EdgeCloud: Network Support for Wide-Area Applications

Richard Mortier, Derek McAuley
University of Nottingham,
Jubilee Campus,
Nottingham NG7 2TU, UK
firstname.lastname@nottingham.ac.uk

Anil Madhavapeddy, Charalampos Rotsos,
Andrew W. Moore, Jon Crowcroft
University of Cambridge,
15, JJ Thomson Ave,
Cambridge CB3 0FD, UK
firstname.lastname@cl.cam.ac.uk

Abstract

In recent years, applications have migrated from the desktop into the network: many now require network connectivity, use web APIs to provide core functionality, and are used on the move from mobile devices. Such *wide-area applications* are often hosted on virtualized infrastructure, to be scaled, migrated and managed in units of individual virtual machines. However, network support for applications has not significantly changed in this time: the basic service offered remains nearly-transparent connectivity, with deployment of middleboxes generally making things less reliable and more resistant to change.

In this paper we propose the *EdgeCloud*, combining OS virtualization and OpenFlow to offer an alternative that allows applications to safely exercise control over the access network. Among its benefits are the ability for applications (many of which are already architected for distributed operation due to running on the cloud) to replicate themselves in the network to serve client API requests close to the user. This exposes interesting trade-offs: the ability to optimize for cost *vs*. latency, and implement delay-tolerant applications without depending on energy-starved client devices. Unlike static content delivery networks, the EdgeCloud also permits ISPs to deploy their own middleboxes (for authentication, caching or filtering) on the same infrastructure, providing an important economic incentive to deployment.

1 Introduction

While the Internet's ossification is often decried in the network research community, it has changed a lot in the last 10 years. We have seen the explosive rise of social media such as Twitter and Facebook, of application hosting platforms such as Amazon EC2, Google App Engine, and of media sharing sites such as Flickr and YouTube. These surface changes have been enabled by underlying technical changes, some radical, some incremental: adoption of the mobile smartphone as an Internet client, widespread deployment of virtualization technologies, and dramatically increased bandwidth and spread of consumer network connections.

In this time, the *architecture of the application* has fundamentally changed. Applications have migrated from the desktop into the network, with more and more expecting network connectivity, accessing or relying on web services and APIs for their core functionality. We refer to such applications as *wide-area applications*, and examples include blogs, social media such as Twitter and Facebook, and the many services and APIs that Google and Microsoft now provide. The mode of use of such applications is also fundamentally different: commonly accessed from mobile devices (smartphones, tablets) on the move, their use involves significant amounts of traffic travelling from the user to the application, rather than traditional delivery of content to the user.

At the same time, the *architecture of hosting* has been radically changing. The rapid take-up of virtualization technology such as Xen, and its use in provisioning cloud platforms has led to explosive growth in the number of deployed networked services and a re-engineering of the way that they are constructed. Rather than rely on multi-purpose installs (for smaller systems) or single purpose physical hosts (for larger systems), these applications are now created as instances of a set of single-purpose *virtual machine images* for hosting on platforms such as Amazon's EC2. Resilience, migration to new hardware, and scaling of the application are thus managed in units of a virtual machine image.

However, while these changes have been taking place the *architecture of the network* has not changed nearly so successfully. Although technology and operational practise have evolved in some areas, notably customer access and the datacenter, by-and-large the same assumptions still apply: you will still likely be charged by the byte to deploy a network service on a hosted platform, and

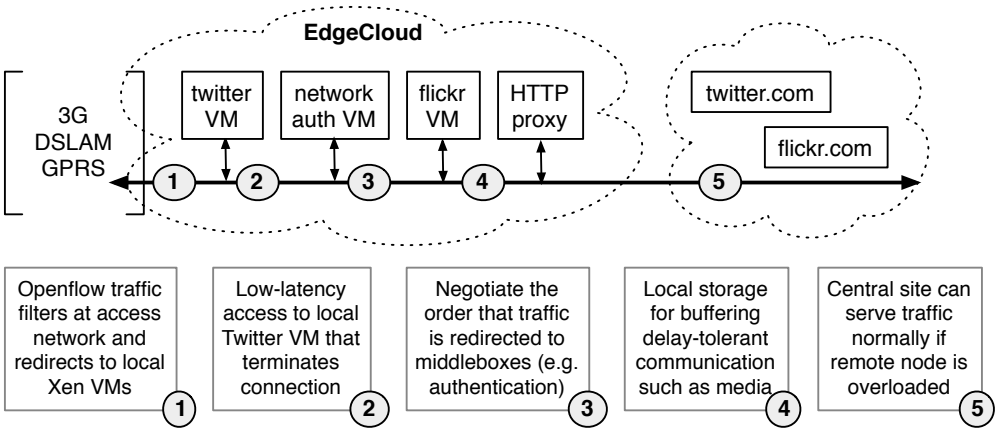

Figure 1: *EdgeCloud* architecture.

