IEEE International Conference on Communications 2013: IEEE ICC'13 - Workshop on Cloud Convergence: challenges for future infrastructures and services (WCC 2013)

Offloading Routing Complexity to the Cloud(s)

Hasan T. Karaoglu* and Murat Yuksel†
hkaraogl@cisco.com, yuksem@cse.unr.edu
*Cisco Systems, Inc. San Jose, CA 95134.
†CSE Department University of Nevada, Reno, NV 89557.

***Abstract*—We propose a new architectural approach, Cloud-Assisted Routing (CAR), that leverages high computation and memory power of cloud services for easing complex routing functions such as forwarding and flow level policy management. We aim to mitigate the increasing routing complexity to the cloud and to seek an answer to the following question: "Can techniques leveraging the memory and computation resources of the cloud remedy the routing scalability issues?" The key contribution of our work is to outline a framework on how to integrate cloud computing with routers and define operational regions where such integration is beneficial.**

***Index Terms*—cloud-assisted routing; software-defined networking; routing scalability**

I. Introduction

Concerns over routing scalability has increased recently [1]–[3]. Currently, there are more than 30,000 service providers advertising more than 350,000 IP prefixes [4], with a typical edge router receiving hundreds of updates per second. Further, BGP churn can grow prohibitively if topology growth and update dampening are not performed carefully [5]. These concerns become more serious as the Internet topology is becoming more flat [6], putting more burden on the core routers. Multi-homing and peering practices as well as the demand on more routing flexibility (e.g., multi-path routing, QoS routing) have been contributing to the routing complexity issues the Internet faces. Operators rightfully expect more router programmability, which further challenges router architectures by inclusion of software-based designs [7] and virtualization of routing as a service [8], [9].

As the complexity on the routers increased, the cost of a router became non-trivial. The cost of routing unit traffic has not been reducing at a pace similar to the performance improvement of computing capabilities of a router [2]. Given the trends on the state and packet processing capacities expected from a BGP router, the cost of a router that can perform the basic routing functions at the Internet core is unlikely to reduce. These trends clearly point to the urgent need for techniques and architectural approaches reducing or offloading complexities on the routers. In this paper, we aim to *mitigate the increasing routing complexity to the cloud* and to seek an answer to the following question: "Can techniques leveraging the memory and computation resources of the cloud remedy the routing scalability issues?"

Dr. H. T. Karaoglu was with the University of Nevada, Reno during a significant portion of this work.
This work is supported in part by National Science Foundation awards 0721600, 0721609, and 0831957

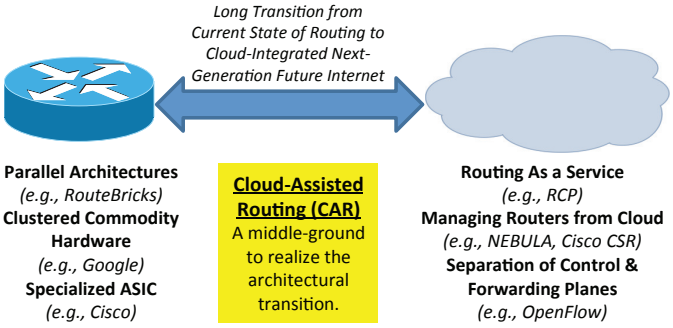

Fig. 1. Architectural Transition and Integration of Cloud Services to Routers

To address the alarming increase in routing complexity, particularly at the inter-domain level, we introduce a new architectural approach, *Cloud-Assisted Routing (CAR)*. Our key contribution is to outline a framework on how to integrate cloud computing with routers and define operational regions where such integration is beneficial. As indicated in Figure 1, a novel property of CAR is its aim to find a middle-ground between a *pure local approach* that targets to *scale* router performance (e.g., RouteBricks [7]) and a *completely cloud-based approach* for seamless and highly *flexible* routing services (e.g., CSR [10]).

The idea of using nearby computing resources for improving a router's performance is not new [7]. "The cloud" is getting "closer" thanks to availability of more sites allowing richer geo-diversity and faster access to the cloud services. Therefore, it became possible to make pragmatic comparisons among cloud providers and select the best one fitting to one's particular needs [11]. The latency to the closest cloud provider and response time on various types of computation and storage tasks can be at sub-second levels [11]. In this context, we believe that the concept of using cloud services to relieve routers' complex duties presents a great opportunity. Such integration of cloud services to router platforms will also enable intra- and inter-domain optimizations by exploiting the "central" role of the cloud.

Though moving networking functionalities to the cloud is an inevitable trend [11], [12], delegating all routing functions to the cloud has credible risks in exposing critical routing services to potential cascading failures [13]. We believe that hybrid approaches that maintain high priority tasks at the router and employ an adaptive cloud-router integration framework have more likelihood of addressing future routing scalability and flexibility tradeoffs, which Figure 2 attempts to illustrate. In that sense, CAR follows an opportunistic

978-1-4673-5753-1/13/$31.00 ©2013 IEEE

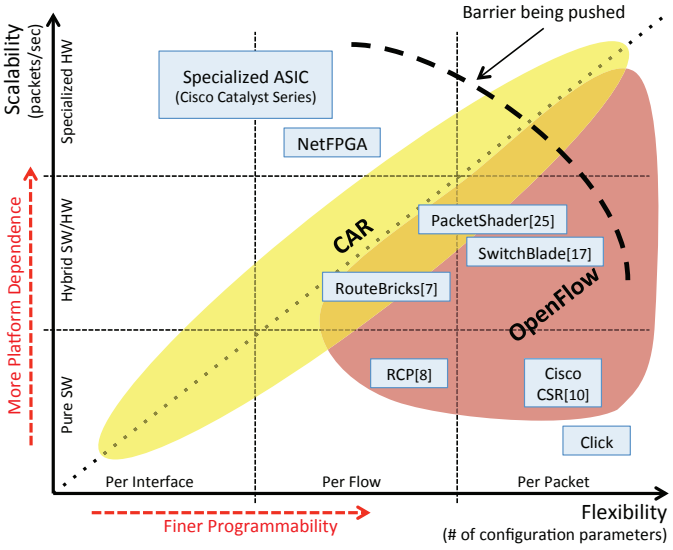

Fig. 2. Scalability-Flexibility Tradeoff for Routing

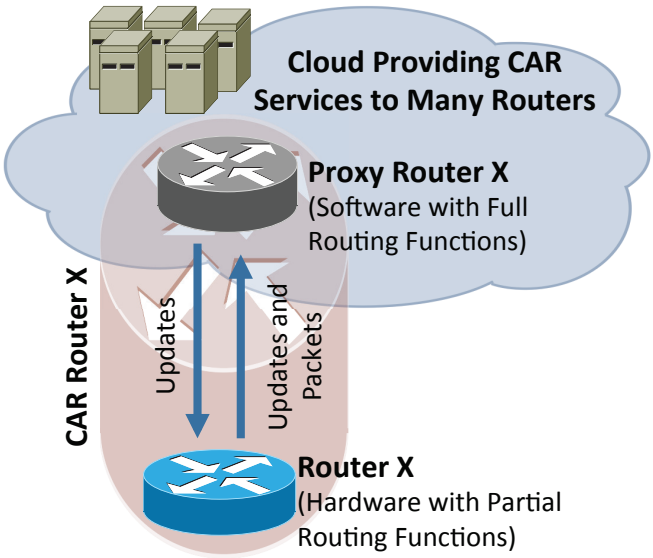

Fig. 3. CAR Architectural View

model, where routers exploit cloud resources whenever they are available and beneficial. Such hybrid approach with partial dependence on the cloud resources will allow routers to quickly retake responsibility of delegated tasks in case of cloud-related connectivity or performance problems.

The rest of the paper is organized as follows: Section II covers recent related work. In Section III, we lay out CAR's architectural motivation, view, and principles. Then, in Section IV, we present a sample case on how one may apply CAR on a key routing problems: Reduction of large CPU bursts at BGP routers. Beyond this sample case, CAR allows many new optimizations both at the inter- and intra-domain routing. Finally, we summarize our work in Section V.

II. Related Work

The Internet's routing has grown to be a complex, customizable service which cannot be left to the routers alone [8]. The concept of *"Routing As a Service"* (RaaS) [9] implies the separation of control and data planes where routing decisions are made and executed. Such a separation can be very beneficial where control plane tasks are delegated to "clouds" which offer as vast computational power, storage and parallelism required by the enlarging and diversifying routing problem. Path calculation with respect to multiple distance metrics (e.g., bandwidth, latency, loss rate, price) on wide-range of possible IPv6 address space would be challenging on existing routers with their limited capacities [14]. Parallel router architectures [7], network processors [15] or GPU-empowered routers [16] can mitigate this complexity temporarily. However, it is still a question whether these approaches can prepare routers for the next billion of Internet users with more challenging application traffic requirements.Yet, *cloud computing offers easily extendable capacity that may address these challenges.*

Routing tasks consume much of the resources on current routers [3]. These operations are inherently compute-heavy tasks which are usually not being computed in an online manner. Delegation of these computation- and memory-intensive operations to the cloud is the key inspiration of our work. Classification of these tasks carried out by a router as "delegatable" or "in-place" will be a first step towards delegating some of them to a cloud-based control plane. Then, released resources by offloaded delegatable tasks can be reclaimed by data plane to offer enhanced in-place services. Location-based characteristics of these in-place (or in-situ) services such as security, traffic management and monitoring require them to be executed on routers instead of a remote location, e.g., the cloud. Along with the delegation of routing tasks and simplified architecture of routers, these network entities can be better designed to support virtualization and programmability as suggested earlier [17]–[19].

III. Architectural Considerations

An architectural view of CAR includes a legacy hardware router with partial routing functions and a software router with full routing functions, which together establish a hybrid "CAR Router" as illustrated in Figure 3. Similar to how virtual memory systems use secondary storage to maintain the full content of the memory, CAR uses the cloud to implement the full functionality of a *Router X*, and keeps *Router X* as 'active' while *Proxy Router X* as 'passive'. The software *Proxy Router X* holds the full forwarding tables; and is the default point of service for data and control plane functions that cannot be handled at the hardware *Router X*. We anticipate that some of the control plane operations such as on-demand route computations due to failures will still be triggered by the hardware *Router X*. However, CAR will host heavy routing optimizations at the *Proxy Router X*.

A. Control Plane

CAR's approach to scaling computational complexity of routing is to delegate control plane functions to the cloud to

the extent possible. Though earlier proposals aimed to exploit parallelism in routing by modularizing a router into many parallel working nodes [7], cloud computing offers extensible resources beyond what can be offered locally. Further, many routing problems require on-demand, large-scale computations (e.g., traffic engineering and calculating backup paths) which naturally fit to the CAR's approach of having an active hardware router.

B. Principles

CAR's key goal is to "mitigate the increasing routing complexity to the cloud." We suggest that, following the Amdahl's Law, CAR treats the router hardware as a precious resource and thus should focus on the most frequent or important routing functions in the router and offload the rest to the cloud. Particularly, the following two principles should be followed when applying CAR to a routing problem:

CPU Principle: *Keep Control Plane Closer to the Cloud.* CAR designer should offload heavy computations to the cloud as much as possible. Example of such heavy but not-so-urgent control plane computations include BGP table exchanges, full-fledged shortest-path calculations, and various traffic engineering optimizations.

Memory Principle: *Keep Data Plane Closer to the Router.* CAR designer should keep the packet forwarding operations in the router to the extent possible. An example conformation to the memory principle is to handle most of the forwarding lookups by maintaining a copy of heavily used prefixes at the router memory and delegate the rest of the lookups to the cloud where the complete set of prefixes is held.

C. Function Placement

The fundamental technical challenge a CAR designer has to tackle is to place the routing functions appropriately. Figure 4 shows the overall picture the designer faces. The full router functionality is to be placed in the Proxy Router located at the cloud. The key question is how much of the router functions should be kept available in the router so that the overall router performance experienced by the incoming traffic is acceptable. In general, placing some of the router function at a remote location like the cloud will degrade the performance. If this placement is done well, then it is possible to serve more than 95% of the traffic [20] at the (hardware) router without having to bother with the Proxy Router at the cloud.

To-delegate or not-to-delegate. Due to the extra delay coming from delegation to the cloud, CAR designer's key metric to decide how to place functions to be delegated to the cloud is the *cloud-router delay*, t_{CR} in Figure 4. Intuitively, if t_{CR} is too high, the designer should not delegate to the cloud and keep more routing functions at the router to achieve higher overall efficiency. But, for a limited hardware router, other factors such as rates of traffic flows and a router's buffer size will play role in identifying which flows to delegate. CAR's contribution here is a framework for exploring efficiency-fairness tradeoff, potentially on a flow-based manner.

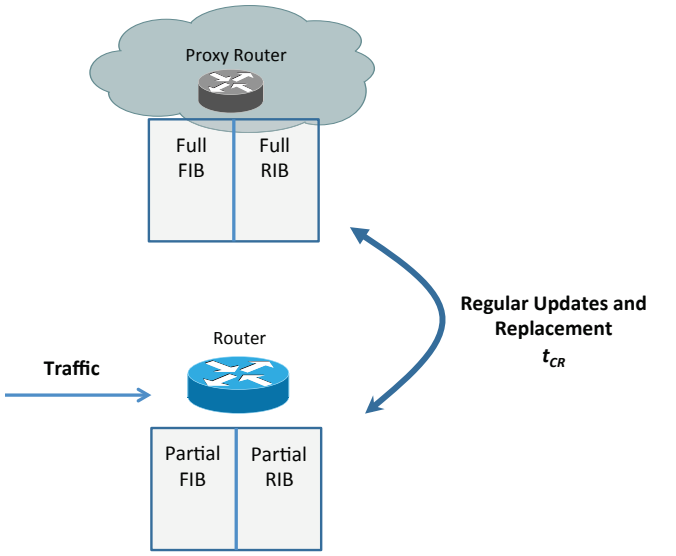

Fig. 4. Function Placement for a CAR Router: Router hardware acts like a cache to the full router function located at the cloud.

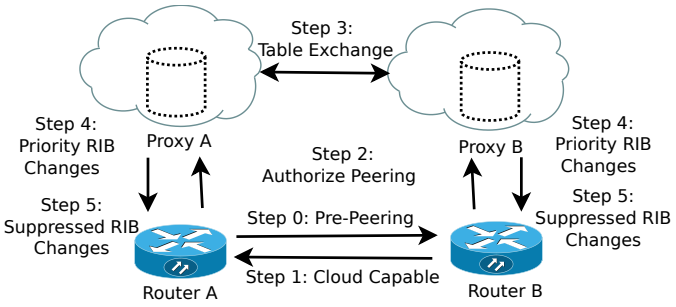

Fig. 5. Peering Establishment Scenario

Adaptive tuning to exploit locality patterns in traffic. The key indicator for fruitfulness of CAR is whether it is possible to achieve a similar performance with smaller router hardware resources like memory. Just like the virtual memory does not pay off if there is no locality, CAR will have to actively leverage the locality patterns in traffic to yield benefits over the existing router designs. This requires adaptive tuning of caching and delegation of router's functions for different traffic patterns and situations. The benefit of CAR is going to be highly dependent on the effectiveness of this adaptive tuning. We anticipate that such adaptive tuning will not be hard to do given the locality and regularity of the traffic patterns.

IV. Sample Scenario: CPU Burst for BGP Peering

BGP peer establishment (PE) is a CPU-intensive process which can leverage CAR, whose basic idea is to exploit cloud-services for performance gain without introducing a hard-dependency on cloud-availability. The PE process requires full-table exchange among neighbors, and best-path selection algorithm is applied on received routes for each prefix entry during the process. Considering the RIB table size of a default-free zone (DFZ) router (≈400K) and the PE between DFZ routers, path-selection process produces a small subset of

routes as primary paths from peer while installing remaining routes as backup paths. In fact, for representative BGP table dumps from RIPE [21] route servers, even peering among routers at distant locations results in selection of 2.5% or less of advertised routes as best paths (see Fig. I).

To improve the amount of traffic and CPU usage during the PE process, there are many proposed schemes [22], [23], which allow a peer to dynamically express its interest on advertisements of specific prefixes instead of full-table exchange without introducing significant convergence delays. These mechanisms are particularly useful for enterprise network scenarios where border gateways may be interested in routes for a few prefixes due to their policies while accepting default route from its provider for the rest. In case of DFZ routers, though, it is not known in advance which paths should be selected as primary routes during PE.

Building upon this observation, we develop a two-phase PE mechanism. As depicted in Figure 5, prior to PE, during the capability exchange phase, routers will inform their peers if they are CAR-capable by exchanging the address of their proxies at the cloud(s). Second step is the authorization of PE between the proxy routers, which keep exact copies of the routing and forwarding state of actual routers and will carry out tasks related to full table exchange and BGP route selection processes. Then after, the proxy routers will classify the changes on routing and forwarding states as 'priority' and 'non-priority'. So, if the selection of a route leads to forwarding-path changes, this entry will be marked as priority. Only routes marked as priority will be exchanged between the actual routers, filtering out all other entries for the first phase of the PE. Consequently, the actual router only has to consume its computational resources on significant updates while suppressing updates related to backup paths for a later phase (or eliminating all together relying on explicit route-refresh mechanisms). Such a scheme can i) significantly shorten the amount of time required for PE via parallelization mechanisms [7], ii) reduce the CPU bursts on peering routers, and iii) reduce BGP control traffic.

For proof-of-concept, we developed a simple prototype of BGP PE process with CAR in the Emulab testbed. In a dumb-bell topology setting, using route injectors, we advertise routes to two Quagga routers running on Dell 2850 servers with Intel Dual Core 3 GHz processors and 2 GB DDR2 SDRAM. These route injectors also generate traffic and replay BGP update messages at constant rates. For three separate BGP router pairs, we gathered BGP table and update message dumps from RIPE and Routeviews [21] for the day of November 8, 2011. We also used aggregated IPv4 bogon filters from the same resources as inbound/outbound prefix filters.

Once the route injectors advertise their full RIB, we initiated a peering between the two Quagga routers and collected CPU utilization records on them. Finally, matching BGP logs and the CPU records, we calculated the duration between PE initiation and the time BGP reaches its convergence state. To emulate the steps 4 and 5 of our CAR PE process, we developed an Expect script that downloads a pre-calculated

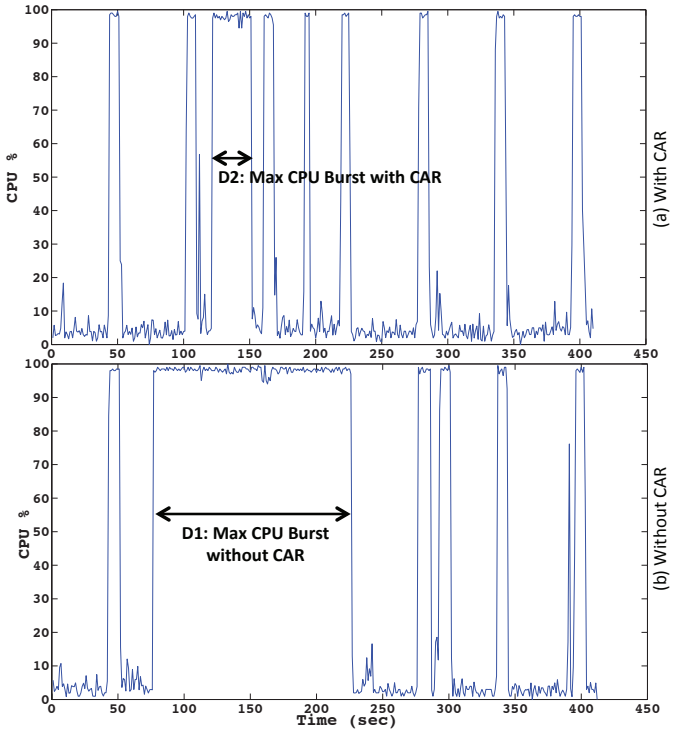

Fig. 6. CPU Bursts of DECIX Peering LINX

prefix-list generated by the proxy router of the peer and installs this prefix-list as outbound filter for eliminating all the routes except the ones selected as best paths (i.e., 'priority') by its peer. Then, we initiate the actual PE and calculate the duration of CPU bursts. We repeated this experimental PE process three times for each BGP router pair. For the three BGP router pairs, the average of the three maximum CPU bursts with (D2) or without (D1) CAR are reported in Table I.

In Figure 6, we plot CPU utilization gathered for setup with and without CAR comparatively for the LINX-DECIX BGP router pair. The large CPU bursts are due to BGP Router Thread which is responsible for processing updates whereas short, periodic CPU burst are generated by BGP Scanner Thread. To ensure short convergence times, BGP Router Thread is given all the free cycles available which may cause CPU utilization bursts for several minutes. As can be seen in Figure 6 and Table I, CAR can significantly reduce these CPU bursts which may cause unresponsive state for other routing protocols running on the router hardware as well as packets requiring optional forwarding. This proof-of-concept prototype is a first step towards addressing many challenges in offloading CPU-intensive tasks to the cloud using CAR. Due to limitations in the Emulab environment and RIPE datasets, our prototype had small traffic rates for each interface (below 1GB/sec), small route-to-prefix ratio ($\approx$1.82), and limited inbound-outbound filtering. In a more real setting, the CPU load on a BGP router is much higher, and thus, the benefit of CAR will be more pronounced.

TABLE I
EMULAB EXPERIMENT SETUP

Dataset	Peers	RIB Prefix	Route Per Prefix	Updates (per min)	Traffic (Kbps)	Primary Prefix	D1 (msec)	D2 (msec)
ISC-EQUIX	16/17	381,455	1.84	2,664	435,200	654	146,870	28,161
AMSIX-NYIIX	10/11	384,180	1.82	3,523	281,600	2,897	141,316	28,529
LINX-DECIX	13/14	390,088	1.82	3,332	358,400	1,105	156,799	28,633

V. SUMMARY AND DISCUSSION

To address the growing complexity of routing tasks, we introduced a new approach, Cloud-Assisted Routing (CAR), that advocates *on-demand seamless integration of the cloud's computational and storage capabilities* into the Internet backbone. We outlined CAR's key components and principles and navigated its possible benefits side-by-side with implied challenges. We presented how one may apply CAR principles to two urgent problems in routers: max CPU burst size and forwarding table size.

A natural question about our proposed CAR architecture is whether or not it is economically viable. One particular argument is to use ASIC memory and more powerful CPUs instead of offloading to a remote cloud service. However, since *CAR is an opportunistic and on-demand approach*, CAR vendors will target all candidate routers, not just very high-end routers. Whether or not it will become a de-facto standard will be determined by its market penetration and standardization success, both of which involve many factors. Nevertheless, there are sufficient indicators to believe that it will draw a significant amount attention and experimentation in the community.

We see CAR as a natural successor to the RaaS concept [9], as the demand for low-price, programmable commodity network/router hardware is likely to be a driving force for a service-oriented, software-based vendor market. CAR, which embraces commoditization of router hardware by mitigating complexity from the hardware to the cloud, can facilitate a healthy transformation to such a market.

Beyond bridging the gap between router hardware and software-based routing services, CAR allows an array of research opportunities for improving ISP backbones such as (i) resiliency to failures via cloud-based forwarding and reroute schemes, (ii) efficiency via more centralized cloud-based optimizations of intra-domain traffic engineering, and (iii) economic competitiveness via cloud-based on-demand service provisioning potentially going beyond domain borders.

CAR could be used for attaching the well-known issue of reducing forwarding table (Forwarding Information Base, also FIB) and routing table (Routing Information Base, also RIB) sizes in core BGP routers. Several studies observed temporal (bursts of packets in the same flows) and spatial (few popular destinations) locality in data packet traffic [20]. CAR can leverage these locality patterns and delegate the less used majority of the FIB/RIB entries to the cloud while keeping the more used minority at the router.

REFERENCES

[1] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB workshop on routing and addressing," *IETF Internet RFC 4984*, September 2007.

[2] L. G. Roberts, "A radical new router," *IEEE Spectrum*, vol. 46, no. 7, pp. 34–39, July 2009.

[3] Q. Wu, Y. Liao, T. Wolf, and L. Gao, "Benchmarking BGP routers," in *IEEE IISWC*, 2007, pp. 79 –88.

[4] G. Huston, "BGP growth revisited," http://www.potaroo.net/ispcol/2011-11/bgp2011.html, November 2011.

[5] A. Elmokashfi, A. Kvalbein, and C. Dovrolis, "On the scalability of bgp: The roles of topology growth, policy and update rate-limiting," *IEEE JSAC*, vol. 28, no. 8, pp. 1250–1261, October 2010.

[6] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "The flattening internet topology: Natural evolution, unsightly barnacles or contrived collapse?" in *PAM*, 2008, vol. 4979, pp. 1–10.

[7] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "Routebricks: exploiting parallelism to scale software routers," in *Proc. of the ACM SOPS*, 2009, pp. 15–28.

[8] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in *Proc. of the ACM SIGCOMM FDNA Workshop*, 2004, pp. 5–12.

[9] K. K. Lakshminarayanan, I. Stoica, S. Shenker, and J. Rexford, "Routing as a service," Tech. Rep. UCB/EECS-2006-19, Feb 2006.

[10] "Enabling Datacenter and Cloud Service Management for Mid-Tier Enterprises (White Paper)," Cisco Systems, 2012.

[11] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: comparing public cloud providers," in *Proceedings of ACM IMC*, November 2010.

[12] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," in *ACM SIGCOMM*, 2012.

[13] B. Ford, "Icebergs in the clouds: The other risks of cloud computing," in *Proceedings of the USENIX HotCloud*, Boston, MA, June 2012, pp. 453–466.

[14] D. Massey, L.Wang, B. Zhang, and L. Zhang, "A scalable routing system design for future internet," in *Proc. of ACM SIGCOMM IPv6+*, 2007.

[15] Q. Wu and T. Wolf, "Design of a network service processing platform for data path customization," in *Proc. of ACM SIGCOMM PRESTO*, 2009, pp. 31–36.

[16] S. Han, K. Jang, K. Park, and S. Moon, "PacketShader: A GPU-accelerated software router," *SIGCOMM CCR*, vol. 40, pp. 195–206, August 2010.

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM CCR*, vol. 38, no. 2, pp. 69–74, April 2008.

[18] M. B. Anwer, M. Motiwala, M. b. Tariq, and N. Feamster, "SwitchBlade: A platform for rapid deployment of network protocols on programmable hardware," *SIGCOMM CCR*, vol. 40, pp. 183–194, August 2010.

[19] S. Bhatia, M. Motiwala, W. Mhlbauer, Y. Mundada, V. Valancius, A. Bavier, N. Feamster, L. Peterson, and J. Rexford, "Trellis: A platform for building flexible, fast virtual networks on commodity hardware," in *In: Proceedings of ROADS*, 2008.

[20] C. Kim, M. Caesar, A. Gerber, and J. Rexford, "Revisiting route caching: The world should be flat," in *PAM*, Seoul, Korea, 2009, pp. 3–12.

[21] Routeviews: http://www.routeviews.org/; RIPE: http://www.ripe.net/.

[22] E. Karpilovsky and J. Rexford, "Using forgetful routing to control BGP table size," in *Proceedings of ACM CoNEXT*, 2006, pp. 2:1–2:12.

[23] E. Chen and Y. Rekhter, "Outbound route filtering capability for BGP-4," RFC 5291, August 2008.