What We Talk About When We Talk About Cloud Network Performance*

* With apologies to Raymond Carver

Jeffrey C Mogul (Google)†
Lucian Popa (HP Labs)

† written while at HP Labs

Disclaimers

This work did not necessarily represent any official position of HP, when wrote it.

This work does not necessarily represent any official position of Google.

This paper was not peer-reviewed by *Computer Communication Review.*

Context: Cloud Computing

We're focussing on Infrastructure-as-a-Service (IaaS) clouds

- Other kinds of clouds might expose similar issues

Cloud computing needs fast/cheap/reliable data-center networks

- Also needs good Internet connections; we're ignoring that

Many cloud customers need performance guarantees

- To support mission-critical applications ...
- ... with predictable results and costs

What's the problem?

Studies have shown huge variations in application performance ...

- ... which are often caused by variable network performance
- See "Towards Predictable Datacenter Networks," Ballani *et al.*, SIGCOMM 2011

No network performance guarantees ⇒ no application predictability

So: cloud customers want network performance guarantees

- (or at least, they should want these)

The network is a globally-shared system of multiple individual resources, which makes guarantees harder than for CPU/RAM/disk

- Best-efforts sharing is not going to be good enough
- Hardware trends are unlikely to save us

Cloud network performance guarantees: That's simple, right?

"Just give me enough bandwidth at a good price"

But:
- Where, when, and how do we measure bandwidth?
- Is bandwidth the only important metric?
- How do we set the price?
- How do we actually make this work in practice?

There are lots of ways to approach these questions ⇒
- so not much agreement on how to structure guarantees
- and it's hard to compare research results
- or to guide research towards useful designs

This talk: an attempt to focus thinking about "Cloud Network Performance"

How we should think about:

- Cloud bandwidth + latency guarantees, and why they matter
- What has already been done
- Unsolved problems and future directions

What kinds of network performance guarantees make sense:

- for cloud customers?
- for cloud providers?

between the VMs of a specific tenant

Out of scope for this talk:

- Performance to/from external (Internet) endpoints
- performance between VMs of different tenants
- performance between "availability zones" (AZs) or "regions"

all of which are important and challenging problems

Outline of the talk

- What kinds of properties do we want to guarantee?
  - Between which end-points?
  - For what time periods?
- The interaction between guarantees and pricing
- Implementation issues
- A taxonomy of some previous work

Outline of the talk

- What kinds of properties do we want to guarantee?
  - Between which end-points?
  - For what time periods?
- The interaction between guarantees and pricing
- Implementation issues
- ~~A taxonomy of some previous work~~ (see the paper for this)

What properties do we want to guarantee?

What kinds of properties do we want for cloud-network performance guarantees?

Customer's point of view:

- Predictable, high bandwidth
- Predictable, low latency
- Predictable, low loss
- Predictable, low cost
- Simple, flexible interface

Provider's point of view:

- Happy customers
- Scalable to lots of VMs
- Efficient implementation
- High utilization of resources
- Predictable profit margins
- Simple/automated management

What kinds of properties do we want for cloud-network performance guarantees?

Customer's point of view:
- Predictable, high bandwidth
- Predictable, low latency
- Predictable, low loss
- Predictable, low cost
- Simple, flexible interface

Provider's point of view:
- Happy customers
- Scalable to lots of VMs
- Efficient implementation
- High utilization of resources
- Predictable profit margins
- Simple/automated management

Notice what isn't on this slide?
- **Fair allocation**
- **Work-conserving allocation**

I'll get to those topics, later on.

OK, so what does "guaranteed bandwidth" mean, anyway?

"Guaranteed bandwidth": not as simple as it might sound:

- Between what endpoints do we measure bandwidth?
- Over what period do we measure it?
- When is the guarantee violated?

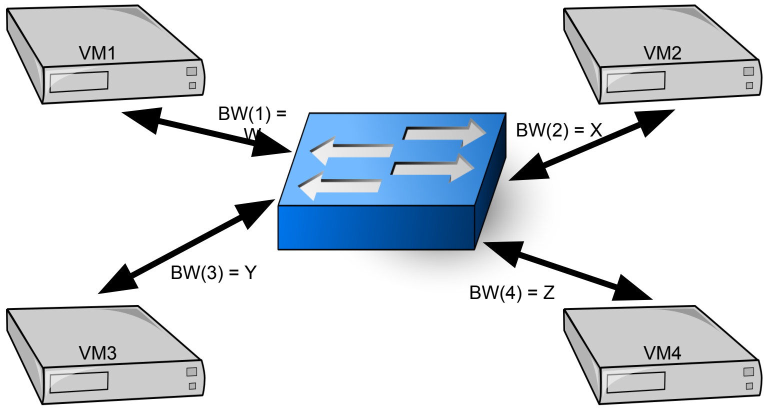
Between which endpoints?
Two popular models (there are others, but not enough time to talk about them)

