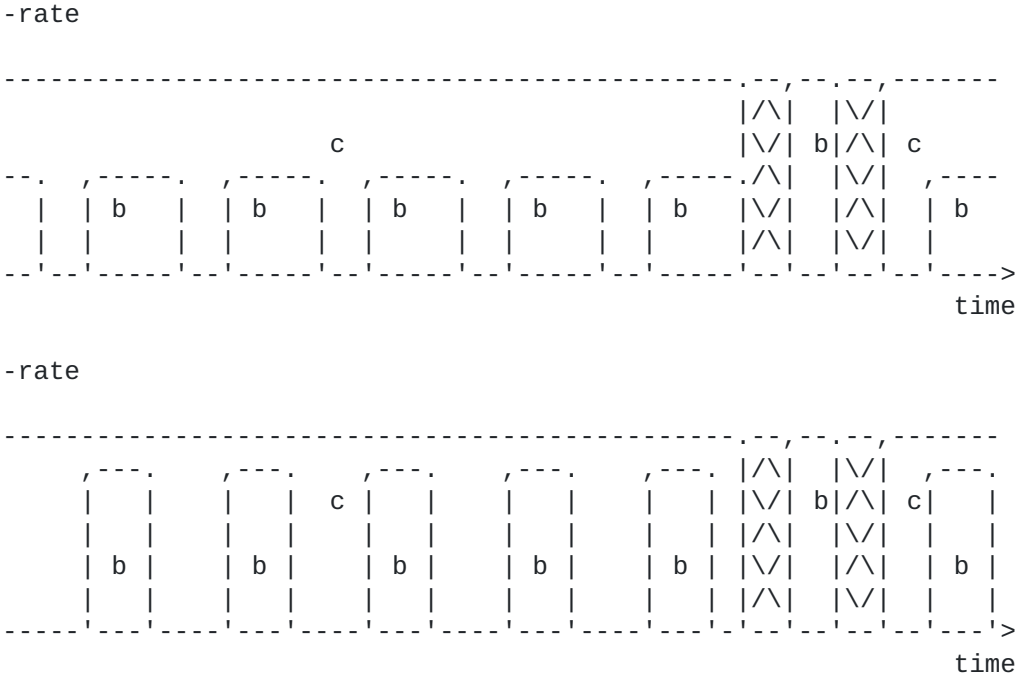

Figure 2: Weighted congestion controls with equal weights (upper) and unequal (lower)

It seems too good to be true that both tenants gain so much and lose so little by 'c' reducing its aggressiveness. The gains are unlikely to be as perfect as this simple model predicts, but we believe they will be nearly as substantial.

It might seem that everyone can keep gaining by everyone agreeing to reduce their weights, ad infinitum. However, the lower the weight,

the less signals the congestion control gets, so it starts to lose its control during dynamics. Nonetheless, congestion policing should encourage congestion control designs to keep reducing their weights, but they will have to stop when they reach the minimum necessary congestion in order to maintain sufficient control signals.

4.5. A Network of Links

So far we have only considered a single link. Congestion policing at the network edge is designed to work across a network of links, treating them all as a pool of resources, as we shall now explain. We will use the dual-homed topology shown in Figure 3 (stretching the bounds of ASCII art) as a very simple example of a pool of resources.

In this case where there are 48 servers (H1, H2, ... Hn where n=48) on the left, with on average 8 virtual machines (VMs) running on each (e.g. server n is running Vn1, Vn2, ... to Vnm where m = 8). Each server is connected by two 1Gb/s links, one to each top-of-rack switch S1 & S2. To the right of the switches, there are 6 links of 10Gb/s each, connecting onwards to customer networks or to the rest of the data centre. There is a total of 48 *2 *1Gb/s = 96Gb/s capacity between the 48 servers and the 2 switches, but there is only 6 *10Gb/s = 60Gb/s to the right of the switches. Nonetheless, data centres are often designed with some level of contention like this, because at the ToR switches a proportion of the traffic from certain hosts turns round locally towards other hosts in the same rack.