the network will still be presented as a fabric that more-or-less transparently connects clients to your application. *More-or-less* because performance mismatches have led to operators deploying a range of middleboxes at various places in the network. Unfortunately, even where traffic remains unencrypted, middlebox vendors simply cannot keep up with rates of application development leading them to 'bake in' assumptions about protocols and their use by applications, making changing things at this level much harder.

We propose *EdgeCloud*, a system in which wide area applications exercise more direct control over the network, enabling dynamic deployment of application-specific middleboxes. This permits improvement to application-perceived service quality, as well as enabling applications to make more efficient and cost-effective use of the range of public hosting services now available. In this paper we will explore scenarios which motivate EdgeCloud (§2) before giving technical details of EdgeCloud (§3), discussing challenges posed by, and benefits, of our design (§4), and concluding (§5).

2 Context & Scenarios

EdgeCloud, depicted in Figure 1, is primarily concerned with support for *wide-area applications*, which we define as networked applications that make use of resources interconnected by a wide-area network because, e.g., they consist of instances running in geographically distributed datacenters. Another key feature of these applications is that they tend to be *read-write*: it is not enough simply to serve up static, or even dynamically generated, content via caches and proxies as provided by services such as Akamai or Inktomi. Clients connect via networks with a wide range of characteristics, from mobile data networks to wifi hotspots to home networks with asymmetric upstream bandwidth. Clients also appear to be highly mobile, both because they physically move but also because clients are drawn from a global population and thus diurnal affects mean that the location of currently active clients appears to shift as time passes.

To take two representative examples at different ends of the spectrum, consider Flickr, the photo sharing platform and Twitter, the micro-blogging platform. Flickr's service is relatively unaffected by latency changes, at least on the order of seconds or minutes, but requires reasonably high bandwidth both to and from clients. In contrast, although each request processed by Twitter involves only low bandwidth communication to each client, it significantly benefits from low latency. Both services will suffer from flash-crowd effects, due to local populations going to bed and waking up, as well as more significant local variation due to events, e.g., hosting the Olympic Games, a major rock concert, or even political upheaval.

Now consider a mobile client accessing these services via a 3G connection. Even if first hop connectivity is acceptable (and it often isn't), it is common that backhaul connectivity out of the mobile operator's network to the Internet, and thus to the wide area application, is poor. Both bandwidth-hungry and latency-sensitive services, such as Flickr and Twitter respectively, would benefit from installing an application-specific middlebox in, or as near as possible to, the mobile operator's network.

For a service such as Flickr, poor backhaul connectivity leads to clients perceiving poor application performance as their photos take a long time to upload. As a result Flickr would benefit from the ability to install an application specific middlebox able to construct thumbnails of the photos for immediate upload to Flickr, saving bandwidth, while scheduling upload of the high resolution originals for a later time. It is preferable to perform this transcoding in the network rather than directly on the mobile client, both to save battery by reducing CPU

use and allowing the radio to be turned off, and to better schedule resources across a large crowd of local clients.

Services like Twitter are early examples of the so-called *real-time web* and rely on low-latency connectivity. Providing location-based services based on, e.g., geo-tagged updates, requires the reported location to remain accurate when the service is provided, e.g., the advert is served. This distinction in priority of data is only visible to the *application*: in edge networks and other resource-constrained networks, only the application can make informed decisions about scheduling and buffering. As bandwidth hungry real-time applications become widespread, the benefits from installing application-specific middleboxes as near as possible to clients will increase.

Similar benefits could be realised for services more commonly used over home broadband, such as BitTorrent and iPlayer. For example, ISPs often throttle more expensive inter-domain BitTorrent traffic in an attempt to encourage BitTorrent clients to prefer service from other (cheaper) nodes within their network.[1] A number of ISPs have expressed concerns over the bandwidth usage of the BBC's iPlayer application. If the BBC could explicitly deploy middleboxes into those networks to cache content they know to be popular and to capture iPlayer clients when they access that content, redirecting them to the pre-cached copy inside that network, then the BBC, consumers and ISPs would all benefit from better service and lower bandwidth usage.

Finally, other examples of fine-grained network control benefiting service providers include BT's OpenZone and Fon offerings. BT OpenZone is a wireless hotspot service requiring subscription for use, but carrying Skype traffic for free even for users without subscription.[2] Similarly, BT Fon is a service allowing BT broadband subscribers to transparently share wireless access points with other subscribers: any Fon subscriber can make use of wireless connectivity provided by any other Fon subscriber. Managing the service provisioning and interpositioning of firewalls, proxies and other middleboxes in such cases is complex and time-consuming. The ability for other application providers to provision service on such an extreme edge network would enable them to provide much better service during flash crowd events such as the upcoming 2012 Olympics, but is currently too complex to engineer in a timely and dynamic fashion.

3 Application Deployed Middleboxes

EdgeCloud aims to allow applications to deploy across the wide-area Internet in ways specific to their needs, subject to the overarching policies imposed by the local network providers. The provider wishes to ensure their network is not used for nefarious, or simply unexpected, purposes; while the customer wishes to ensure their network use is efficient and predictable, as they must usually pay for bandwidth and local resources consumed. The only technologies currently deployed to aid either party in these aims are either vendor-specific edge boxes from, e.g., Akamai, Riverbed, Ironport, which are typically small-scale and/or support only relatively static content; or the traffic interception middleboxes imposed by the provider, which can hinder as much as they help [2].

The prime motivations for cloud computing are the *convenience* of managing software images over physical hardware, and the *efficiency* gains due to statistical multiplexing of resources. There are particularly incentives to realise these benefits in ISP access networks as they are geographically dispersed and thus often under-provisioned due to long hardware upgrade cycles.

EdgeCloud assumes a typical cloud computing context: physical machines running the Xen hypervisor to host Linux or Windows virtual machines on which customers run their own software and services. The Xen hypervisor has a trusted domain, `dom0`, which runs a variety of management software including an OpenFlow controller which manages the hypervisor's Open vSwitch instance to provide network connectivity to guest domains. The core of our system is to embed OpenFlow controller functionality as near to the application as possible, allowing it to make informed decisions of network control while still respecting local policy.

We envision that existing distributed application platforms can use EdgeCloud to dynamically provision instances closer to their users, benefiting from reduced real-time latency and increased predictability, as well as aggregating and scheduling upstream traffic based on its criticality and delay-tolerance. Meanwhile, ISPs can also use the platform to deploy virtualized middleboxes for existing services such as authentication or compression, more efficiently reusing physical resources.

This architecture immediately raises four questions which we address in subsequent sections: (*i*) How do applications gather the information required to make informed network control decisions? (*ii*) How can the network inform applications of changes to its state? (*iii*) How can this control be exposed to the application, in ways that allow it to benefit from the extra complexity exposed? (*iv*) How does the operator ensure their policies are still applied network-wide, while allowing the application to exercise useful control over the network?

3.1 Application Control

To make use of EdgeCloud, the application must connect to its local network nodes to obtain information and de-

[1] `http://wiki.vuze.com/w/Bad_ISPs`

[2] `http://about.skype.com/press/2011/02/wifi.html`

clare its desired use of the network. There are several mechanisms by which this can be implemented, depending on the transparency required by the application developer:

Transparent monitoring. Where *zeroconf* protocols such as mDNS, UPNP, and Bonjour are in use, passive monitoring of the network allows information about the services exposed by different hosts to be gathered. This can be used to infer the connectivity required by those hosts.

API interposition. To support existing applications, and to ease the burden on developers more generally, a transparent interposition library can be used to wrap the standard Berkeley Sockets API. This is straightforward in UNIX systems using facilities such as `LD_PRELOAD` to wrap socket API calls so that they interact with the various components of EdgeCloud in addition to their usual actions. This technique has been successfully used by others [7] to upgrade existing applications.

Modifying frameworks. Many modern Internet applications do not use the sockets API directly, instead building on application frameworks that abstract away the concerns of connection establishment, load balancing and request routing. In addition, the shift towards "NoSQL" databases [9] means these applications are often specifically required to cope with eventual consistency. This combination means it is feasible to integrate EdgeCloud directly into these frameworks to control aspects such as load-balancing and delay-tolerant data batching, without requiring an extensive rewrite of the bulk of the application logic.

For example, Twitter use the open-source Finagle library[3] to handle over 13 billion API requests daily. Finagle provides applications with notifications of connection back-pressure (whether due to malicious clients or a congested link), and service discovery via Apache Zookeeper, all of which notifications EdgeCloud can hook into. Most deployed cloud applications already support failover via for multiple clusters, isolated from each other against various classes of failures.[4] EdgeCloud nodes simply look like additional clusters with request routing logic, often as simple as geographical clustering for social networking applications [15, 12].

In all cases the aim is simply to gather the information to enable the correct flow entries to be created to support application requirements.

3.2 Applying Policy

A natural tension exists between the application provider and the ISP, who will place requirements on use of their access network. For example, the ISP may wish to restrict use of broadcast protocols in a shared hosting environment; they may apply default firewall policies across their whole network; or they may wish to interpose proxies for protocols such as HTTP on all outbound connections. Traditionally, these middlebox services interpose transparently, often degrading the offered service, and certainly leading to stagnation of protocol development as assumptions about behaviour get baked into middlebox hardware.

Instead we propose that interposition of middleboxes be explicitly shared with edge content delivery services, permitting negotiations between application developer and hosting provider. Recall our earlier example of Skype wishing to pay for free access to their service from wireless hotspots in the UK: by unifying the traffic redirection platform, this sort of deal is much easier to deploy.

This negotiation starts with a *flow manifest*, a statement of desired network connectivity by the application developer, e.g., bidirectional ARP, matching incoming/outgoing DNS requests/responses, or incoming HTTP/HTTPS connections. In a process analogous to use of Kerberos [5], the manifest is passed to a service running in the network/datacenter where it is checked against provider's policies and exceptions raised if required. If the manifest is deemed acceptable, signed tokens are returned which the application passes along with any requests to its host's dom0 to prove that the operation is permitted by the provider's policy.

3.3 Gathering Information

In order to make informed decisions controlling the network, each controller must have access to the network's current state. This state ranges from relatively static (existence of particular switches), to dynamic (state of particular links), to highly dynamic (link load/congestion information). While applications may be able to optimise their network use given transient link congestion information, it is likely that the timescales involved mean that doing so via OpenFlow control mechanisms will lead to instability, and so we leave the problem of managing transient congestion to protocols better placed to do so, such as TCP. However, it may be of benefit for applications to gather longer-term statistics concerning the relative loading of links to aid flow placement, particularly when the flows concerned are likely to be long-lived.

The key information that must therefore be provided is concerned with link liveness: which links exist and which are currently available. Current OpenFlow deployments appear to assume the pre-existence of a secure channel from controller to each switch, and then use protocols such as LLDP [8] to gather information about each switch's state. Information to bootstrap these connections could also be gathered and propagated through LLDP. For example, the LLDP protocol could be used to disseminate link liveness data, to be flooded through the

[3] `https://github.com/twitter/finagle`

[4] Amazon EC2 dubs these *availability zones*.

datacentre, and passed to all dom0 controllers by default. Each dom0 controller can then act as a natural cache of this information for all guest domains above it.

In addition to link-state data, the controllers need to know sufficient layer 3 routing data to direct traffic to destinations outside the local network. In small networks making use of default routes, it might be feasible to disseminate routing information to every controller. In larger networks, particularly those carrying full routing tables, summarisation is likely to be required perhaps to the extent of only making each controller aware of (some) gateways keyed by, e.g., destination subnet.

No matter the mechanism used to bootstrap the controller secure channels, information of whether a switch is up and a link present can easily be extracted via LLDP or even directly from OpenFlow using parts of its statistics API: `of_port_stats`, `of_ping`. These methods also allow collection and aggregation of performance statistics by controllers as discussed above. It is likely that, for these more detailed statistics, the operator would wish to run an access-controlled centralised collection point.

4 Discussion

EdgeCloud creates something of a hybrid system, blurring the boundaries between host and network, application and protocol. Although not wholly controlled by any application, the network is certainly no longer such a black-box. In this section we note challenges posed by, and benefits arising from, this architectural change.

4.1 Challenges

But isn't this just ... ? At the core of EdgeCloud lies the ability for applications to better manage network resource by instantiating and controlling middleboxes at suitable points in the network. This idea has some similarity to significant past bodies of work, e.g., active networks [14], quality-of-service [17], and filter-based networking [1]. In general, these ideas have not seen significant take-up for practical reasons — in those cases, hardware developments, unscalable amounts of per-flow state, and router performance and security. The combination of virtualization and open network control makes EdgeCloud both more feasible and immediately applicable.