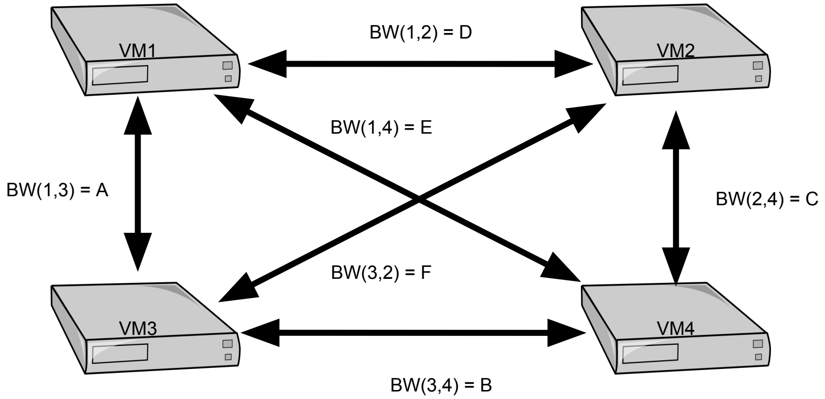
"Hose Model"

- VMs all connected via one abstract "big switch"
- Bandwidth guaranteed between switch and VMs

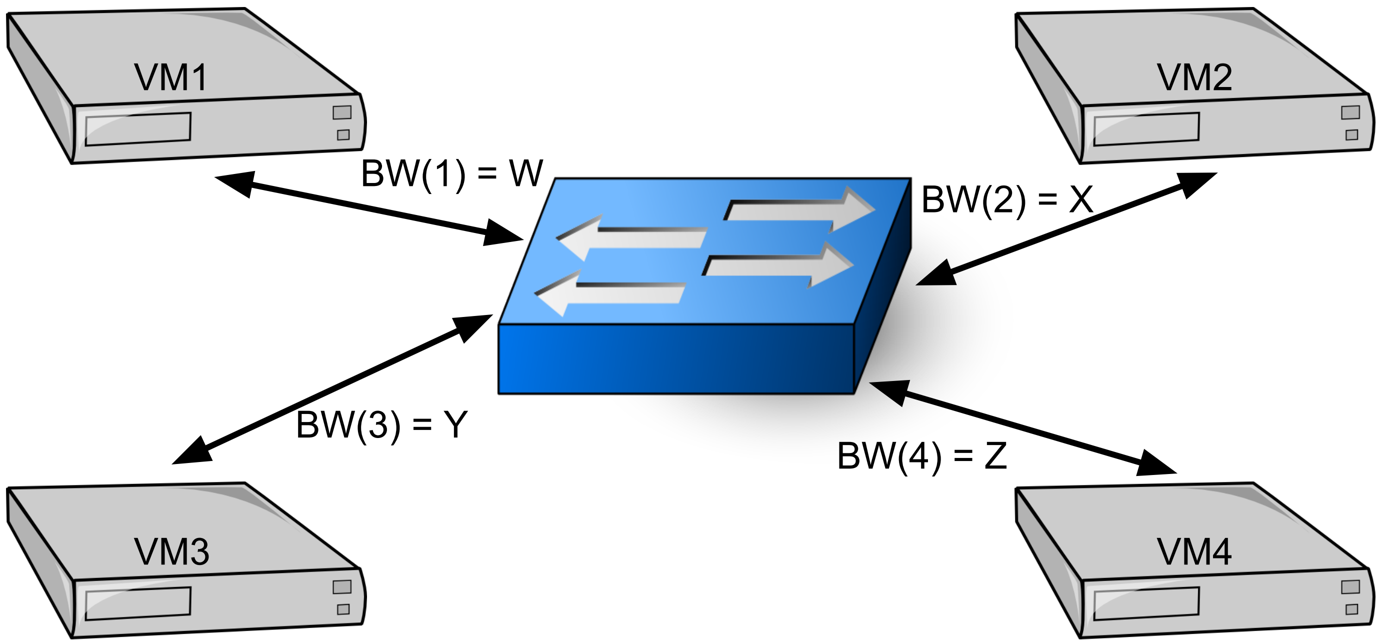
"Pipe Model"