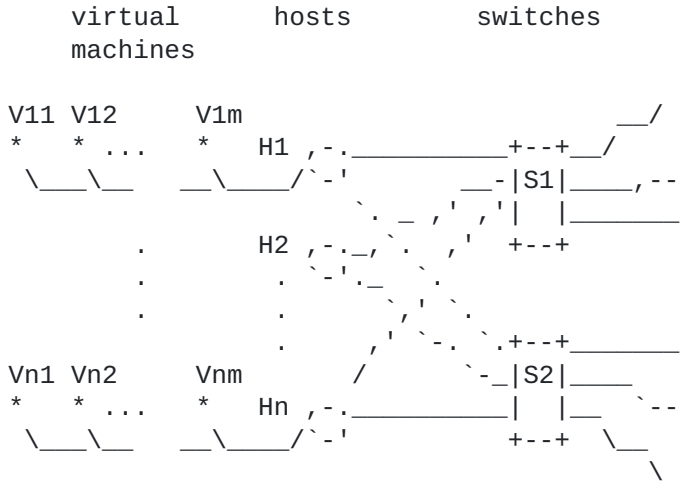

Figure 3: Dual-Homed Topology -- a Simple Resource Pool

The congestion policer proposed in this document is based on the 'hose' model, where a tenant's congestion allowance can be used for sending data over any path, including many paths at once. Therefore,

any one of the virtual machines on the left can use its allowance to contribute to congestion on any or all of the 6 links on the right (or any other link in the diagram actually, including those from the server to the switches and those turning back to other hosts).

Nonetheless, if congestion policers are to enforce performance isolation, they should stop one tenant squeezing the capacity available to another tenant who needs to use a particular bottleneck link or links. They should work whether the offending tenant is acting deliberately or merely carelessly.

The only way a tenant can become squeezed is if another tenant uses more of the bottleneck capacity, which can only happen if the other tenant sends more flows (or more aggressive flows) over that link. In the following we will call the tenant that is shifting flows 'active', and the ones already on a link 'passive'. These terms have been chosen so as not to imply one is bad and the other good -- just different.

The active tenant will increase flow completion times for all tenants (passive and active) using that bottleneck. Such an active tenant might shift flows from other paths to focus them onto one, which would not of itself use up any more congestion allowance (recall that a scalable congestion control uses up its congestion allowance at the same rate per flow whatever bit-rate it is going at Section 4.3 and therefore whatever path it is using). However, although the instantaneous rate at which the active tenant uses up its allowance won't alter, the increased completion times due to increased congestion will use up more of the active tenant's allowance over time (same rate but for more of the time). If the passive tenants are using up part of their allowances on other links, the increase in congestion will use up a relatively smaller proportion of their allowances. Once such an increase exceeds the active tenant's congestion allowance, the congestion policer will protect the passive tenants from further performance degradation.

A policer may not even have to directly intervene for tenants to be protected; load balancing may remove the problem first. Load balancing might either be provided by the network (usually just random), or some of the 'passive' tenants might themselves actively shift traffic off the increasingly congested bottleneck and onto other paths. Some of them might be using the multipath TCP protocol (MPTCP -- see experimental [RFC6356]) that would achieve this automatically, or ultimately they might shift their virtual machine to a different endpoint to circumvent the congestion hot-spot completely. Even if one passive tenant were not using MPTCP or could not shift easily, others shifting away would achieve the same outcome. Essentially, the deterrent effect of congestion policers

encourages everyone to even out congestion, shifting load away from hot spots. Then performance isolation becomes an emergent property of everyone's behaviour, due to the deterrent effect of policers, rather than always through explicit policer intervention.

{ToDo: Add numerical example}

In contrast, enforcement mechanisms based on scheduling algorithms like WRR or WFQ have to be deployed at each link, and each one works in isolation from the others. Therefore, each one doesn't know how much of other links the tenant is using. This is fine for networks with a single known bottleneck per customer (e.g. many access networks). However, in data centres there are many potential bottlenecks and each tenant generally only uses a share of a small number of them. A mechanism like WRR would not isolate anyone's performance if it gave every tenant the right to use the same share of all the links in the network, without regard to how many they were using.

The correct approach, as proposed here, is to give a tenant a share of the whole pool, not the same share of each link.

4.6. Links of Different Sizes

Congestion policing treats a Mb/s of capacity in one link as identical to a Mb/s of capacity in another link, even if the size of each link is different. For instance, consider the case where one of the three links to the right of each switch in Figure 3 were upgraded to 40Gb/s while the other two remained at 10Gb/s (perhaps to accommodate the extra traffic from a couple of the dual homed 1Gb/s servers being upgraded to dual-homed 10Gb/s).

Two congestion control algorithms running at the same rate will cause the same level of congestion probability, whatever size link they are sharing.

- If 50 equal flows share a 10Gb/s link (10Gb/s / 50 = 200Mb/s each) they will cause 0.01% congestion probability;

- If 200 equal flows share a 40Gb/s link (40Gb/s / 200 = 200Mb/s each) they will still cause 0.01% congestion probability;

This is because the congestion probability is determined by the congestion control algorithms, not by the link.

