ConEx
Internet-Draft
Intended status: Informational
Expires: August 18, 2014

B. Briscoe
BT
M. Sridharan
Microsoft

February 14, 2014

Network Performance Isolation in Data Centres using Congestion Policing

draft-briscoe-conex-data-centre-02

Abstract

This document describes how a multi-tenant (or multi-department) 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) beneath the tenant's virtual machines on every physical server connected to the network. These collectively enforce a very simple distributed contract - a single network allowance that each tenant can allocate among their virtual machines, even if distributed around the network. The solution uses layer-3 switches that support explicit congestion notification (ECN). It is best if the sending operating system supports congestion exposure (ConEx). Nonetheless, the operator can unilaterally deploy a complete solution while operating systems are being incrementally upgraded to support ConEx and ECN.

Status of This Memo

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) or 'cloud computing'. 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 offer isolated network performance to each tenant by deploying congestion policing at every ingress to the data centre network, e.g. in all the hypervisors (or containers). The data packets pick up congestion information as they traverse the network, which is brought to the ingress using one of two approaches: feedback tunnels or ConEx (or a mix of the two). Then, these ingress congestion policers have sufficient information to limit the amount of congestion any tenant can cause anywhere in the whole meshed pool of data centre network resources. This isolates the network performance experienced by each tenant from the behaviour of all the others, without any tenant-related configuration on any of the switches.

_How _it works is very simple and quick to describe. _Why_ this approach provides performance isolation may be more difficult to grasp. In particular, why it provides performance isolation across a network of links, even though there is no isolation mechanism in each link. Essentially, rather than limiting how much traffic can go where, traffic is allowed anywhere and the policer finds out whenever and wherever any traffic causes a small amount of congestion so that it can prevent heavier congestion.

This document explains how it works, while a companion document [conex-policing] builds up an intuition for why it works. Nonetheless to make this document self-contained, brief summaries of both the 'how' and the 'why' are given in sections 3 & 4. Then Section 5 gives details of the design and Section 6 explains the aspects of the design that enable incremental deployment. Finally Section 7 introduces other attempts to solve the network performance isolation problem and why they fall down in various ways.

The solution would also be just as applicable to isolate the network performance of different departments within the private 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.

2. Features of the Solution

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
- Extensible to wide-area and inter-data-centre interconnection
- Doesn't require traffic classes, or manages traffic within each class

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 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 it can find an IP header).

Weighted Performance Differentiation: A tenant gets network performance in proportion to their allowance when constrained by others, with no constraint otherwise. Importantly, this 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 later with reference to the numerical examples in [conex-policing].

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 this contracted allowance that can be shared between all the tenant's 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.

Multi-machine: A tenant operating multiple VMs has no need to decide in advance which VMs will need more allowance and which less--an automated process can allocate 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 enhanced platform to complement the basic infrastructure (platform as a service or PaaS).

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 sending allowance.

For instance, when a VM sends data to a storage facility the tenant that owns the VM consumes as much of their allowance as necessary to achieve the desired sending performance. But by default when that tenant 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. The design of this feature is beyond the scope of this document, but the system provides all the hooks to build it at the application (or transport) layer.

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 against only those tenants who most persistently constrain others. By default, this intervention is oblivious to flows and to the protocols and algorithms being used above the IP layer. However, flow-aware or application-aware prioritisation can be built on top, either by the tenant or by the data centre operator as a complementary PaaS facility.

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 6.2.

Intra-class: The solution does not need traffic to have been classified into classes. Or if traffic is divided into classes, it manages contention for the resources of each class, independently of any scheduling between classes.

3. Outline Design

```
              virtual     hypervisors   hosts        switches
              machines

          V11 V12     V1m +--------+                         __/
          *   * ...   *   |  ____  |  H1 ,-.__________+--+__/
           \___\__   __\__|__\T1/__|____/`-'       __-|S1|____,--
                          |  /__\  |        `. _ ,' ,'|  |_______
                  .       +--------|  H2 ,-._,`.  ,'  +--+
                  .                    .  `-'._  `.
                  .       +--------+   .      `,' `.
                          |  ____  |   .     ,' `-. `.+--+_______
          Vn1 Vn2     Vnm | _\T1/_ |        /      `-_|S2|____
          *   * ...   *   |/ /__\ \|  Hn ,-.__________|  |__  `--
           \___\__   __\__/policers\____/`-'          +--+  \__
                          \  ____  /                           \
                          |\_\T2/_/|
                          |  /__\  |
                          +--------+
```

The two (or more) policers associated with tenant T1 act as one logical policer.

Figure 1: Edge Policing and the Hose Traffic Model

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 (Figure 1). In this respect, the approach has a similar arrangement to the Diffserv architecture with traffic policers forming a ring around the network [RFC2475].

(Multi-)Hose model: Each policer controls all traffic from the set of VMs associated with each tenant without regard to destination, similar to the Diffserv 'hose' model. If the tenant has VMs spread across multiple physical hosts, they are all constrained by one logical policer that feeds tokens to individual sub-policers within each hypervisor on each physical host (e.g. the two policers associated with tenant T1 in Figure 1). In other words, the network is treated as one resource pool.

Congestion policing: A congestion policer is very similar to a traditional bit-rate policer. A classifier associates each packet with the relevant tenant's meter to drain tokens from the associated token bucket, while at the same time the bucket fills with tokens at the tenant's contracted rate (Figure 2).

However, unlike a traditional policer, the tokens in a congestion policer represent congested bits (i.e. discarded or ECN-marked bits), not just any bits. So, the bits in ECN-marked packets in Figure 2 count as congested bits, while all other bits don't drain anything from the token bucket--unmarked packets are invisible to the meter. And a tenant's contracted fill rate (wi for tenant Ti in Figure 2) is only the rate of congested bits, not all bits. Then if, on average, any tenant tries to cause more congestion than their allowance, the policer will focus discards on that tenant's traffic to prevent any further increase in congestion for everyone else.

The detail design section describes how congestion policers at the network ingress know the congestion that each packet will encounter in the network, as well as how the congestion policer limits both peak and average rates of congestion.

```
                                                 ______________________
                     |          |          |    | Legend               |
                     |w1        |w2        |wi  |                      |
                     |          |          |    | [_] [_]packet stream |
                     V          V          V    |                      |
       congestion    .          .          .    | [*]    marked packet |
       token bucket| . |      | . |    __|___|  | ___                  |
                 __|___|      | . |   |  |:::|  | \ /    policer       |
                |  |:::|    __|___|   |  |:::|  | /_\                  |
                |  +---+   |  +---+   |  +---+  |                      |
   bucket depth |    :     |    :     |    :    | /\     marking meter |
   controls the |    .     |    :     |    .    | \/                   |
      policer  _V_   .     |    :     |    .    |______________________|
           ____\ /__/\___________________________             downstream
          /[*] /_\  \/ [_] |    : [_] |    : [_] \            /->network
   class-/                 |    .     |    .      \          /    /--->
   ifier/T1               _V_   .     |    .       \        /    /
  __,--.__________________\ /__/\___________________\______/____/ loss
    `--' T2   [*]  [_] [_]/_\  \/ [_] |    .  [*]   /      \    \-X--->
        \                             |    .       /        \-->
         \Ti                         _V_   :      /          \  loss
          \__________________________\ /__/\_____/            \-X--->
           [_]   [*]  [_] [*] [_]    /_\  \/ [_]
```

Figure 2: Bulk Congestion Policer Schematic

Optional Per-Flow policing: A congestion policer could be designed to focus policing on the particular data flow(s) contributing most to the excess congestion-bit-rate. However bulk per-tenant congestion policing is sufficient to protect other tenants, then each tenant can choose per-flow policing if it wants.

FIFO forwarding: If scheduling by traffic class is used in network buffers (for whatever reason), congestion policing can be used to isolate tenants from each other within each class. However, congestion policing will tend to keep queues short, therefore it is more likely that simple first-in first-out (FIFO) will be sufficient, with no need for any priority scheduling.

ECN marking recommended: All queues that might become congested should support bulk ECN marking. For any non-ECN-capable flows or packets, the solution enables ECN universally in the outer IP header of an edge-to-edge tunnel. It can use the edge-to-edge tunnel created by one of the network virtualisation overlay approaches, e.g. [nvgre, vxlan].

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 the rate at which their congestion policer will allow them to contribute to congestion. [conex-policing] explains how the operator or tenant would determine an appropriate allowance.

4. Performance Isolation: Intuition

4.1. Performance Isolation: The Problem

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, because a large number of users might all want to simulataneously share any one link, even though they rarely all use it at the same time.

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.

Round robin or fair queuing are other forms of isolation that 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. The argument is no different for the weighted forms of these algorithms: WRR & WFQ).

Both these traditional forms of isolation try to give one tenant assured instantaneous bit-rate by constraining the instantaneous bit-rate of everyone else. This approach is flawed except in the special case when the load from every tenant on every link is continuous and fairly constant. The reality is usually very different: sources are on-off and the route taken varies, so that on any one link a source is more often off than on.

> In these more realistic (non-constant) scenarios, the capacity available for any one tenant depends much more on _how often_ everyone else uses a link, not just _how much_ bit-rate everyone else would be entitled to if they did use it.

For instance, if 100 tenants are using a 1Gb/s link for 1% of the time, there is a good chance each will get the full 1Gb/s link capacity. But if just six of those tenants suddenly start using the link 50% of the time, whenever the other 94 tenants need the link, they will typically find 3 of these heavier tenants using it already. If a 1/N approach like round-robin were used, then the light tenants would suddently get 1/4 * 1Gb/s = 250Mb/s on average. Round-robin cannot claim to isolate tenants from each other if they usually get 1Gb/s but sometimes they get 250Mb/s (and only 10Mb/s guaranteed in the worst case when all 100 tenants are active).

In contrast, congestion policing is the key to network performance isolation because it focuses policing only on those tenants that 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. In this way, congestion policing takes account of highly variable loads (varying in time and varying across routes). And, if everyone's load happens to be constant, congestion policing converges on the same outcome as the traditional forms of isolation.

The other flaw in the traditional approaches to isolation, like WRR & WFQ, is that they actually prevent long-running flows from yielding to brief bursts from lighter tenants. A long-running flow can yield to brief flows and still complete nearly as soon as it would have otherwise (the brief flows complete sooner, freeing up the capacity for the longer flow sooner). However, WRR & WFQ prevent flows from even seeing the congestion signals that would allow them to co-ordinate between themselves, because they isolate each tenant completely into separate queues.

In summary, superficially, traditional approaches with separate queues sound good for isolation, but:

1. not when everyone's load is variable, so each tenant has no assurance about how many other queues there will be;

2. and not when each tenant can no longer even see the congestion signals from other tenants, so no-one's control algorithms can determine whether they would benefit most by pushing harder or yielding.

4.2. Why Congestion Policing Works

[conex-policing] explains why congestion policing works using numerical examples from a data centre and schematic traffic plots (in ASCII art). The bullets below provide a summary of that explanation, which builds from the simple case of long-running flows through a single link up to a full meshed network with on-off flows of different sizes and different behaviours:

- Starting with the simple case of long-running flows focused on a single bottleneck link, tenants get weighted shares of the link, much like weighted round robin, but with no mechanism in any of the links. This is because losses (or ECN marks) are random, so if one tenant sends twice as much bit-rate it will suffer twice as many lost bits (or ECN-marked bits). So, at least for constant long-running flows, regulating congestion-bits gives the same outcome as regulating bits;

- 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 sending (or not sending) is the significant factor for performance isolation, not whether bit-rate is shared equally whenever a source happens to be sending;

- Although it might seem that data volume would be a good measure of how often a tenant is sending, we then show why it is not. For instance, a tenant can send a large volume of data but hardly affect the performance of others -- by being more responsive to congestion. Using congestion-volume (congestion-bit-rate over time) in a policer encourages large data senders to be more responsive (to yield), giving other tenants much higher performance while hardly affecting their own performance. Whereas, using straight volume as an allocation metric provides no distinction between high volume sources that yield and high volume sources that do not yield (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.

- We also show that a congestion policer can be arranged to limit bursts of congestion from sources that focus traffic onto a single link, even where one source may consist of a large aggregate of sources.

- We show that a congestion policer rewards traffic that shifts to less congested paths (e.g. multipath TCP or virtual machine motion). This means congestion policing encourages and ultimately forces end-systems to balance their load over the whole pool of bandwidth. The network can attempt to balance the load, but bulk congestion policing is particularly designed to encourage end-systems to do the job, either at the transport layer with multipath TCP [RFC6356] or at the application layer by moving virtual machines or choosing peer virtual machines in a similar way to BitTorrent.

- We show that congestion policing works on the pool of links, irrespective of whether individual links have significantly different capacities.

- We show that a congestion policer allows a wide variety of responses to congestion (e.g. New Reno TCP, Cubic TCP, Compound TCP, Data Centre TCP and even unresponsive UDP traffic), while still encouraging and enforcing a sufficient response to congestion from all sources taken together.

- Congestion policing can and will enforce a congestion response if a tenant persistently causes excessive congestion. This ensures that each tenant's minimum performance is isolated from the combined effects of everyone else. 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 and balanced across the network.

[conex-policing] also includes a section that gives guidance on how to estimate appropriate fill rates and sizes for congestion token buckets.

5. Design

The design involves the following elements, each detailed in the following subsections:

1. Trustworthy Congestion Signals at Ingress

2. Switch/Router Support

3. Congestion Policing

4. Distributed Token Buckets

5.1. Trustworthy Congestion Signals at Ingress

```
   ,---------.                                               ,---------.
   |Transport|                                               |Transport|
   | Sender  |   .                                           |Receiver |
   |         |  /|___________________________________________|         |
   |     ,-<---------------Congestion-Feedback-Signals--<--------.     |
   |     |   |/                                              |   |     |
   |     |   |\           Transport Layer Feedback Flow      |   |     |
   |     |   | \  ___________________________________________|   |     |
   |     |   |  \|                                           |   |     |
   |     |   |   '         ,-----------.               .     |   |     |
   |     |   |_____________|           |_______________|\    |   |     |
   |     |   |    IP Layer |           |  Data Flow      \   |   |     |
   |     |   |             |(Congested)|           ,-----.\  |   |     |
   |     |   |             |  Network  |--Congestion-Signals--->-'     |
   |     |   |             |  Device   |           |     |  \|         |
   |     |   |             |           |           |Audit|  /|         |
   |     `----------->--(new)-IP-Layer-ConEx-Signals-------->|         |
   |         |             |           |           `-----'/  |         |
   |         |_____________|           |_______________  /   |         |
   |         |             |           |               |/    |         |
   `---------'             `-----------'               '     `---------'
```

Figure 3: The ConEx Protocol in the Internet Architecture

The operator of the data centre infrastructure needs to trust this information, 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 controlled by the infrastructure operator, 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]) with a ConEx audit function at the egress edge to check ConEx signals against actual congestion signals (Figure 3);

5.1.1. Tunnel Feedback vs. ConEx

The feedback tunnel approach (a) is inefficient because it duplicates end-to-end feedback and it introduces at least a round trip's delay, whereas the ConEx approach (b) is more efficient and not delayed, because ConEx packets signal a conservative estimate of congestion in the upcoming round trip. Avoiding feedback delay is important for controlling congestion from aggregated short flows. However, ConEx signals will not necessarily be supported by the sending operating system.

Therefore, given ConEx IP packets are self-identifying, the best approach is to rely on ConEx signals when present and fill in with tunnelled feedback when not, on a packet-by-packet basis.

5.1.2. ECN Recommended

Both approaches are much easier if explicit congestion notification (ECN [RFC3168]) is enabled on network switches and if all packets are ECN-capable. For non-ECN-capable packets, ECN support can be turned on in the outer of an edge-to-edge tunnel. The reasons that ECN helps in each case are:

a. Tunnel Feeback: To feed back congestion signals, the tunnel egress needs to be able to detect forward congestion signals in the first place. If the only symptom of congestion is dropped packets, the egress has to watch for gaps in the sequence space of the transport protocol, which cannot be guaranteed to be possible--the IP payload may be encrypted, or an unknown protocol, or parts of the flow may be sent over diverse paths. The tunnel ingress could add its own sequence numbers (as done by some pseudowire protocols), but it is easier to simply turn on ECN at the ingress so that the egress can detect ECN markings.

b. ConEx: The audit function needs to be able to compare ConEx signals with actual congestion. So, as before, it needs to be able to detect congestion at the egress. Therefore the same arguments for ECN apply.

5.1.3. Summary: Trustworthy Congestion Signals at Ingress

The above cases can be arranged in a 2x2 matrix, to show when edge-to-edge tunnelling is needed and what function the tunnel would need to serve:

ConEx-capable?	ECN-capable: Y	ECN-capable: N
Y	No tunnel needed	ECN-enabled tunnel
N	Tunnel Feedback	ECN-enabled tunnel + Tunnel feedback

We can now summarise the steps necessary to ensure an ingress congestion policer obtains trustworthy congestion signals:

1. Sending operating system:

    1. The sender SHOULD send ConEx-enabled and ECN-enabled packets whenever possible.

    2. If the sender uses IPv6 it can signal ConEx in a destination option header [conex-destopt].

    3. If the sender uses IPv4, it can signal ConEx markings by encoding them within the packet ID field as proposed in [ipv4-id-reuse].

2. Ingress edge:

    1. If an arriving packet is either not ConEx-capable or not ECN-capable it SHOULD be tunnelled to the appropriate egress edge in an outer IP header.

    2. A pre-existing edge-to-edge tunnel (e.g. [nvgre, vxlan]) can be used, irrespective of whether the packet is not ConEx-capable or not ECN-capable.

    3. Incoming ConEx signals MUST be copied to the outer. For an incoming IPv4 packet, this implies copying the ID field. For an incoming IPv6 packet, this implies copying the Destination Option header.

    4. In all cases, the tunnel ingress MUST use the normal mode of ECN tunnelling [RFC6040].

3. Directly after encapsulation (but not if the packet was not encapsulated):

    1. If and only if the ECN field of the outer header is not ECN-capable (Not-ECT i.e. 00) it MUST be made ECN-capable by remarking it to ECT(0), i.e. 01.

    2. If the outer ECN field carries any other value than 00, it should be left unchanged.

4. Directly before the edge egress (irrespective of whether the packet is encapsulated):

    1. If the outer IP header is ConEx-capable, it MUST be passed through a ConEx audit function

    2. If the packet is not ConEx-capable, it MUST be passed to a function that feeds back ECN marking statistics to the tunnel ingress. Such a function is also a requirement of [tunnel-cong-exp], which may be re-usable for this purpose {ToDo: to be confirmed}.

5. Egress Edge Decapsulator:

    1. Decapsulation must comply with [RFC6040]. This ensures that, a congestion experienced marking (CE or 11) on the outer will lead to the packet being dropped if the inner indicates that the endpoints will not understand ECN (i.e. the inner ECN field is Not-ECT or 00). Effectively the egress edge drops such packets on behalf of the congested upstream buffer that marked it because the packet appeared to be ECN-capable on the outside, but it is not ECN-capable on this inside. [RFC6040] was deliberately arranged like this so that it would drop such packets to give an equivalent congestion signal to the end-to-end transport.

5.2. Switch/Router Support

Network switches/routers do not need any modification. However, both congestion detection by the tunnel (approach a) and ConEx audit (approach b) are significantly easier if switches support ECN.

Once switches support ECN, Data centre TCP [DCTCP] could optionally be used (DCTCP requires ECN). It also requires modified sender and receiver TCP algorithms as well as a more aggressive configuration of the active queue management (AQM) in the L3 switches or routers.

5.3. Congestion Policing

Innovation in the design of congestion policers is expected and encouraged, but here we wlil describe one specific design to be concrete.

A bulk congestion policing function would most likely be implemented as a shim in the hypervisor. The hypervisor would create one instance of a bulk congestion policer per tenant on the physical machine, and it would ensure that all traffic sent by that tenant's VMs into the network would pass through the relevant congestion policer by associating every new virtual machine with the relevant policer.

A bulk congestion policing function has already been outlined in Section 3. To recap, it consists of a token bucket that is filled with congestion tokens at a constant rate. The bucket is drained by the size of every packet that carries a congestion marking. If the tunnel-feedback approach (a) were used, the bucket would be drained by congestion feedback from the tunnel egress, rather than markings on packets. If the ConEx approach (b) were used, the bucket would be drained by ConEx markings on the actual data packets being forwarded. A congestion policer will need to drain in response to either form of signal, because it is recommended that both approaches are used in combination.

Various more sophisticated congestion policer designs have been evaluated [CPolTrilogyExp]. In these experiments, it was found that it is better if the policer gradually increases discards as the bucket becomes empty. Also isolation between tenants is better if each tenant is policed based on the combination of two buckets, not one (Figure 4):

1. A deep bucket (that would take minutes or even hours to fill at the contracted fill-rate) that constrains the tenant's long-term average rate of congestion (wi)

2. a very shallow bucket (e.g. only two or three MTU) that is filled considerably faster than the deep bucket (c * wi), where c = ~10, which prevents a tenant storing up a large backlog of tokens then causing congestion in one large burst.

In this arrangement each marked packet drains tokens from both buckets, and the probability of policer discard is taken as the worse of the two buckets.

```
                                |         |
     Legend:                    |c*wi     |wi
     See previous figure        V         V
                                .         .
                                .       | . | deep bucket
                _ _ _ _ _ _ _ _ _ _ _ _ |___|
               |                .       |:::|
               |_ _ _ _ _ _ _ |___|     |:::|
               |      shallow +---+     +---+
   worse of the|       bucket
    two buckets|               \____   ____/
       triggers|                    \ / both buckets
      policing V                     :  drained by
              ___                    .  marked packets
   ___________\ /___________________/ \__________________
      [_] [_] /_\   [_]  [*]   [_]  \ /  [_]  [_]   [_]
```

Figure 4: Dual Congestion Token Bucket (in place of each single bucket in the previous figure)

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 within the transport layer of their own operating system. Tenants can readily apply any such per-flow, per-user or per-application limitations. 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).

5.4. Distributed Token Buckets

A customer may run virtual machines on multiple physical nodes, in which case at the time each VM is instantiated the data centre operator will deploy a congestion policer in the hypervisor on each node where the customer is running a VM.The DC operator can arrange for these congestion policers 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], which uses a gossip-like protocol to fill the sub-buckets. Alternatively, a logically centralised bucket of congestion tokens could be used. it could be replicated for reliability then there could be simple 1-1 communication between the central bucket and each local token bucket.

Importantly, congestion tokens can be freely reassigned between different VMs, because a congestion token is equivalent at any place or time in a network. In contrast, traditional bit-rate tokens cannot simply be reassigned from one VM to another without implications on the balance of network loading. This is because the parameters used for bit-rate policing depend on the topology and its capacity planning (open loop), whereas congestion policing complements the closed loop congestion avoidance system that adapts to the prevailing traffic and topology.

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, but that is beyond the current scope and therefore deferred to future work.

6. Incremental Deployment

6.1. Migration

A mechanism to bring trustworthy congestion signals to the ingress (Section 5.1) is critical to this performance isolation solution. Section 5.1.1 compares the two solutions: b) ConEx, which is efficient and it's timely enough to police short flows; and a) tunnel-feedback, which is neither. However, ConEx requires deployment in host operating systems first, while tunnel feedback can be deployed unilaterally by the data centre operator in all hypervisors (or containers), without requiring support in guest operating systems.

The section describes the steps necessary to support both approaches. This would provide an incremental deployment route with the best of both worlds: tunnel feedback could be deployed initially for unmodified guest OSs despite its weaknesses, and ConEx could gradually take over as it was deployed more widely in guest OSs. It is important not to deploy the tunnel feedback approach without checking for ConEx-capable packets, otherwise it will never be possible to migrate to ConEx. The advantages of being able to migrate to ConEx are:

- no duplicate feedback channel between hypervisors (sending and forwarding a large proportion of tiny packets), which would cause considerable packet processing overhead

- performance isolation includes the contribution to congestion from short (sub-round-trip-time) flows

6.2. Evolution

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

Once this approach becomes deployed within data centres and possibly across interconnects between data centres and enterprise LANs, the necessary support will be implemented in a wide range of equipment used in these scenarios. Similar equipment is also used in other networks (e.g. broadband access and backhaul), so that it would start to be possible for these other networks to deploy a similar approach.

7. 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 [conex-policing] for comparison with WRR & WFQ}

Seawall {ToDo} [Seawall]

8. Security Considerations

{ToDo}

9. IANA Considerations (to be removed by RFC Editor)

This document does not require actions by IANA.

10. Conclusions

{ToDo}

11. Acknowledgments

Thanks to Yu-Shun Wang for comments on some of the practicalities.

Bob Briscoe is part-funded by the European Community under its Seventh Framework Programme through the Trilogy 2 project (ICT-317756). The views expressed here are solely those of the author.

12. Informative References

[CPolTrilogyExp] Raiciu, C., Ed., "Progress on resource control", Trilogy EU 7th Framework Project ICT-216372 Deliverable 9, December 2009, <http://trilogy-project.org/publications/deliverables.html>.

[ConEx-Abstract-Mech] Mathis, M. and B. Briscoe, "Congestion

Exposure (ConEx) Concepts and Abstract Mechanism", draft-ietf-conex-abstract-mech-08 (work in progress), October 2013.

[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>.

[DCTCP] Alizadeh, M., Greenberg, A., Maltz, D., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and M. Sridharan, "Data Center TCP (DCTCP)", ACM SIGCOMM CCR 40(4)63--74, October 2010, <http://portal.acm.org/citation.cfm?id=1851192>.

[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.

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

[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-destopt] Krishnan, S., Kuehlewind, M., and C. Ucendo, "IPv6 Destination Option for ConEx", draft-ietf-conex-destopt-05 (work in progress), October 2013.

[conex-policing] Briscoe, B., "Network Performance Isolation using Congestion Policing", draft-briscoe-conex-policing-01 (work in progress), February 2014.

[ipv4-id-reuse] Briscoe, B., "Reusing the IPv4 Identification Field in Atomic Packets", draft-briscoe-intarea-ipv4-id-reuse-04 (work in progress), February 2014.

[nvgre] Sridhavan, M., Greenberg, A., Wang, Y., Garg, P., Duda, K., Venkataramaiah, N., Ganga, I., Lin, G., Pearson, M., Thaler, P., and C. Tumuluri, "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre-04 (work in progress), February 2014.

[tunnel-cong-exp] Zhu, L., Zhang, H., and X. Gong, "Tunnel Congestion Exposure", draft-zhang-tsvwg-tunnel-congestion-exposure-00 (work in progress), October 2012.

[vxlan] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", draft-mahalingam-dutt-dcops-vxlan-08 (work in progress), February 2014.

Appendix A. Summary of Changes between Drafts (to be removed by RFC Editor)

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

From briscoe-01 to briscoe-02: Added clarification about intra-class applicability. Updated references.

From briscoe-conex-data-centre-00 to briscoe-conex-data-centre-01:

* Took out text Section 4 "Performance Isolation Intuition" and Section 6. "Parameter Setting" into a separate draft [conex-policing] and instead included only a summary in these sections, referring out for details.

* Considerably updated Section 5 "Design"

* Clarifications and updates throughout, including addition of diagrams

From briscoe-conex-initial-deploy-02 to 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: