CAR: Cloud-Assisted Routing

Prasun Kanti Dey, Murat Yuksel
Dept of Electrical and Computer Engineering
University of Central Florida
Email: prasun@knights.ucf.edu, murat.yuksel@ucf.edu

***Abstract*—We propose a new hybrid software-defined networking (SDN) approach, Cloud-Assisted Routing (CAR), that utilizes high computation and memory power of cloud services by splitting both control and data plane functions between a local router and a remote cloud computing platform. Instead of a complete separation of the two planes, our approach maintains *most* of the control plane at the cloud and the *least* of it at the local router, and vice versa for the data plane. We present the architectural view of CAR and results from an initial prototype of forwarding table size reduction using CAR. We discuss possible intra- and inter-domain optimizations by highlighting the use-cases of multi-cloud design paradigm and perform a cost comparison between legacy router vs. CAR to identify the break-even points and key components that make CAR monetarily beneficial.**

I. Introduction

Due to the perception of the Internet usage as a virtual delivery platform for almost all services and applications, its routing has grown to be a complex, customizable service which cannot be left as the responsibility of the routers alone [9]. The advent of software-defined networking (SDN) [10] attests the success of this trend of *offloading control complexity from the network to software platforms* for centralized optimizations rather than leaving them to the core components like routers. One step further, recent consensus is to do that with open interfaces, as in OpenFlow [19], [21], allowing outsourcing of control plane tasks to remote platforms that are potentially managed by enterprises other than the network operator, such as the cloud [25].

Although they offer great flexibility and programmability, software platforms may not be sufficient in addressing growing scalability concerns on routers. BGP FIB size has now exceeded 500K entries and the most conservative forecasts point to a quadratic growth of the routing table size [2]. This table size growth combined with longer prefixes (due to shrinking IPv4 space and growing use of IPv6) and line rates (45Mbps in 1998 to 100Gbps in 2013) is becoming harder to tackle. Time budget for forwarding lookups is getting more stringent due to increase in backbone speeds, and, further, more complex lookups are in the horizon due to more flexibility and friendliness expected from network routing as in NDN [29]. Although FIB compression and aggregation efforts [30] have promised 60% reductions in SRAM requirements

Authors were affiliated with University of Nevada Reno when most of the work has been done.
This work was supported in part by NSF award CNS-1321069.

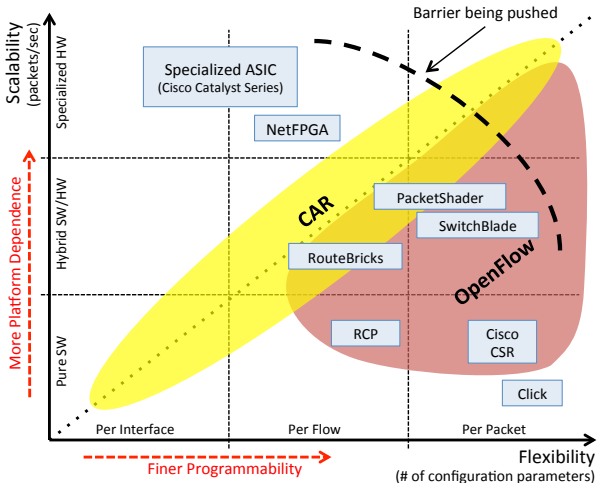


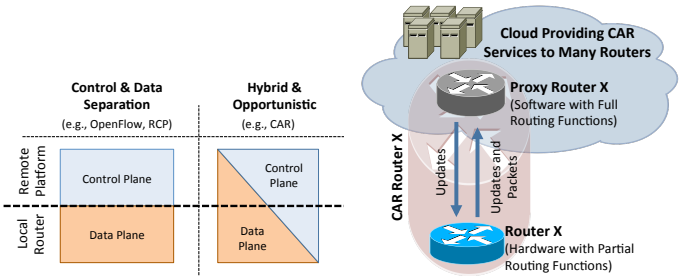
Fig. 1: Routing in terms of scalability-flexibility tradeoffs

[23], it is questionable these efforts *alone* can continue fitting the growing routing tables into legacy SRAMs while matching the shrinking lookup budget.

Cloud-based management is economically attractive. The emergence of cloud computing services with high flexibility and virtualization has made the networking community consider offloading networking services to the cloud [25]. Cloud storage prices have been declining 3-10% a year [24] (as our study in Figure 4a also supports), and this reduction makes it an attractive platform for long run. Market indicators point to "commoditization" of cloud prices [3], which, arguably, will never be costlier than the specialized ASIC of routers.

Cloud is getting closer and higher quality. The latency to the closest cloud provider and response time on various types of computation and storage tasks can be at sub-second levels [18]. Further, getting service from multiple cloud providers (i.e., "hybrid" cloud) has already become a practice.

Hybrid and opportunistic routers. Moving *all* management and control plane tasks to cloud has picked up pace [25], [18] mainly due to its elegant flexibility in network administration, even though it may be at the price of degrading the performance and the risks of exposing critical routing services to potential cascading failures [11]. We think, hybrid approaches that maintain high priority tasks at router with *partial* offloading to remote platforms and employ an adaptive cloud-router integration framework will be more likely to



(a) Separation of planes (b) CAR architectural view

Fig. 2: CAR architecture as a new SDN approach

address future routing scalability and flexibility trade-offs (Figure 1).