Therefore, if an average of 300 flows were spread across the above links (1x 40Gb/s and 2 x 10Gb/s), the numbers on each link would tend towards respectively 200:50:50, so that each flow would get 200Mb/s

and each link would have 0.01% congestion on it. Sometimes, there might be more flows on the bigger link, resulting in less than 200Mb/s per flow and congestion higher than 0.01%. However, whenever the congestion level was less on one link than another, congestion policing would encourage flows to balance out the congestion level across the links (as long as some flows could use congestion balancing mechanisms like MPTCP).

In summary, all the outcomes of congestion policing described so far (emulating WRR etc) apply across a pool of diverse link sizes just as much as they apply to single links.

4.7. Diverse Congestion Control Algorithms

Throughout this explanation we have assumed a scalable congestion control algorithm, which we justified Section 4.1 as the 'boundary' case if congestion policing had to intervene, which is all that is relevant when considering whether the policer can enforce performance isolation.

This performance isolation approach still works, whether or not the congestion controls in daily use by tenants fit this scalable model. A bulk congestion policer constrains the sum of all the congestion controls being used by a tenant so that they collectively remain below a large-scale envelope that is itself shaped like the sum of many scalable algorithms. Bulk congestion policers will constrain the overall congestion effect (the sum) of any mix of algorithms within it, including flows that are completely unresponsive to congestion. This is explained around Fig 3 of [CongPol].

{ToDo, summarise the relevant part of that paper here and perhaps even add ASCII art for the plot...}

{ToDo, bring in discussion of slow-start as effectively another variant of congestion control, with considerable overshoots, etc.}

The defining difference between the scalable congestion we have assumed and the congestion controls in widespread production operating systems (New Reno, Compound, Cubic, Data Centre TCP etc) is the way congestion probability decreases as flow-rate increases (for a long-running flow). With a scalable congestion control, if flow-rate doubles, congestion probability halves. Whereas, with most production congestion controls, if flow-rate doubles, congestion probability reduces to less than half. For instance, New Reno TCP reduces congestion to a quarter. The responses of Cubic and Compound are closer to the ideal scalable control than to New Reno, but they do not depart too far from TCP to ensure they can co-exist happily with New Reno.

5. Design

The design involves the following elements, all involving changes solely in the hypervisor or operating systems, not network switches:

Congestion Information at Ingress: This information needs to be trusted by the operator of the data centre infrastructure, therefore it cannot just use the feedback in the end-to-end transport (e.g. TCP SACK or ECN echo congestion experienced flags) that might anyway be encrypted. Trusted congestion feedback may be implemented in either of the following two ways:

A. either as a shim in both sending and receiving hypervisors using an edge-to-edge (host-host) tunnel, with feedback messages reporting congestion back to the sending host's hypervisor (in addition to the e2e feedback at the transport layer).

B. or in the sending operating system using the congestion exposure protocol (ConEx [ConEx-Abstract-Mech]);

Approach a) could be applied solely to traffic from operating systems that do not yet support the simpler approach b)

The host-host feedback tunnel (approach a) is easier to implement if a tunnelling overlay is already in use in the data centre. For instance, we believe it would be possible to build the necessary feedback facilities using the proposed network virtualisation approach based on generic routing encapsulation (GRE) [nvgre]. The tunnel egress would also need to be able to detect congestion. This would be simple for e2e flows with ECN enabled, because this will lead to ECN also being enabled in the outer IP header [RFC6040]. However, for non-ECN enabled flows, it is more problematic. It might be possible to add sequence numbers to the outer headers, as is done in many pseudowire technologies. However, a simpler alternative is possible in a data centre where the switches can be ECN-enabled. It would then be possible to enable ECN in the outer headers, even if the e2e transport is not ECN-capable (Not-ECT in the inner header). At the egress, if the outer is marked as 'congestion experienced', but the inner is not-ECT, the packet would have to be dropped, being the only congestion signal the e2e transport would understand. But before dropping it, the ECN marking in the outer would have served the purpose of a congestion signal to the tunnel egress. Beyond this, implementation details of approach a) are still work in progress.

If the ConEx option is used (approach b), a congestion audit function will also be required as a shim in the hypervisor (or

container) layer where data leaves the network and enters the receiving host. The ConEx option is only applicable if the guest OS at the sender has been modified to send ConEx markings. For IPv6 this protocol is defined in [conex-destopt]. The ConEx markings could be encoded in the IPv4 header by hiding them within the packet ID field as proposed in [intarea-ipv4-id-reuse].