- Bandwidth guarantees between pairs of VMs





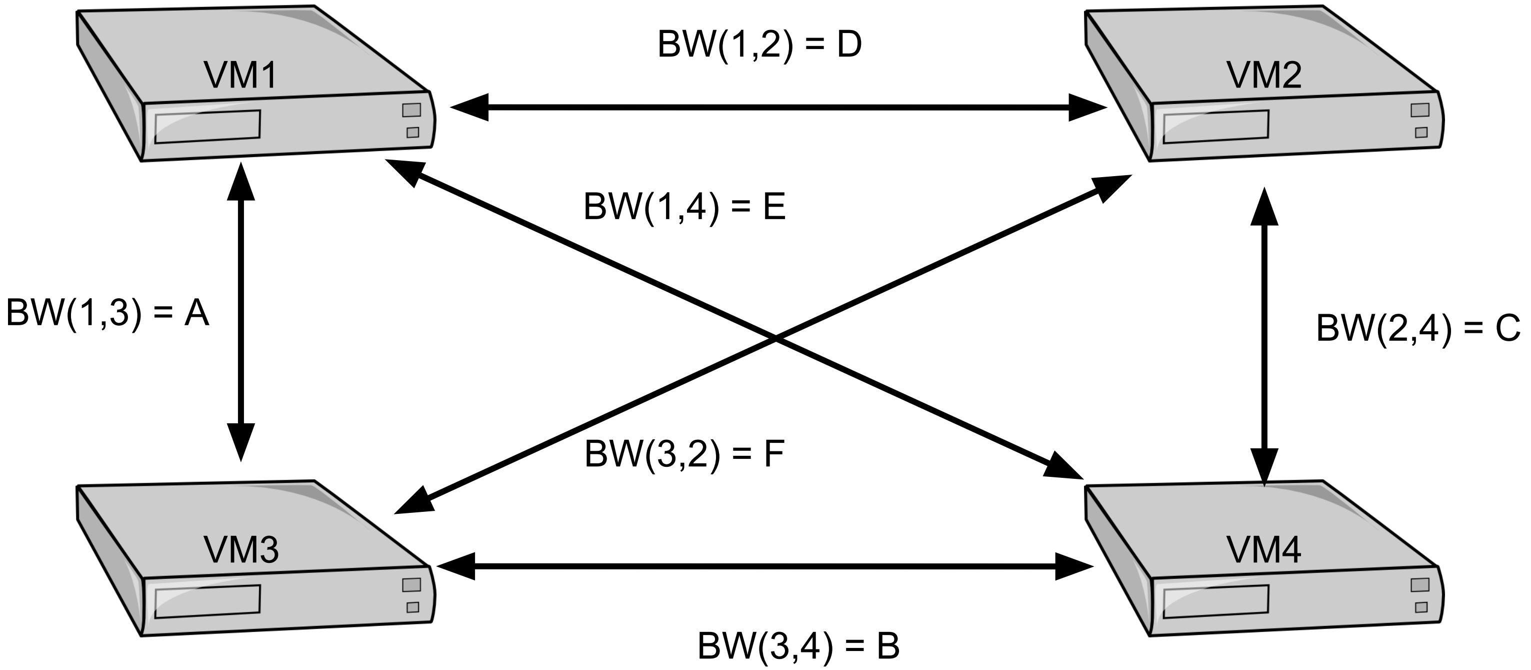
Hose model



Pros & cons:

- + Simple abstraction, matches "real world" provisioning
- + Easy to specify: one value/VM (or 2, for bidirectional)
- − May force over-provisioning of underlying real resources
  - E.g., for certain 3-tier services (see "CloudMirror," HotCloud '13)

Pipe model



Pros & cons:

- + Captures actual inter-VM requirements
    - Effectively, the inter-VM traffic matrix
- - Requires O(N^2) parameters (vs. O(N) for hose model)

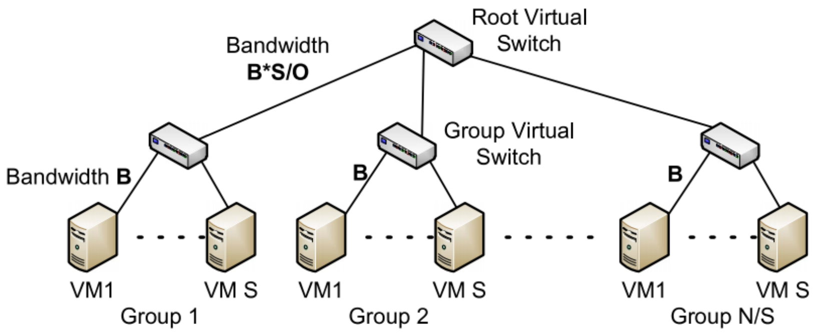
Variations on the hose model

Hierarchical hose model

- E.g., "Virtual Oversubscribed Cluster" (Oktopus)

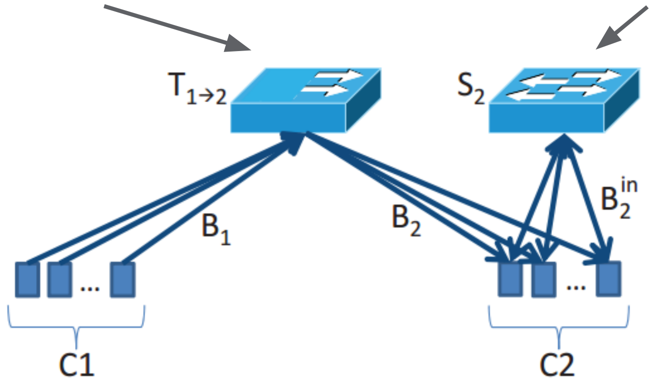
Tiered graph

- E.g., "Tenant Application Graph" (CloudMirror)



Request <N, S, B, O>
N VMs in groups of size S, Oversubscription factor O
Group switch bandwidth = S*B, Root switch bandwidth = N*B/O

Inter-tier virtual switch Intra-tier virtual switch



Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. *Towards predictable datacenter networks.* In *Proc. SIGCOMM 2011*

Jeongkeun Lee, Myungjin Lee, Lucian Popa, Yoshio Turner, Sujata Banerjee, Puneet Sharma and Bryan Stephenson. *CloudMirror: Application-Aware Bandwidth Reservations in the Cloud.* In *Proc. USENIX HotCloud*, June 2013

Things change

Bandwidth demands aren't static

- Workloads vary over time
  - Predictably, over long periods -- e.g., daily/weekly cycles
  - Predictably, over short periods -- e.g., phases of MapReduce jobs
  - Unpredictably -- e.g., flash crowds
  - Cloud computing is often sold as a way to easily "flex" capacity
- Typically, cloud customers can add/remove VMs fairly easily
- How do bandwidth guarantees handle time-varying needs?

Some possible approaches:

- Proteus (SIGCOMM '12) suggests scheduling MapReduce jobs so as to interleave their high-bandwidth phases
- CloudMirror (HotCloud '13) adapts to changes in #of VMs at each tier
- Cicada (unpub.) uses ML to predict future bandwidth needs

What are we measuring?

We could measure/guarantee:

- Mean bandwidth over a given period P
- Peak bandwidth
  - e.g., measure over short intervals of length Δ, and guarantee that the worst-case result over period P is bounded (Δ << P) 99.99% of time
- Latency
- "Tail latency" (e.g., 99.99%ile latency)
- Loss rate

Different applications will require different approaches

- Batch jobs: mean bandwidth is probably OK
- Interactive applications: need bounds on tail latency ...
- ... or perhaps flow completion time?
- When guaranteeing latency is hard, peak-bandwidth guarantees may be the best we can do.

What properties do we want to guarantee?
Summing up

One size does not fit all:

- Mean bandwidth vs. peak bandwidth vs. latency
- 100% guarantees vs. probabilistic guarantees
- Models of communication patterns (pipe/hose/VOC/etc.)
    - Assertion: the pipe model is the most expressive
    - … but customers will need automation to ask for the right pipes
- How to handle time-varying needs
- ... and how to support all of these without wasting resources

Cloud providers and customers will need to agree on what matters

How do guarantees interact with pricing?

Cloud networking is a business

Money flows from customers (tenants) to providers

- Accounting for network use within a cloud data-center is quite different from the accounting arrangements on the Internet
- Allocation of resources can follow the money more closely

Be careful about transferring ideas about desirable properties of a network from our experience with the public Internet, especially:

- Fairness
- Work-conservation

Does fairness matter?

Fairness is **one** way to allocate resources among competing needs

- in the absence of other allocation mechanisms, it's not bad
- between the flows of one application, fairness might be important
  - E.g.: to prevent a MapReduce job from suffering from "stragglers"

Assertion: fairness **between** cloud tenants **does not matter**

- If you get the BW guarantee you pay for, why do you care?
- "Price discrimination" is a fact of life
  - You paid more for your airplane ticket than the person next to you
  - Coke might have to pay more than Pepsi on the same cloud
    - All Coke really cares is that Pepsi can't undermine their guarantee

FairCloud (SIGCOMM '12) showed you can't simultaneously have both fairness and minimum bandwidth guarantees

- What we want most is guarantees, for predictable performance/costs

What about "spare" bandwidth?

We typically like work-conserving systems

- Otherwise, some capacity goes to waste
- but these are not fully predictable -- what you get tomorrow might not be what you got today.

Cloud providers do not want to give away spare BW for free

- Otherwise they risk training their customers to expect this

but they would like to be able to sell spare bandwidth

Ideal (?) solution:

- Build a work-conserving system
- Bill customers for spare BW, at a reduced rate
  - possibly with segregated traffic classes visible to the customer
- Occasionally remind customers that un-guaranteed = unreliable

Pricing of multiple resources

Ballani *et al.* (HotNets '11) point out problems with simplistic pricing

- VM time can be wasted waiting for slow network paths
- Simply adding a bandwidth cost to the VM cost is a bad idea
  - Sneaky provider could stall network to increase VM hours billed per job

They propose "Dominant Resource Pricing":

- You pay only for the resource (CPU or net) you're using more of
  - If network is the bottleneck, don't pay for idle CPU time
  - If CPU is the bottleneck, don't pay for idle network bandwidth

Implementation Issues

What is hard to implement? *

Hose model guarantees in an oversubscribed network

- Because the "one big virtual switch" isn't non-blocking
- VOC does this with 2-level hose model + careful placement

Scalable pipe-model guarantees

- Switch-based rate limiters: not enough of them
- Hypervisor software rate limiters: don't scale easily
    - Issues with locking, high-precision timing, TSO, etc.

Hose model guarantees + fair work-conserving allocation

- This seems to require pipe-style rate limiters
- plus some sort of feedback mechanism
- EyeQ and ElasticCloud demonstrate promise

* not an exhaustive list

What’s next?

Where are we headed?

The evidence suggests that we have not entirely converged

- Few cloud providers offer bandwidth guarantees
- Customers may not really understand how to request/use them
- Efficient, scalable implementation is still an open problem
- Technical solutions cannot be independent of pricing models

Crystal ball:

- We'll start to see some clouds with bandwidth guarantees ...
- ... but they won't all guarantee the same thing
- ... and not all customers will want the same kinds of guarantees
- Ultimately, providers will converge on a small menu of options

Promising areas for future research

Can clouds support guarantees for network latency?

- Especially for (say) 99.9%ile latency

How do we deal with changing workloads?

- Including "flexing" of the number of active VMs
- on various timescales

both in terms of guarantee structures and for implementation

How can academic researchers cope with these challenges?

- Researchers lack good data on how cloud networks get used
  - E.g., Cloud providers are unlikely to release packet traces
- It's hard to try things out at scale
- True infrastructure costs (HW/SW/power/people) are secret

Thanks!

Backup stuff

A taxonomy of <u>some</u> previous work

An ^Oversimplified taxonomy of <u>some</u> previous work

Hose-model approaches

Simple hose:

- Gatekeeper (assumes non-congested core)
  - Even with an “infinite core,” this is still a non-trivial problem
- SecondNet (for its “type-1 service”)

Work-conserving simple hose:

- EyeQ
- ElasticSwitch

Structured hose:

- Oktopus/VOC
- CloudMirror

Probably these could be made work-conserving, too.

Pipe-model approaches

User-specified pipe model:

- SecondNet (for its “type-0” service)
- Seawall (provides fairness between pipes, not reservations)

Measurement-based pipe models:

- Choreo (place tasks on VMs based on available pipe BWs)
- Cicada (predict future pipe BWs based on past behavior)

Time-interleaved pipes:

- Proteus (based on profiles and careful scheduling)

Tenant-aggregate approaches

- Distributed Rate Limiting
  - tries to limit total bandwidth for all VMs of a tenant
- ConEx (IETF Internet-Draft)
  - tries to equalize packet-loss rates across tenants