Indeed, in many ways EdgeCloud is an *inevitable* evolution of the static content caches provided by companies such as Inktomi and Akamai, and WAN accelerator devices such as Riverbed's Services Platform.[5] The need for more computation at the edge network started with ICAP [4] for dynamic web content; grew with the widescale deployment of appliance middleboxes, e.g., Ironport and Bluecoat; and now culminates with support for general-purpose applications via the combination of OS virtualization, distributed application frameworks and open network APIs.

What about net neutrality? Arguments over net neutrality, the mandated even-handed treatment of all traffic by the network, continue to rage. In many ways EdgeCloud explicitly violates net neutrality: applications can request special treatment of their traffic by the network provider. However, we believe this is actually a good thing: by making these decisions both explicit and dynamic, sufficient fluidity and transparency allow creation of a market within which negotiation can take place, to the benefit of both application and network providers. The former can ensure that when resources are scarce, their requirements can be expressed to the limit they are willing to pay; the latter is able to differentiate service and more directly leverage the inherent value of having low-latency, high-bandwidth connectivity to a pool of consumers.

Is OpenFlow entirely sufficient? We have presented EdgeCloud as using OpenFlow for pragmatic reasons: it is actively under development, already deployed in a variety of switches and, as used by our prototype implementation, integrated into XenServer via Open vSwitch. A number of other researchers use OpenFlow in proposing several filter-based network security systems for enterprise networks [3, 16], hosts [10] and hypervisors [11]. However, the use of HTTP as the new 'narrow waist' of the IP protocol stack means OpenFlow cannot directly express the control we need: it is effectively limited to control based on Ethernet/MPLS, IP addresses and TCP/UDP port numbers. This is easily addressed either with logic implemented in coordinated local controllers at strategic points in the network or, given suitably powerful processing in switches, by extending OpenFlow to enable flow matching against selected HTTP protocol header fields, e.g., Host and Request-URI. Indeed, there are already proposals under consideration[6] to make HTTP more middlebox-aware, which would only improve the integration between EdgeCloud and end hosts.

4.2 Benefits

Mobility. In a hosted environment, the system we propose effectively whitelists *services* rather than (host, port) pairs. In particular, the direct link to the flow entries in the network makes virtual machine image mobility straightforward: the provider can identify the flow path into which packets should be placed using the DNS A record response to a name query as an opaque identifier. Flow entries can then be updated as the virtual ma-

[5] `http://www.layer47.com/riverbed_RSP.html`

[6] `http://www.ietf.org/id/draft-nottingham-http-portal-02.txt`

chine image migrates around the datacenter: as standard IP routing and aggregation is not in use, the image need not have its addresses updated and can remain oblivious to the moves. Similarly, as the network manages the relation between the application's traffic and any installed middleboxes, the multitude of configuration files in use need not be changed: the network simply ensures traffic continues to flow according to application and ISP policies. If desired, the ISP could expose more information to applications that care in DNS TXT, SRV or WKS records, such as the services offered by different names and where those images are located.

Security. EdgeCloud requires the ISP and the application provider to express their network demands more explicitly than at present, and makes the network fabric more capable of dynamically enforcing these requirements. This helps provide resilience against configuration errors and changes: the network fabric can filter traffic according to policies negotiated between ISP and application provider. If proposals to introduce OpenFlow into the home router [13] were deployed, these security benefits would extend into the home: traffic that had not been explicitly requested by devices in the home would not be able to flow toward the home.

Dynamic services. As briefly noted earlier, the combination of EdgeCloud with spot pricing of cloud resources[7] gives rise to some interesting possibilities. Applications can explicitly and *dynamically* trade-off hosting costs against service level, choosing when to deploy a new instance on the EdgeCloud in light of current and expected load, and the advertised spot price for hosting and network resources. The same infrastructure that handles VM migration can respond to update DNS and flow entries to enable load balancing between the multiple images of the application. Similarly, deployed instances can make intelligent decisions about when to utilise available resources such as network.

Finally, the creation of a market for network resources in which ISP and application provider participate makes it much easier for the application provider to be flexible in their service pricing. The ability to place application instances within the ISP's network allows the application provider to create special offers to selected consumers. For example, Twitter might use EdgeCloud to host instances within BT and Vodafone's networks, allowing them to offer free mobile tweeting for all Twitter users in London during the Olympic Games. To do so currently would require extensive negotiation between the companies involved to ensure that traffic volumes would not be seriously impacted. With EdgeCloud, the traffic involved is easily billed directly to Twitter, and its network use can be managed directly by their application instance(s).

[7] `http://aws.amazon.com/ec2/spot-instances/`

5 Conclusion

We have discussed the EdgeCloud, a hybrid network architecture using open virtualization and network control to allow application providers and ISPs to dynamically deploy application-specific middleboxes. We believe that by opening the access network up to be safely controlled directly by applications, we give the developer a far richer API than is currently available. At the same time, we retain the ability for the operator to impose their own policies over the customers' use of their network and benefit economically. Finally, a virtualised edge infrastructure built with commodity hardware [6] is far easier to upgrade than vendor-specific middleboxes, improving the ability to experiment with new services and protocols.

References

[1] Argyraki, K., and Cheriton, D. R. Network capabilities: The good, the bad and the ugly. In *Proc. 4th ACM HotNets* (2005).

[2] Carpenter, B., and Brim, S. Middleboxes: Taxonomy and Issues. RFC 3234, IETF, Feb. 2002.

[3] Casado, M., Freedman, M. J., Pettit, J., Luo, J., Gude, N., McKeown, N., and Shenker, S. Rethinking enterprise network control. *IEEE/ACM Trans. Netw. 17*, 4 (Aug. 2009), 1270–1283.

[4] Elson, J., and Cerpa, A. Internet Content Adaptation Protocol (ICAP). RFC 3507, IETF, Apr. 2003.

[5] Garman, J. *Kerberos: The Definitive Guide*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2003.

[6] Greenhalgh, A., Huici, F., Hoerdt, M., Papadimitriou, P., Handley, M., and Mathy, L. Flow processing and the rise of commodity network hardware. *SIGCOMM Comput. Commun. Rev. 39* (March 2009), 20–26.

[7] Guha, S., and Francis, P. An end-middle-end approach to connection establishment. In *Proc. ACM SIGCOMM* (2007), pp. 193–204.

[8] IEEE Computer Society. *802.1AB IEEE Standard for Local and Metropolitan Area Networks: Station and Media Access Control Connectivity Discovery*, 2005. IEEE Std 802.1AB–2005.

[9] Leavitt, N. Will NoSQL databases live up to their promise? *Computer 43*, 2 (Feb. 2010), 12–14.

[10] Nayak, A. K., Reimers, A., Feamster, N., and Clark, R. Resonance: dynamic access control for enterprise networks. In *Proc. 1st ACM Workshop on Research on Enterprise Networking (WREN)* (2009), pp. 11–18.

[11] Popa, L., Yu, M., Ko, S. Y., Ratnasamy, S., and Stoica, I. CloudPolice: taking access control out of the network. In *Proc. Ninth ACM HotNets* (2010), pp. 1–7.

[12] Pujol, J. M., Erramilli, V., Siganos, G., Yang, X., Laoutaris, N., Chhabra, P., and Rodriguez, P. The little engine(s) that could: scaling online social networks. In *Proc. ACM SIGCOMM* (2010), pp. 375–386.

[13] Valancius, V., Laoutaris, N., Massoulié, L., Diot, C., and Rodriguez, P. Greening the Internet with nano data centers. In *Proc. 5th ACM CoNEXT* (2009), pp. 37–48.

[14] van der Merwe, J. E., and Leslie, I. M. Switchlets and dynamic virtual ATM networks. In *Proc. 5th IEEE International Symposium on Integrated Network Management* (1997), pp. 355–368.

[15] Wittie, M. P., Pejovic, V., Deek, L., Almeroth, K. C., and Zhao, B. Y. Exploiting locality of interest in online social networks. In *Proc. 6th ACM CoNEXT* (2010), pp. 1–12.

[16] Yu, M., Rexford, J., Freedman, M. J., and Wang, J. Scalable flow-based networking with DIFANE. In *Proc. ACM SIGCOMM* (2010), pp. 351–362.

[17] Zhang, Z.-L., Duan, Z., Gao, L., and Hou, Y. T. Decoupling QoS control from core routers: a novel bandwidth broker architecture for scalable support of guaranteed services. In *Proc. ACM SIGCOMM* (2000), pp. 71–83.