Congestion Policing: A bulk congestion policing function would be associated with each tenant's virtual machine to police all the traffic it sends into the network. If would most likely be implemented as a shim in the hypervisor. It would be expected that various policer designs might be developed, but here we propose a simple but effective one in order to be concrete. A token bucket is filled with tokens at a constant rate that represents the tenant's congestion allowance. The bucket is drained by the size of every packet with a congestion marking, as described in [CongPol]. If approach a) were used to get "Congestion Information at the Ingress" , the bucket would be drained by congestion feedback from the tunel egress. If approach b) were used, the bucket would be drained by ConEx markings on the actual data packets being forwarded (ConEx re-inserts the e2e feedback from the transport receiver back onto packets on the forward data path).
{ToDo: Add details of congestion burst limiting}

While the data centre network operator only needs to police congestion in bulk, tenants may wish to enforce their own limits on individual users or applications, as sub-limits of their overall allowance. Given all the information used for policing is readily available to tenants in the transport layer below their sender, any such per-flow, per-user or per-application limitations can be readily applied. The tenant may operate their own fine-grained policing software, or such detailed control capabilities may be offered as part of the platform (platform as a service or PaaS) above the more general infrastructure as a service (IaaS).

Distributed Token Buckets: A customer may run virtual machines on multiple physical nodes, in which case the data centre operator would ensure that it deployed a policer in the hypervisor on each node where the customer was running a VM, at the time each VM was instantiated.The DC operator would arrange for them to collectively enforce the per-customer congestion allowance, as a distributed policer.

A function to distribute a customer's tokens to the policer associated with each of the customer's VMs would be needed. This could be similar to the distributed rate limiting of [DRL]. Alternatively, a logically centralised bucket of congestion tokens

could be used with simple 1-1 communication between it and each local token bucket in the hypervisor under each VM.

Importantly, traditional bit-rate tokens cannot simply be reassigned from one VM to another without implications on the balance of network loading (requiring operator intervention each time), whereas congestion tokens can be freely reassigned between different VMs, because a congestion token is equivalent at any place or time in a network;

As well as distribution of tokens between the VMs of a tenant, it would similarly be feasible to allow transfer of tokens between tenants, also without breaking the performance isolation properties of the system. Secure token transfer mechanisms could be built above the underlying policing design described here. Therefore the details of token transfer need not concern us here, and can be deferred to future work.

Switch/Router Support: Network switches/routers would not need any modification. However, both congestion detection by the tunnel (approach a) and ConEx audit (approach b) would be easier if switches supported ECN.

Data centre TCP might be used as well, although not essential. DCTCP requires ECN and is designed for data centres. DCTCP requires modified sender and receiver TCP algorithms as well as a more aggressive active queue management algorithm in the L3 switches. The AQM involves a step threshold at a very shallow queue length for ECN marking.

6. Parameter Setting

{ToDo: }

7. Incremental Deployment

7.1. Migration

A pre-requisite for ingress congestion policing is the function entitled "Congestion Information at Ingress " in Section 5. Tunnel feedback (approach a) is a more processing intensive change to the hypervisors, but it can be deployed unilaterally by the data centre operator in all hypervisors (or containers), without requiring support in guest operating systems.

Using ConEx markings (approach b) is only applicable if a particular guest OS supports the marking of outgoing packets with ConEx markings. But if available this is simpler and more efficient.

Both functions could be implemented in each hypervisor, and a simple filter could be installed to allow ConEx packets through into the data centre network (approach a) without going through the feedback tunnel shim, while non-ConEx packets would need to be tunnelled and to elicit tunnel feedback (approach b). This would provide an incremental deployment scenario with the best of both worlds: it would work for unmodified guest OSs, but for guest OSs with ConEx support, it would require less processing (therefore being faster) and not require the considerable overhead of a duplicate feedback channel between hypervisors (sending and forwarding a large proportion of tiny packets).

{ToDo: Note that the main reason for preferring ConEx information will be because it is designed to represent a conservative expectation of congestion, whereas tunnel feedback represents congestion only after it has happened.}

7.2. Evolution

Initially, the approach would be confined to intra-data centre traffic. With the addition of ECN support on network equipment in the WAN between data centres, it could straightforwardly be extended to inter-data centre scenarios, including across interconnected backbone networks.

Having proved the approach within and between data centres and across interconnect, more mass-market devices might be expected to turned on support for ECN feedback, and ECN might be turned on in equipment in wider networks most likely to be bottlenecks (access and backhaul).

8. Related Approaches

The Related Work section of [CongPol] provides a useful comparison of the approach proposed here against other attempts to solve similar problems.

When the hose model is used with Diffserv, capacity has to be considerably over-provisioned for all the unfortunate cases when multiple sources of traffic happen to coincide even though they are all in-contract at their respective ingress policers. Even so, every node within a Diffserv network also has to be configured to limit higher traffic classes to a maximum rate in case of really unusual traffic distributions that would starve lower priority classes. Therefore, for really important performance assurances, Diffserv is used in the 'pipe' model where the policer constrains traffic separately for each destination, and sufficient capacity is provided at each network node for the sum of all the peak contracted rates for paths crossing that node.

In contrast, the congestion policing approach is designed to give full performance assurances across a meshed network (the hose model), without having to divide a network up into pipes. If an unexpected distribution of traffic from all sources focuses on a congestion hotspot, it will increase the congestion-bit-rate seen by the policers of all sources contributing to the hot-spot. The congestion policers then focus on these sources, which in turn limits the severity of the hot-spot.

The critical improvement over Diffserv is that the ingress edges receive information about any congestion occuring in the middle, so they can limit how much congestion occurs, wherever it happens to occur. Previously Diffserv edge policers had to limit traffic generally in case it caused congestion, because they never knew whether it would (open loop control).

Congestion policing mechanisms could be used to assure the performance of one data flow (the 'pipe' model), but this would involve unnecessary complexity, given the approach works well for the 'hose' model.

Therefore, congestion policing allows capacity to be provisioned for the average case, not for the near-worst case when many unlikely cases coincide. It assures performance for all traffic using just one traffic class, whereas Diffserv only assures performance for a small proportion of traffic by partitioning it off into higher priority classes and over-provisioning relative to the traffic contracts sold for for this class.

{ToDo: Refer to Section 4 for comparison with WRR & WFQ}

Seawall {ToDo} [Seawall]

9. Security Considerations

10. IANA Considerations

This document does not require actions by IANA.

11. Conclusions

{ToDo}

12. Acknowledgments

13. Informative References

[ConEx-Abstract-Mech] Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx) Concepts and Abstract Mechanism", draft-ietf-conex-abstract-mech-03 (work in progress), October 2011.

[CongPol] Jacquet, A., Briscoe, B., and T. Moncaster, "Policing Freedom to Use the Internet Resource Pool", Proc ACM Workshop on Re-Architecting the Internet (ReArch'08) , December 2008, <http://bobbriscoe.net/projects/refb/#polfree>.

[DRL] Raghavan, B., Vishwanath, K., Ramabhadran, S., Yocum, K., and A. Snoeren, "Cloud control with distributed rate limiting", ACM SIGCOMM CCR 37(4)337--348, 2007, <http://doi.acm.org/10.1145/1282427.1282419>.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC3649] Floyd, S., "HighSpeed TCP for Large Congestion Windows", RFC 3649, December 2003.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.

[RFC6356] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, October 2011.

[Seawall] Shieh, A., Kandula, S., Greenberg, A., and C. Kim, "Seawall: Performance Isolation in Cloud Datacenter Networks", Proc 2nd USENIX Workshop on Hot Topics in Cloud Computing ,

June 2010, <http://research.microsoft.com/en-us/projects/seawall/>.

[conex-dc_tr] Briscoe, "Network Performance Isolation in Data Centres by Congestion Exposure to Edge Policers", BT Technical Report TR-DES8-2011-004, November 2011.

Work in progress

[conex-destopt] Krishnan, S., Kuehlewind, M., and C. Ucendo, "IPv6 Destination Option for Conex", draft-ietf-conex-destopt-01 (work in progress), October 2011.

[intarea-ipv4-id-reuse] Briscoe, B., "Reusing the IPv4 Identification Field in Atomic Packets", draft-briscoe-intarea-ipv4-id-reuse-01 (work in progress), March 2012.

[nvgre] Sridhavan, M., Greenberg, A., Venkataramaiah, N., Wang, Y., Duda, K., Ganga, I., Lin, G., Pearson, M., Thaler, P., and C. Tumuluri, "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre-01 (work in progress), July 2012.

Appendix A. Summary of Changes between Drafts

Detailed changes are available from http://tools.ietf.org/html/draft-briscoe-conex-data-centre

From draft-briscoe-conex-initial-deploy-02 to draft-briscoe-conex-data-centre-00:

* Split off data-centre scenario as a separate document, by popular request.

Authors' Addresses

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

Phone: +44 1473 645196
EMail: bob.briscoe@bt.com
URI: http://bobbriscoe.net/

Murari Sridharan
Microsoft
1 Microsoft Way
Redmond, WA 98052

Phone:
Fax:
EMail: muraris@microsoft.com
URI: