What We Talk About When We Talk About Cloud Network Performance

Jeffrey C. Mogul
HP Labs
jeff.mogul@hp.com

Lucian Popa
HP Labs
lucian.popa@hp.com

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

Abstract

Infrastructure-as-a-Service ("Cloud") data-centers intrinsically depend on high-performance networks to connect servers within the data-center and to the rest of the world. Cloud providers typically offer different service levels, and associated prices, for different sizes of virtual machine, memory, and disk storage. However, while all cloud providers provide network connectivity to tenant VMs, they seldom make any promises about network performance, and so cloud tenants suffer from highly-variable, unpredictable network performance.

Many cloud customers do want to be able to rely on network performance guarantees, and many cloud providers would like to offer (and charge for) these guarantees. But nobody really agrees on how to define these guarantees, and it turns out to be challenging to define "network performance" in a way that is useful to both customers and providers. We attempt to bring some clarity to this question.

Categories and Subject Descriptors

C.2.1 [**Computer-communication Networks**]: Network Architecture and Design

General Terms

Performance

Keywords

cloud networks, performance guarantees

1. INTRODUCTION

Without high-performance networks, there would be no such thing as cloud computing. Cloud data centers intrinsically depend on high-performance networks to connect servers within the data-center and to the rest of the world.

In particular, Infrastructure-as-a-Service (IaaS)[1] providers typically offer different service levels, and associated prices, for different sizes of virtual machine, memory, and disk storage. However, while all cloud providers provide network connectivity to tenant VMs, they seldom make any promises about network performance – bandwidth, latency, and loss – and so cloud tenants suffer from highly-variable, unpredictable network performance.

Many cloud customers do want to be able to rely on network performance guarantees, and many cloud providers would like to offer (and charge for) these guarantees. But most cloud providers offer no guarantee for network performance. This leads to application performance problems; for example, Ballani *et al.* [3] summarize several measurement studies showing huge (order-of-magnitude) variations in intra-cloud bandwidth, and describe how performance variability leads to "poor and unpredictable application performance." Some of these studies have also shown that no-guarantee cloud networks also suffer from high and highly variable latency, and high loss rates.

But nobody really agrees on how to define these guarantees, and it turns out to be challenging to define "network performance" in a way that is both useful to customers and plausibly implementable by providers.

In fact, there is plenty of prior work (see Section 2) aimed at various specific approaches to providing cloud network guarantees. Almost all of the prior work, however, proposes one approach and compares it against a no-guarantee network; they have not tried to explore where they are in the space of possible options.

We hope, in this paper, to bring clarity to this question: what kinds of network performance guarantees make sense, for both cloud customers and cloud providers?

To simplify the discussion, in this short paper, we do not address the challenges of guaranteeing performance between tenant VMs and the Internet, between different tenants, or across geographical regions or "availability zones" (AZs). We focus on guarantees for communication between the VMs of one tenant in one AZ.

2. PROPOSED APPROACHES

We start by reviewing some of the proposals for providing performance guarantees within cloud networks. These summaries are necessarily brief and hence may be oversimplified.

No multi-tenancy: Some cloud providers offer the equivalent of non-shared networks (e.g., Amazon's Cluster Compute "placement groups" offer 10 Gbps full bisection bandwidth between VMs[2]). However, these offerings typically support little or no resource multiplexing, and hence cost a lot more.

Distributed Rate Limiting: Raghavan *et al.* [13] described DRL, which enforces a global limit on the sum of the traffic generated by all of the sites for a given tenant (if the tenant has VMs in multiple sites). The DRL approach could perhaps be extended to limiting inter-VM flows within a site. One focus of DRL was to support flat-rate, rather than usage-based pricing, and to allow a given service some flexibility in how it allocates the total bandwidth among its VMs.

NetShare: Lam *et al.* [11] described NetShare, which assigns a specific weight to each tenant, e.g., based on payment, and then allocates congested links between tenants in proportion to weights.

[1]We will use the term "cloud" informally instead of the uglier, but more precise, "Iaas." Much of this paper might be applicable to other cloud forms – PaaS, SaaS, etc. – but we focus on IaaS.

[2]`https://aws.amazon.com/hpc-applications/`

Since the weights are network-wide, but congestion is link-by-link and can involve varying numbers of VMs, NetShare does not provide straightforward bandwidth guarantees.

SecondNet: Guo *et al.* [9] proposed the "Virtual Data Center" (VDC) abstraction for their SecondNet architecture, with three service models: "type-0" service guarantees bandwidth between pairs of VMs[3]; "type-1" service provides ingress/egress guarantees for a specific endpoints; other traffic is treated as best-effort. An endpoint can be an entire VM, or a TCP/UDP port on a VM.

Seawall: Shieh *et al.* proposed Seawall, which "achieves max-min fairness across [source] tenant VMs by sending traffic through congestion-controlled, hypervisor-to-hypervisor tunnels" [15]. Seawall focusses on scaling to large numbers of tenants and VMs, and on efficient use of bandwidth (by avoiding fixed resource reservations). Seawall provides no performance predictability, however.

The Seawall paper also attempted to "[explore] the design space for achieving performance isolation between tenants," comparing a variety of existing approaches (e.g., 802.1p/1qaz, QCN, MPLS, RSVP, etc.). That paper did not carefully explore the space of possible policies – the forms that performance guarantees might take – and instead focussed on link-level fairness between VM-to-VM flow aggregates.

Topology switching: Webb *et al.* [17] observed that different applications ("tasks") have different kinds of requirements for network performance. Their proposed design lets tasks request specific properties of the physical topologies that underlie their virtual networks. Properties are associated with paths between VMs, and include bandwidth, resilience (in terms of the number of link-level cuts the path can tolerate), and isolation.

They express the latter as k-isolation, where k is the number of other tasks sharing a link. From the point of view of task performance, the meaning of $k = 0$ is clear; it is less clear what (if anything) is guaranteed when $k > 0$. Also, this approach might not scale beyond a small number of tenants, because it might be impossible to find a satisfactory mapping of virtual paths to real links.

Gatekeeper: Rodrigues *et al.* [14] described Gatekeeper, which focusses on providing predictable performance. Gatekeeper attempts to provide each tenant with the illusion of a single, non-blocking switch connecting all of its VMs. Each VM is given guaranteed bandwidth, specified per-VM, into and out of this switch.[4] Optionally, a VM's maximum bandwidth can be set larger than its guarantee, to allow use of otherwise underutilized bandwidth. This allows the provider to trade off between efficiency and predictability, by adjusting either or both of the minimum and maximum bandwidths.

Gatekeeper uses hypervisor-based rate limits, and a feedback-based mechanism to prevent remote VMs from sending more traffic to a VM than it is allowed to receive. This is especially important to prevent problems caused by non-TCP-friendly flows. Gatekeeper attempts to prevent congestion on access links, but it assumes that the core is fully-provisioned (i.e., core links are never congested), which might be optimistic.

Oktopus: Ballani *et al.* [3] also focus on performance predictability. They start with "Virtual Cluster" abstraction, similar to Gatekeeper, in which all VMs of a tenant appear to be a cluster connected to a single switch, with links of capacity B. They then extend this to the "Virtual Oversubscribed Cluster" (VOC) model, in which clusters with switch-to-VM bandwidth B are interconnected with a fixed oversubscription factor O. VOC recognizes that applications have structure, and respects application structures that do not need, and do not wish to pay for, full bandwidth among all pairs of VMs.

Oktopus is an implementation of the VOC model, using hypervisor-based rate limiters together with a detailed algorithm for placing VMs in order to meet the requested bandwidth demands (or to reject requests that cannot be met.) Because Oktopus relies on VM placement, it might introduce some delays associated with moving VMs as workloads change.

FairCloud: Popa *et al.* [12] analyzed a set of goals in the design space for sharing cloud networks, and the inherent tradeoffs between these goals. The authors argued that cloud networks should provide, in addition to minimum bandwidth guarantees, high utilization of network links in the presence of unsatisfied demands, and *network proportionality*: division of bandwidth among tenants in proportion to the number of a tenant's VMs. They showed that one cannot simultaneously provide both bandwidth guarantees and network proportionality. The paper proposes mechanisms that can achieve different subsets of the desired properties: link-level proportionality, restricted forms of network proportionality, and minimum bandwidth guarantees over tree-structured networks. However, most of the mechanisms proposed in the FairCloud paper require switch hardware upgrades.

ConEx: Briscoe and Sridharan [4] have proposed a model which focusses on the amount of congestion that each tenant imposes on the network, based on the intuition that a tenant's network load that does not create congestion does not interfere with other tenants. Tenants purchase "congestion-bit-rates", which represent congestion allowances for that tenant. ConEx uses ECN support from switches to measure congestion, and hypervisor rate-limiters to ensure tenants do not cause more congestion than they have purchased. The ConEx proposal, however, might not match what cloud tenants want: it requires them to express their requests in terms of congestion allowances instead of bandwidth guarantees, and it could be hard to provide predictable behavior.

2.1 Location-related approaches

Although most proposals attempt to provide quantified guarantees, some researchers have instead focussed on the implications of VM location.

Location independence: Ballani *et al.* [2] do not directly address the problem of providing network performance guarantees, but instead focus on how to ensure that "tenant costs, in spite of (possibly) variable network performance, do not depend on the location of their [VMs]." They emphasize the importance not just of providing good network performance, but of considering how this affects tenants' total costs for both VMs and network bandwidth.

They propose a model called "Dominant Resource Pricing" (DRP), which avoids the problem of a tenant, whose VM is blocked by a slow network, being charged a lot for un-usable VM hours, even if the network charges remain low.

Choreo: Instead of assuming that the provider offers a specific bandwidth guarantee, LaCurts *et al.* [10] use an application-level approach. They profile the application to find its network demands; measure the cloud provider's network to discover the bandwidth available between pairs of VMs; and then the application places its own workload to optimize its predicted performance. Like VOC, Choreo supports the communication structure of the application; unlike VOC, Choreo does not try to express that structure explicitly to the provider.

Choreo expects the provider to allow applications to place VMs

[3]Expressing guarantees in terms of bandwidth between pairs of VMs is known as the "pipe model"; see Section 3.3.

[4]This kind of guarantee is known as the "hose model"; see Section 3.3.

on specific servers, and it expects that path bandwidths will remain fairly stable. We are not sure if these assumptions generalize.

2.2 Time-varying approaches

Most research proposals have ignored the possibility that an application's network demands can vary over time, in a predictable way. For a time-varying workload, no fixed bandwidth guarantee is optimal. A static guarantee high enough to satisfy the peak workload could cause the tenant to pay more than necessary, and it can prevent the provider from selling the unused (but still committed) bandwidth to a different tenant.

Proteus: Xie *et al.* [18] profiled several data-intensive MapReduce-style applications, and showed that many such applications exhibit predictable time-varying behavior at timescales on the order of tens of seconds. They proposed a Temporally-Interleaved Virtual Cluster (TIVC) abstraction, similar to VOC but which allows the provider to admit multiple jobs that effectively time-share the same bandwidth. Their Proteus system profiles a running application to derive a model, uses the model to offer the tenant several choices of cost/performance tradeoffs for future executions, and then runs an allocation algorithm to place VMs so as to maximize throughput.

Cicada: In our own work in progress, we started from several observations: (1) that many workloads vary predictably over time, although we focus on diurnal variations rather than the much shorter timescales of Proteus; (2) that, as with the basis for the Oktopus VOC model, traffic demands also vary in space – some VM pairs communicate more intensively than others; and (3) precise bandwidth guarantees might be both less necessary and less feasible than coarsely-specified guarantees. In the Cicada approach, the tenant starts with an initial placement of VMs and an estimated traffic matrix. The provider then profiles the application (similar to Proteus, but over longer timescales), with the aim of predicting the traffic matrix and how it varies diurnally. If the matrix appears predictable, the provider may propose to the tenant a time-varying or spatially-varying guarantee for future periods.

3. DESIRABLE PROPERTIES

Essentially every paper on the topic of cloud network performance sets out a list of desirable properties. However, people do not always agree on what is desirable, and sometimes there are inherent conflicts to resolve (as demonstrated in [12]). Beyond that, there are some issues that have not surfaced, to our knowledge, in prior work.

3.1 Consensus properties

There is agreement in broad terms that a solution needs to be *scalable*, *efficient*, and *predictable*. By "scalable" we mean that the solution must work for very large numbers of VMs, tenants, server machines, and (potentially) switches. It also means that the solution should scale to higher bandwidths. By "efficient" we mean both that the solution not require expensive hardware resources, and that it does not waste resources – the low-cost business model for cloud computing requires this.

Predictable performance is important in cloud networks. Most applications are themselves expected, by their owners or customers, to meet performance objectives (system-level objectives, or SLOs). When network performance (bandwidth, latency, or loss) becomes highly variable, either the SLOs suffer, or the application must be so over-provisioned with resources that its cost structure suffers.

Generally, the requirements for scalability and efficiency have led to designs that rely on resources at the edge (for example, hypervisor-based rate limiters) and avoid resources in the network core (e.g., switch-HW rate limiters). Hypervisor resources scale nicely with Moore's Law, and generally are easier to implement and upgrade than switch HW features.

3.2 Contentious properties

After these consensus properties, things become more complex. The problem is exacerbated by a lingering tendency to transfer desirable properties from the public Internet to a multi-tenant cloud network.

For example, we traditionally have wanted our networks to be *work-conserving* and *fair*. Both of these properties can be valuable in a cloud network, but not necessarily as we have previously understood them.

A work-conserving solution certainly improves efficiency over one that supports only static guarantees and therefore cannot always exploit reserved-but-unused resources. (E.g., Oktopus as described in [3] is not work-conserving.) However, a work-conserving system is not fully predictable: the excess bandwidth that you got yesterday might not be available tomorrow. Some cloud providers are reluctant to offer excess capacity at zero cost, because this risks "training" their customers to expect a lower total cost than is justified, or better performance than can be sustained in the future. Hence, we believe that the ideal solution should be work-conserving, but should support billing for bandwidth in excess of any guarantees.

Similarly, we almost certainly want to provide fairness between the flows of a given tenant; otherwise, applications such as MapReduce suffer from "stragglers."

But we do not believe it is necessary to strive for fairness *between tenants*. This statement might seem surprising at first.[5] However, while bandwidth guarantees provide a clear benefit to tenants, in the form of predictable bounds on runtime performance, fairness is an abstract concept that is harder to justify. Consider two customers (e.g., Coors and Budweiser) sharing the same cloud network. As long as Coors' VMs get the bandwidth that the provider has guaranteed to them, and can obtain additional bandwidth at reasonable prices, why should they care whether the provider is allocating the additional bandwidth fairly? In fact, what Coors should care about is *isolation*, in the sense that Budweiser cannot, by exceeding its own bandwidth limits, intentionally interfere with Coors' traffic.[6]

Beyond that, Popa *et al.* [12] showed that one cannot simultaneously achieve both fairness and minimum bandwidth guarantees. Whether network-level fairness is even achievable remains an open question, and it might require complex mechanisms.

The cloud provider, in fact, might want to engage in price discrimination – that is, providing different prices to different customers for the same level of service. Price discrimination is a common mechanism (think of airline tickets), and what it requires is that unfairness (if it exists) is under the control of the cloud provider, or that the provider can at least correct for unfairness in the spot-bandwidth billing mechanism. Simply preventing unfairness, per se, might not be worth the cost.

In general, we believe that one cannot discuss cloud performance without considering pricing. In particular, discussing whether a cloud network's services should be work-conserving or fair cannot be divorced from the economics of selling its services.

Is there a tradeoff between predictable performance, predictable bills, and minimal cost? Ballani *et al.* [2] suggest that DRP can provide predictable pricing to "job-like" (batch) applications, but that

[5]Some previous papers (e.g., [1, 12]) have made inter-tenant fairness an explicit goal, and many others have implied that such fairness is their goal.

[6]Lam *et al.* [11] similarly argue that "max-min fair sharing at the TCP level ... is the wrong model."

long-running user-facing services will get more predictable pricing from a model such as Oktopus or GateKeeper. However, while the latter models provide a fixed monthly cost rather than a predictable per-job cost, they cannot both do that and also maintain service-level performance guarantees under varying workloads.

3.3 Granularity of guarantees

There are several basic ways in which the endpoints (the "who" as opposed to the "what") for bandwidth guarantees can be expressed:

- **Tenant aggregate**: as in DRL, where an application specifies one global aggregate bandwidth.
- **per-VM Hose model**: as in Gatekeeper, where an application specifies $O(N)$ per-VM bandwidths, and perhaps latency limits.
- **per-VM Pipe model**: as in SecondNet where an application using N VMs would need to specify $O(N^2)$ pairwise bandwidths, and perhaps latencies.
- **per-flow QoS model**: as in SecondNet, which optionally supports QoS guarantees (bandwidth and/or latency) for TCP connections or TCP ports.

We list these models in order of increasing granularity and complexity of expressing a tenant's needs. There is a tradeoff here: low-complexity models (such as DRL) are easy to specify, but potentially hard to reason about, in terms of what the guarantees actually mean.[7] Fine-grained models offer more control to the application, but also impose more complexity on the application programmer, and could require considerably more resources to implement. (Even hypervisor cycles are not free.)

The Gatekeeper paper [14] has argued that the pipe and QoS models are too detailed, because "tenants do not understand their applications' communication patterns well enough to specify their bandwidth requirements between each pair of VMs." While SecondNet [11] supports per-flow or per-port granularity as an option, that seems to require excessive detail for most users. (Providers might need to allow users to control the relative priorities of some of their own flows.)

The VOC model [3] may strike a useful intermediate point in the complexity space: mostly it sticks to the hose model (which is plausibly easy to reason about, but not too complex to specify), but it allows the application's bandwidth requirements to reflect its coarse structure. VOC thus allows customers to pay only for the bandwidth that their applications need.

We fully expect additional research to generate new abstractions for expressing bandwidth guarantees.

3.4 Topology and location: hide or expose?

Ballani *et al.* [2] have argued for location-independent costs: "a tenant's location is a knob for the provider and is of no interest to the tenant." On the other hand, LaCurts *et al.* [10] showed that a tenant that knows the locations of its VMs in a cloud network can exploit this information to improve performance, and Webb *et al.* [17] have proposed that tenants should be able to specify properties related to the underlying physical topology.

From a provider's point of view, it can be risky to reveal too much about the underlying topology. Providers may need the flexibility to change their network topologies as technologies evolve, and do not want to be locked into an outdated design simply because their customers have become dependent on specific details. (While a provider might not rewire an existing availability zone, it might wire new AZs differently, and would not want a customer to depend on all AZs having the same topology.)

A cloud provider wishing to support customers with both job-like applications and long-running services may wish to expose a carefully chosen level of visibility for topology and VM location within this topology. This view could remain *virtual*, in the sense that a customer should only see enough to make rational choices about its contract with the provider, and about where to place functions within its virtual network.

A provider might not want to make actual physical links visible to customers – that is, to allow a customer to know or care whether two of its VMs are connected via any specific arrangement of links and switches. First, this might not provide any useful value to customers, as long as the provider is willing to support reasonable guarantees in other forms. Second, exposing this information could lock a provider into specific placements of tenant VMs, which would reduce the provider's ability to redistribute VMs for reasons such as churn, power management, or hardware maintenance.

Customers should expect to pay more for virtual-topology requests that place more specific constraints on the provider. For example, customer L has long-running analytics jobs, customer S runs a social-networking site, and both start with 10 VMs provisioned for 1Gbps (using the hose model). If either customer requests an increase to 20 VMs, L might tolerate some downtime, but S, with online users, would not. Since S's need for responsiveness precludes certain techniques that a provider could use to find more network bandwidth, such as moving VMs between racks, S might expect to pay more than L for the same per-VM bandwidth.

3.5 Other issues

A cloud provider that makes guarantees about network performance must also address several other issues.

Responsiveness: Almost all of the prior work has focussed on providing performance guarantees based on the nature or location of a VM, but with an implicit assumption that a VM's bandwidth needs are constant. For example, a long-running service may flex its VM allocation up or down, but its bandwidth requirements per VM can be considered constant. Batch applications, however, may have phases during which their bandwidth requirements change significantly, either in aggregate or with respect to hotspots in their traffic matrices.

Greenberg *et al.* [8] reported that traffic matrices in their data centers have little predictability on scales of 100 sec. or more. Lam *et al.* [11] identified the issue of "responsiveness": how quickly can the provider respond to a change in bandwidth requirements? We see responsiveness as a critical aspect of cloud guarantees; that is, a useful guarantee not only provides a bound on bandwidth and latency when the tenant's requirements are steady, but also provides a bound on how long it takes to restore these guarantees after a tenant requests a bandwidth increase or decrease (assuming the provider accepts the request). Providers can also benefit from clearly-specified non-zero bounds on responsiveness, since this gives them a known target for designing the bandwidth-reallocation mechanism.

Responsiveness is distinct from the opportunity identified by Xie *et al.* [18] as the basis for their Proteus system. While Proteus can exploit predictable patterns time-varying behavior, it does not address the problem of responding to unpredictable changes in requirements.

Support for operators, not just for programmers: For many potential cloud customers, applications are built by one team and operated by another. Because cloud applications often flex up and

[7] Vogels [16] reports that "pricing complexity" created problems for users of Amazon's SimpleDB service.

down as workloads change, tenant programmers need to think in terms of abstract sets of resources, but tenant operators must work with concrete resources (e.g., "today we need 15 VMs to run this application; tomorrow we will need 18 VMs.") Operators therefore probably need to be able to request changes in guaranteed or maximum bandwidth, in order to maintain predictable service levels, without having to make changes to application code, and without having to make per-flow or per-pipe configuration changes.

Billing intervals: Since existing cloud providers make no guarantees about network performance, they generally do not bill tenants for transfers within an AZ, and they bill by the GByte for other transfers. However, this simple model does not work well when a provider offers spot-market pricing for non-guaranteed bandwidth, since the billing system needs to distinguish between guaranteed-bandwidth bytes and excess (spot-market) bytes. This, in turn, requires the provider to choose a billing-measurement interval that is short enough to clearly distinguish between the two kinds of bytes, but long enough to avoid excessive measurement overheads.

The DRP mechanism proposed by Ballani *et al.* [2] presumably must make a similar distinction between periods when the VM's CPU is the bottleneck resource (and so the user is charged only for CPU utilization) and when the VM is network-limited (and so the user is charged only for network use).

Availability and reliability: Topology switching (Webb *et al.* [17]) offered tenants control not only over network bandwidth but also failure resilience, which they defined in terms of "the number of cuts r in the physical substrate required to break a logical link." They point out that increasing resilience might require increasing path length, which can hurt both latency and bandwidth. More generally, a provider could offer varying levels of network availability System-Level Agreements (SLAs) at different prices. While customers might want both high performance and high availability, these two properties might sometimes conflict, especially if the network performance guarantees must survive underlying link or switch failures. (This would require the provider to set aside some bandwidth for use during partial network failures.)

4. SUMMARY

We see a widespread recognition that cloud providers will have to start providing network-related guarantees in order to support a broad range of applications that demand predictable performance and cost. Many researchers have attacked cloud network performance, often with unique definitions of the problem to guide their choice of a solution.

Our view is that cloud-network performance guarantees:

- Should be **scalable, efficient, and predictable**; there is little controversy on these points.
- Should focus on **inter-tenant isolation and predictable pricing**, not on inter-tenant fairness, which by itself is not a primary concern for most tenants.
- Will need to strike a **balance between expressiveness and complexity**, especially in how much they aggregate guarantees over multiple flows and endpoints.
- Should **expose the tradeoff between availability, performance, and cost**, allowing the tenants to make choices when necessary.
- Should recognize that **an application's traffic demands are not always uniform in space and time**, and that a guarantee based on uniform demands can lead to inefficient economics.

While we do not expect all providers to adopt a single network performance model – in fact, there are many good reasons to offer a variety of options – we believe that further progress requires a common understanding of the choices and their implications for the relationship between cloud providers and their customers. In this paper, we have attempted to bring the research community closer to such an understanding.

5. REFERENCES

[1] M. B. Anwer, A. Nayak, N. Feamster, and L. Liu. Network I/O fairness in virtual machines. In *Proc. VISA*, pages 73–80, 2010.

[2] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. The price is right: towards location-independent costs in datacenters. In *Proc. HotNets*, 2011.

[3] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. In *Proc. SIGCOMM*, pages 242–253, 2011.

[4] B. Briscoe and M. Sridharan. Network Performance Isolation in Data Centres using Congestion Exposure (ConEx). `http://datatracker.ietf.org/doc/draft-briscoe-conex-data-centre/?includ%e_text=1`, 2012.

[5] R. Carver. *What We Talk About When We Talk About Love*. Knopf, 1981.

[6] N. Englander. *What We Talk About When We Talk About Anne Frank*. Knopf, 2012.

[7] M. Gerber and J. Schwarz. What We Talk About When We Talk About Doughnuts. *The New Yorker*, page 51, May 10 1999.

[8] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: A Scalable and Flexible Data Center Network. In *Proc. SIGCOMM*, 2009.

[9] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees. In *Proc. Co-NEXT*, 2010.

[10] K. LaCurts, S. Deng, and H. Balakrishnan. Choreo: Network-aware Workload Placement for Cloud Computing Systems. Unpublished work in progress.

[11] T. Lam, S. Radhakrishnan, A. Vahdat, and G. Varghese. NetShare: Virtualizing Data Center Networks across Services. Technical Report CS2010-0957, UCSD-CSE, May 2010.

[12] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica. FairCloud: Sharing the Network in Cloud Computing. In *Proc. SIGCOMM*, 2012.

[13] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren. Cloud control with distributed rate limiting. In *Proc. SIGCOMM*, pages 337–348, 2007.

[14] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes. Gatekeeper: supporting bandwidth guarantees for multi-tenant datacenter networks. In *Proc. WIOV*, 2011.

[15] A. Shieh, S. Kandula, A. Greenberg, and C. Kim. Seawall: performance isolation for cloud datacenter networks. In *Proc. HotCloud*, 2010.

[16] W. Vogels. Amazon DynamoDB. `http://www.allthingsdistributed.com/2012/01/amazon-dynamodb.html`, 2012.

[17] K. C. Webb, A. C. Snoeren, and K. Yocum. Topology switching for data center networks. In *Proc. Hot-ICE*, 2011.

[18] D. Xie, N. Ding, and Y. C. Hu. The Only Constant is Change: Incorporating Time-Varying Network Reservations in Data Centers. In *Proc. SIGCOMM*, 2012.