Towards a Network Marketplace in a Cloud

Da Yu
Brown University

Luo Mai
Imperial College London

Somaya Arianfar
Cisco Systems

Rodrigo Fonseca
Brown University

Orran Krieger
Boston University

David Oran
Cisco Systems

Abstract

Virtually all public clouds today are run by single providers, and this creates near-monopolies, inefficient markets, and hinders innovation at the infrastructure level. There are current proposals to change this, by creating open architectures that allow providers of computing and storage resources to compete for tenant services at multiple levels, all the way down to the bare metal. Networking, however, is not part of this, and is viewed as a commodity much like power or cooling. In this paper we borrow ideas from the Internet architecture, and propose to structure the cloud datacenter network as a marketplace where multiple service providers can offer connectivity services to tenants. Our marketplace, NetEx, divides the network into independently managed pods of resources, interconnected with multiple providers through special programmable switches that play a role analogous to that of an IXP. We demonstrate the feasibility of such an architecture by a prototype in Mininet, and argue that this can be a way to provide innovation, competition, and efficiency in future cloud datacenter networks.

1 Introduction

Cloud computing is currently dominated by a small number of cloud providers. Despite offering multiple public services to clients, each provider is vertically integrated and controls its own infrastructure. This monolithic provider model reduces the complexity of managing an infrastructure, but in turn limits the flexibility of using resources available at the lowest level of a cloud stack. For example, while it is possible for anyone to rent virtual machines from Amazon and offer a MapReduce service to clients, this service offering would be at a perpetual disadvantage vis-a-vis Amazon's own Elastic MapReduce offering. In this case, Amazon knows the location of data blocks, the network topology and VM allocation. It can then place computation for its own service where it is most advantageous. The single provider model inhibits competition at the infrastructure level, creates vendor lock-in, and artificial non-market-based pricing for resources [2].

To address the limitations of a single provider model, the Massachusetts Open Cloud Exchange (OCX) project [6] is a proposal to build a truly open, multi-provider cloud environment. The vision is to enable unprivileged service providers to compete in a marketplace with offerings at all levels of the system stack. The only privileged infrastructure, beyond power, cooling, and basic connectivity, is a hardware-as-a-service allocation layer, coupled with a set of exchanges where offers from providers are matched with service requests from clients. In the OCX, a provider can physically deploy machines in a common datacenter, and advertise access to these machines to clients. These clients can then, say, offer an OpenStack service on top of such leased physical machines, or use them to run Spark and offer it as a service to clients of their own. Different providers can also deploy physical machines and compete on features or price.

While the notion of multiple competing providers applies naturally to storage and compute, it does not immediately apply to networking inside the datacenter. In this paper we examine whether it makes sense for multiple providers to compete for networking services within an open cloud datacenter, *including at the physical layer*. The prevailing view is, rather, that the network is a utility, a common substrate, and if it is sufficiently provisioned, should not be a cause for concern. While it is possible that this is the case, this view hinders innovation and differentiation at the network level, and our goal is to have an architecture that does not preclude this.

Adopting a multiple provider model is also a natural response to rapid innovations in networking infrastructures. These innovations include novel transport stacks such as DCTCP [8] and RoCE [5], in-network processing elements such as pFabric [4] and programmable middleboxes [18], and emerging communication media such as reconfigurable optical switches [10] and 60GHz wireless [13]. Provisioning these heterogeneous fabrics in a marketplace not only allows clients to customize machine connectivity for improved communication performance, but also enables infrastructure providers to offer services that perhaps not be feasible to offer for the entire datacenter, or do so at differentiated prices for increased revenue.

The Internet offers a powerful analogy. Since its early prototype in 1980s, Internet has evolved from a single-provider environment into a successful ecosystem that can accommodate multiple network providers to compete in a marketplace. An organization today can physically connect to an IXP, and from there choose services from different transit providers that compete on capacity, re-

liability, connectivity, and cost. This open architecture allows the coexistence of 18+ low-latency providers between the New York and Chicago exchanges [15], or the multiple undersea cables with similar routes.

In the OCX architecture [6], multiple pools of physical resources – compute and storage – are provisioned and managed by different providers. These sets are called *pods*, and they can be, for example, a rack-scale computer [7], a physical container, or a storage pod [16]. Each pod is responsible for its internal networking, and they are inter-connected by a commodity network.

We extend this architecture by allowing multiple network providers to bridge these pods and compete by offering services in an open marketplace. To realize this, we design a network marketplace called Network Exchange (NetEx). To register in NetEx, network providers physically connect to a set of programmable Edge-of-Pod (EoP) switches. Similarly to the role of Internet Exchange Points (IXPs), these switches break a datacenter network into an inter-pod network that has alternative physical networking infrastructure exchanged in NetEx, and a closed, fast intra-pod network, e.g., a FatTree [3] network that provides full-bisection intra-pod bandwidth.

Users interact with NetEx by submitting high-level requests for connectivity. The marketplace forwards these requests to eligible providers (i.e., those connecting the relevant pods), which, in turn, return offers consisting of priced path segments, along with their characteristics. NetEx then facilitates the transfer of payment and the provisioning of the path segments selected by users. This high-level interface allows users to obtain service without being aware of the complexities of the different underlays, and providers to only expose the minimum required information for the market operation. Providers are free to implement paths however they see fit, and to use a wide spectrum of valuation and business strategies.

Although, as far as we know, we are the first to propose a network marketplace extending to the physical layer for a cloud datacenter, our design borrows liberally from the datapath mechanisms from Pathlet routing [11], Segment routing [9], and from some of the market aspects of the ChoiceNet architecture [21]. In this paper, we present a preliminary design for our network marketplace, NetEx (§2), and describe our initial Mininet prototype of the architecture (§3), which uses OpenFlow-directed MPLS forwarding to provision and connect path segments according to client requests. We conclude the paper with a number of interesting remaining research challenges.

2 Marketplace Design

In this section we describe our goals for a datacenter network that enables a marketplace where network providers can offer services to clients. We start with a motivating example, and then present the design for our NetEx architecture.

2.1 A Motivating Example

Figure 1 presents an overview of our datacenter network architecture. As mentioned in §1, the datacenter comprises independently provisioned and managed resource pods with their own internal networking. These pods are connected by one or more network providers via EoP switches. Each EoP switch is bridging a pod with potentially multiple network providers.

Suppose Alice wants to run a distributed machine learning task which can benefit from specific network properties. In the example, pod A has GPU compute resources, while pod C provides high-performance SSD storage. Suppose further that there are three network providers between pods A and C: provider X being the standard network with a regular TCP/IP over Ethernet; provider Y offering high-capacity DCTCP-enabled paths; and provider Z offering low-latency (but more expensive) RDMA over Converged Ethernet (RoCE).

Alice is willing to pay more for improved network performance, and queries the NetEx with requirements for low latency and high bandwidth. NetEx identifies providers X, Y, and Z as connecting the two pods, and forwards the request to them. They, in turn, reply with path segment offers and prices.

Alice knows training a model is communication intensive [17]. It requires (1) transferring a large number of training samples, and (2) frequently synchronizing model states across parallel workers. Considering this, she selects the DCTCP option for shipping samples as well as the RoCE option for reducing I/O blocking during synchronization. Her selections are passed to corresponding network providers: Y and Z. Upon receiving a path installation request, the providers install relevant path segments and return opaque handles for the paths. The pods involved then use these labels, plus the labels selected by the other pod, to install the proper tagging and forwarding rules. Part of Alice's request specifies the type of traffic to use each path (e.g., TCP RPC flows on port 5993 should use provider Z), and these become match rules for tagging at the ToR switches on each pod.

2.2 Design Goals

Independence and Isolation First and foremost, the architecture should respect network providers that are independent, and have the right incentives to compete. This means that they should have the freedom to deploy their infrastructure, set prices, and policies, however they see fit. The architecture requires providers to expose information that is necessary to offer accountable services. Changes internal to a provider or pod should not impact other players in the marketplace. The only changes to a

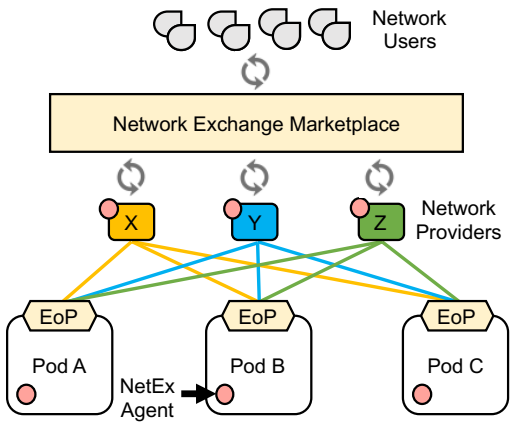

Figure 1: NetEx architecture overview

provider that should be visible to others is when the connectivity between players changes, e.g., when a provider connects to a new pod. This means that providers and pods should have a stable, well defined interface among them that enables independent internal evolution.

Neutrality As a marketplace that may accommodate a vast number of competing parties, NetEx is expected not to make any economical decision itself in order to avoid violating a neutrality principle. Network providers calculate their own prices for connectivity offers. These prices can reflect a wide spectrum of economic variables, making it possible to enforce various business policies.

Scalability The architecture should scale with the number of providers and pods. The forwarding performance of the architecture should be dependent on the internal capabilities of pods and providers, and the new interconnections among them should not introduce new bottlenecks.

Simplicity It should be simple for a network provider, or a pod, to take part in the marketplace through simple, well-defined and stable interfaces. It should also be straightforward for users to obtain marketplace services. Hiding internal details from market participants, as well as having the marketplace be the intermediary in provisioning selected services is key to lower adoption barriers.

2.3 Architecture Overview

The marketplace has two classes of participants. On one side, users make requests for connectivity, which the marketplace forwards to network providers. On the other side, both pods and network providers run their own NetEx agents, through which they interact in the marketplace. One key design decision is what is the object of negotiation. Similarly to Pathlet Routing [11], Segment Routing [9], or ChoiceNet [21], the basic unit of negotiation in NetEx is a path segment. A path segment can have any number of properties and capabilities, which clients express in *requests* to NetEx, and providers match or exceed in *offers*. Properties can include different forms of bandwidth or latency guarantees, as well as loss rate, reordering, dedicated queues, or disjointedness, for example. Capabilities may include the ability for clients to set parameters of individual switches, such as buffer sizes, queue configuration, or ECN parameters. The NetEx agents at pods and providers are responsible for provisioning and tearing down path segments.

To provide independence, providers reply to path requests with path segment properties; each request can elicit multiple responses. If these are selected, the provider provisions the path internally, and returns to NetEx an opaque handle to this path. This handle, combined with the ingress point in the provider's network (which indicates a particular pod), identifies a path segment for the purposes of forwarding and accounting.

To install an end-to-end route, NetEx queries the two end pods to gather their internal handles corresponding to the selected path. An end-to-end route is defined by the sequence of providers and their respective path segment handles. In our prototype, NetEx provides the agents at each pod with the complete set of labels for the path. In the data plane, paths are selected by having the source pod, perhaps in the ToR switch closest to the source, push a stack of labels that correspond to the path onto the packet. This is, in essence, a form of loose source routing similar to Pathlet Routing or ChoiceNet.[1]

To provide neutrality, NetEx does not make any choice or apply any policies to path negotiations.[2]

To achieve simplicity and scalability, the EoP switches are designed to be easy to attach to, and stable. Although they are programmable, they only have forwarding rules at the provider granularity, as opposed to having per tenant or per flow rules. Thus, it is only when there is a connectivity change to the EoP that NetEx has to install new rules at the EoPs.

Finally, the architecture allows for incremental deployment, accommodating clients which do not wish to negotiate paths, and pods which are not connected with providers through EoPs. All pods are interconnected by a standard network, which already exists in any datacenter, and is "free" (as in, the fees that pods or tenants pay to use the datacenter include a "tax" to maintain the regular network). In the absence of special configuration, default routes go through the standard network; when pods do deploy EoP switches, the standard network becomes a default, free network provider. Other providers offer value-added services and may charge for them.

[1] A viable alternative is to use a form of label switching at each provider, and have the route state distributed along the path, rather than fully embedded in the packets.

[2] it may, as in ChoiceNet [21], filter offers for clients to only consider Pareto-optimal offers.

2.4 Path Deployment

A packet may need to go through multiple physical underlays, managed by different pods and network providers. To keep their autonomy, we reduce the information they share to opaque labels. We make packets self-contained by encapsulating traversal-related information within their headers. This enables packets to reach their destinations without depending on external information.

Our current implementation relies on standard label stacks in MPLS, which is extensively available on commodity hardware, implying its sufficient performance in a production environment. At its first step along the network path, a packet can match arbitrary rules on the ToR switch, installed by the pod as a result of path provisioning. This rule pushes corresponding multi-level MPLS labels onto the packet header, which can be viewed as a series of nested tunnels, or, alternatively, as loose source routes. We do not preclude the possibility of using other techniques to implement the opaque handles, for example, VLAN tags, as well as the nested tunnel, for example, QinQ in the 802.1q standard.

2.5 Edge-of-Pod Switch Design

A key component of NetEx are the EoP switches. A naïve design for the EoP would leverage its programmability to install per-flow rules to steer traffic. However, a production datacenter could have a massive number of flows [8]. To maintain a relatively stable and small match-action table, we instead adopt per-provider tunnel rules. Per-provider tunnels are much more stable than per-flow tunnels. By using nested labels, we share the responsibility of steering the flows with the individual providers. At an EoP switch, we aggregate all tunnel rules pointing to the same network provider. These aggregated rules enable routing packets into corresponding providers at a modest space cost. As each provider is only responsible for a subset of flows, enforcing per-flow tunnels at the provider level becomes feasible. It is also easier to upgrade the infrastructure of one provider than it is to change the EoP, disrupting many providers and a pod at the same time.

EoP switches, then, need to be programmable, be able to perform high-performance switching based on (nested) labels, and have a modest number of high-speed ports, proportional to the number of providers. Ideally they would also be modular, to enable switching between different network technologies.

3 Prototype

To evaluate the feasibility of the NetEx proposal, we built a proof-of-concept prototype in Mininet [14]. For our emulated switches, we use `ofsoftswitch` [1], which is a user-level software switch compliant with OpenFlow 1.3.

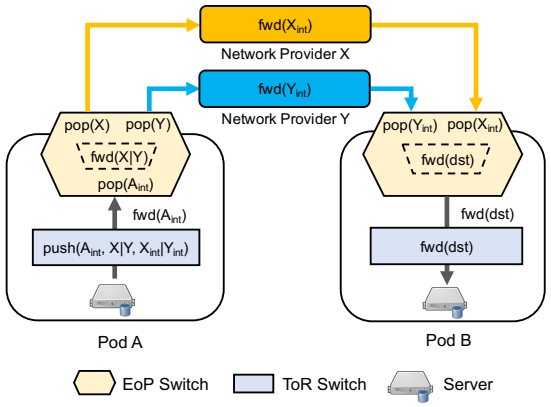

Figure 2: Packet forwarding detail

This version of OpenFlow supports a multi-level stack of MPLS labels, which we use to implement the NetEx datapath forwarding. Though preliminary, this environment allows us to examine if the proposed techniques are workable under a practical setting.

We describe our implementation through a detailed example, and show the results of a very simple experiment in which we steered ICMP traffic from one congested provider to another to avoid congested queues and reduce the latency of our traffic.

Figure 2 illustrates in some detail our setup for the experiment. We emulated two pods, A and B, connected by two network providers, X and Y. We assume that the paths have already been provisioned in the figure, some of the client's traffic will go through provider X, and some through provider Y, with specific rules indicating which traffic installed at the ToR switches (the boxes immediately above the machine icons). For simplicity, we will only walk through details in one direction. The reverse direction is analogous.

When a packet sent from the source host arrives at the ToR with the destination host IP, it will be matched on a rule that directs the ToR to tag this packet with a stack of labels (from outer to inner): one internal to provider A (A_{int}), one with a global identifier for the provider (X or Y), and an internal handle for the provisioned path segment within the provider (X_{int} and Y_{int}). Once this packet arrives at the EoP, EoP removes the first label (A_{int}) and checks the second label (X or Y) to identify which network provider this packet should be directed to. Before pushing this packet into the wire that connects with the network provider, EoP pops out the second label. One key aspect of this is that the EoP rules only depend on the provider, and not on this particular path; these rules are only changed in the event of physical connectivity changes to the EoP.

The selected provider forwards the packet as it sees fit using its internal label (X_{int} or Y_{int}), and the the packet

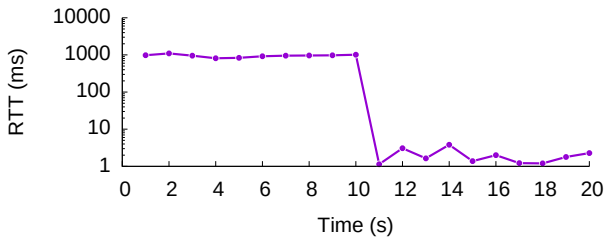

Figure 3: RTTs of ICMP packets. After 10s we steer ICMP packets to a new, uncongested network provider.

finally reaches the EoP in pod B. Similarly to the EoP in pod A, B's EoP decapsulates the last label (X_{int} or Y_{int}) and forwards this packet to an internal switch based on the destination. Of course, it is possible to have a specially provisioned path in pod B for this traffic, which would be represented by another nested label. The packet is then delivered to its destination.

We tested this setup with the following experiment: we installed a default rule steering all traffic from A to B to go through provider X, and ran an iperf test to saturate the path. We then started a set of ICMP ping messages to measure the round-trip time along this provider. After 10 seconds we provisioned a new path for ICMP packets, going through provider Y, which was idle. This involved provisioning a path at provider Y and obtaining a handle for that path, as well as installing a proper matching rule at the ToR switch at A (for the forward path), and at the ToR switch at B (for the reverse path). For simplicity, in each provider, we allocated an even label for a path in one direction and an odd label for the symmetrical reverse path. As we can see in Figure 3, the RTTs drop sharply.

4 Discussion and Future Directions

Our preliminary design for NetEx leaves many questions and challenges to be addressed, and we discuss some of them here. We also discuss the possibility of extending these ideas to the wide area.

Viability A common concern we have encountered with this design is whether it makes sense at all. Is it really viable to have multiple physical network providers in a single datacenter? While we cannot say whether a networking marketplace will harbor many providers and foster innovation, our goal is to design an architecture that allows this to happen, rather than to perpetuate the current single-provider architectures that prevent innovation. The Massachusetts Open Cloud (MOC) [19] project is building a multi-provider datacenter that has many industry and academic partners, and has a multi-provider architecture for computing and storage resources. We intent to pilot our network architecture at the MOC, to test the feasibility of these ideas. In a single-provider cloud, while the market component may be moot, our architecture can offer to tenants rich networking options at different prices, and to the provider a path to transparent deployment of and migration to new networking technologies.

EoP Design In our current design we strived to keep the EoP as simple as possible. While programmable, it has modest table space requirement, as its forwarding rules are at the granularity of providers, and not of tenants, or of flows. The main challenge in our architecture is to guarantee that the EoP will not be a bottleneck, which would prevent, for example, latency guarantees by network providers. One solution to this is to require traffic shaping close to sources (*a la* IntServ [20]). It may be possible to divide this responsibility, though, by using a small number of outgoing queues per EoP port, to at least protect one provider from another, but we are still investigating this issue.

Another challenge relates to connecting different technologies to EoPs. For example, a provider of 60GHz wireless links would probably need to add hardware at the EoPs to connect to its infrastructure. Our architecture does not preclude multiple EoP switches per pod, and in this scenario it is possible for the provider to subsidize specialty EoP switches that have the right interfaces.

Accountability and Transparency Since NetEx gives providers the power to control pricing strategies, accountability and transparency are key requirements for the marketplace to be successful. Given the competitive structure of the market, providers should have the incentive to honor their promised services, as customers can measure the obtained performance and choose other providers when not satisfied. However, inferring bottlenecks from end-to-end measurements may be hard, and we are investigating integrating transparent metering of path segments in the architecture. This will also aid in client charging and provide incentives for clients to behave. We are also considering providing transparency in the market prices of network offerings, to prevent discriminatory pricing.

Wide Area Extending the notion of a marketplace into WANs is also a promising direction. Today it is hard, for example, to have any control over the paths connecting cloud deployments that span multiple datacenters. To build a marketplace for WANs, we could largely reuse the building blocks for NetEx except the EoP switches. This gap can be filled by extending the emerging Software-Defined Exchanges [12]. Benefiting from their similar roles, most of our principles adopted in the EoP switch design can be applied to the marketplace-enabled SDX switch as well. By seamlessly extending our NetEx proposal, we might be able to spawn a unified marketplace for WANs and DCNs in the future.

References

[1] Openflow 1.3 software switch. https://github.com/CPqD/ofsoftswitch13. Accessed: 2016-03-07.

[2] AGMON BEN-YEHUDA, O., BEN-YEHUDA, M., SCHUSTER, A., AND TSAFRIR, D. Deconstructing amazon ec2 spot instance pricing. In *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science* (Washington, DC, USA, 2011), CLOUDCOM '11, IEEE Computer Society, pp. 304–311.

[3] AL-FARES, M., LOUKISSAS, A., AND VAHDAT, A. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev. 38*, 4 (Aug. 2008), 63–74.

[4] ALIZADEH, M., YANG, S., SHARIF, M., KATTI, S., MCKEOWN, N., PRABHAKAR, B., AND SHENKER, S. pFabric: Minimal Near-optimal Datacenter Transport. In *SIGCOMM '13* (New York, NY, USA, 2013), ACM, pp. 435–446.

[5] BECK, M., AND KAGAN, M. Performance Evaluation of the RDMA over Ethernet (RoCE) Standard in Enterprise Data Centers Infrastructure. In *DC-CaVES '11* (2011), International Teletraffic Congress, pp. 9–15.

[6] BESTAVROS, A., AND KRIEGER, O. Toward an open cloud marketplace: Vision and first steps. *Internet Computing, IEEE 18*, 1 (Jan 2014), 72–77.

[7] COSTA, P., BALLANI, H., RAZAVI, K., AND KASH, I. R2C2: A Network Stack for Rack-scale Computers. In *Proceedings of the ACM SIGCOMM Conference on Data Communication (SIGCOMM'15)* (London, UK, August 2015), ACM.

[8] CURTIS, A. R., MOGUL, J. C., TOURRILHES, J., YALAGANDULA, P., SHARMA, P., AND BANERJEE, S. Devoflow: scaling flow management for high-performance networks. In *ACM SIGCOMM Computer Communication Review* (2011), vol. 41, ACM, pp. 254–265.

[9] FILSFILS, C., PREVIDI, S., BASHANDY, A., DECRAENE, B., LITKOWSKI, S., HORNEFFER, M., SHAKIR, R., TANTSURA, J., AND CRABBE, E. Segment routing with mpls data plane. *Internet Engineering Task Force, Internet Draft (Work in Progress) draft-ietf-spring-segment-routing-mpls-00* (2014).

[10] FUNNELL, A., BENJAMIN, J., BALLANI, H., COSTA, P., WATTS, P., AND THOMSEN, B. High Port Count Hybrid Wavelength Switched TDMA (WS-TDMA) Optical Switch for Data Centers. In *Proceedings of the 2016 Optical Fiber Communication Conference (OFC'16)* (Anaheim, CA, US, March 2016).

[11] GODFREY, P. B., GANICHEV, I., SHENKER, S., AND STOICA, I. Pathlet routing. *SIGCOMM Comput. Commun. Rev. 39*, 4 (Aug. 2009), 111–122.

[12] GUPTA, A., VANBEVER, L., SHAHBAZ, M., DONOVAN, S. P., SCHLINKER, B., FEAMSTER, N., REXFORD, J., SHENKER, S., CLARK, R., AND KATZ-BASSETT, E. Sdx: A software defined internet exchange. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (New York, NY, USA, 2014), SIGCOMM '14, ACM, pp. 551–562.

[13] HALPERIN, D., KANDULA, S., PADHYE, J., BAHL, P., AND WETHERALL, D. Augmenting data center networks with multi-gigabit wireless links. In *ACM SIGCOMM Computer Communication Review* (2011), vol. 41, ACM, pp. 38–49.

[14] HANDIGOL, N., HELLER, B., JEYAKUMAR, V., LANTZ, B., AND MCKEOWN, N. Reproducible network experiments using container-based emulation. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies* (2012), ACM, pp. 253–264.

[15] LAUGHLIN, G., AGUIRRE, A., AND GRUNDFEST, J. Information transmission between financial markets in chicago and new york. Working Paper 442, Stanford Law and Economics Olin, November 2012.

[16] LI, H., GHODSI, A., ZAHARIA, M., SHENKER, S., AND STOICA, I. Tachyon: Reliable, memory speed storage for cluster computing frameworks. In *Proceedings of the ACM Symposium on Cloud Computing* (2014), ACM, pp. 1–15.

[17] MAI, L., HONG, C., AND COSTA, P. Optimizing network performance in distributed machine learning. In *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)* (Santa Clara, CA, July 2015), USENIX Association.

[18] MAI, L., RUPPRECHT, L., ALIM, A., COSTA, P., MIGLIAVACCA, M., PIETZUCH, P., AND WOLF, A. L. NetAgg: Using Middleboxes for Application-specific On-path Aggregation in Data Centres. In *Proceedings of the 10^{th} ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT'14)* (Sidney, Australia, December 2014), ACM, pp. 249–262.

[19] Massachusetts open cloud. `http://info.massopencloud.org/about/`. Last accessed May 2016.

[20] PONNAPPAN, A., YANG, L., PILLAI, R., AND BRAUN, P. A policy based qos management system for the intserv/diffserv based internet. In *Policies for Distributed Systems and Networks, 2002. Proceedings. Third International Workshop on* (2002), IEEE, pp. 159–168.

[21] ROUSKAS, G. N., BALDINE, I., CALVERT, K., DUTTA, R., GRIFFIOEN, J., NAGURNEY, A., AND WOLF, T. Choicenet: Network innovation through choice. In *Optical Network Design and Modeling (ONDM), 2013 17th International Conference on* (2013), IEEE, pp. 1–6.