To address the alarming increase in routing complexity, particularly at the inter-domain level, we introduce a new architectural approach, *Cloud-Assisted Routing (CAR)*, that leverages high computation power and memory resources of cloud services for easing complex routing functions such as forwarding in the data plane and flow level policy management in the control plane. Our key contribution is to outline a framework on how to integrate cloud computing with the routers and define operational regions where such integration is beneficial.

The rest of the paper is as follows: We, first, detail CAR architecture and its function placement principles in Section II. Section III discusses the potential usage of CAR and presents our modeling work on CAR cost versus legacy routing cost to show CAR savings with regards to memory size. Section IV presents our initial prototypes of CAR and its evaluation considering the reduction of FIB table size. Finally, we summarize our work and discuss future work in Section V.

II. CAR Architecture

A. Architectural View and Principles

An architectural view of a hybrid "CAR router" which combines a legacy hardware router with partial functionality and a software router with full functionality is illustrated in Figure 2b. Similar to how virtual memory systems use secondary storage to maintain full memory content, CAR uses cloud to implement the full functionality of local *Router X (RX)*, and keeps *RX* as 'active' while *Proxy Router X (PRX)* as 'passive'. The software *PRX* holds the full forwarding tables and performs the full routing computations, and is the default point of service for data and control plane functions that cannot be handled at the hardware *RX*. We anticipate that some of the control plane operations such as on-demand route computations due to failures or collection of flow-level simple statistics will still be done at *RX* similar to how recent OpenFlow switches account for the flow statistics and send them to controller. However, CAR will host heavy routing optimizations at *PRX*.

As indicated in Figures 1 and 2, CAR aims to find a middle-ground between a *pure local approach* that targets to *scale* router performance and a *completely cloud-based approach* for seamless and highly *flexible* routing services. Rather than establishing a clean separation of control and data planes, CAR argues for a hybrid opportunistic approach (Figure 2a). Particularly, CAR places most of the heavy control plane functions (e.g., peering establishment) in the cloud while the most important and urgent control functions (e.g., response to failures) stay in local router. Intertwined with this control plane placement, most of CAR's data plane stays at the local router while a small portion of the data packet forwarding gets delegated to the cloud.

1) Control Plane: To scale computational complexity of routing, CAR advocates for delegating control plane tasks to cloud to the extent possible and perform large-scale routing computations by exploiting the cloud's cheap parallelism speedups as well as centralized role in traffic engineering. Though earlier proposals aimed to exploit parallelism in routing by modularizing a router into many parallel working nodes [8], cloud computing offers extensible resources beyond what can be offered locally. Yet, many routing problems require on-demand fast computations (e.g., calculating backup paths) which naturally fits to the CAR's approach of keeping a small portion of the control plane at *RX*.

2) Data Plane: Commercial routers provide highly optimized data plane implementing forwarding operations in extremely fast ASIC circuits and forwarding tables in custom TCAM memories. Many existing proposals suggest more programmable packet classification and forwarding function definitions at data plane [5]. For CAR also, packet classification, flow descriptions and corresponding forwarding actions (e.g., traffic shaping, DiffServ mechanisms, packet filtering, provisioning) are defined as "movable states" computed by control plane. CAR's data plane consists of virtualized low layers, movable upper layer states, and forwarding components used to compose both hardware and software level customizable data plane functions. In comparison to legacy SDN approach to routing, CAR's key difference is to move a small portion of the data plane forwarding actions to the cloud rather than keeping all of them at *RX*.

3) Principles: Following the Amdahl's Law, CAR treats the router hardware as a precious resource that should only focus on the most frequent or important routing functions and offload the rest to the cloud. Particularly, the following two principles should be followed when applying CAR.

CPU Principle: *Keep Control Plane Closer to the Cloud.* RX should be designed in such a way that it can offload computation intensive but on the same time not-so-urgent control plane tasks (such as BGP table exchange, full-fledged shortest-path calculations for traffic engineering optimization) to cloud to the extent possible. Our early prototype [12] attained 5 times reduction in CPU utilization burst size while establishing a BGP peering by exchanging a very small portion of table at RX and the full exchange at PRX.

Memory Principle: *Keep Data Plane Closer to the Router.* RX should be designed to handle most of the packet for-

warding operations locally. CAR designer can follow this memory principle simply by periodically identifying the most frequently used prefixes and storing a copy of them in router memory to deal with most of the forwarding lookups. If this can be optimally designed, RX will have to delegate few lookup requests to PRX which should keep the entire set of prefixes and will act as the default point of failure.

B. Function Placement

The fundamental technical challenge a CAR designer faces is how to place the routing data and control plane functionality between *RX* and *PRX*. In general, placing some of the router function at a remote location like the cloud will degrade the router's performance, i.e., the overall speed. Yet, with strategic placement, it is possible to keep the overall performance while handling much larger loads. For instance, by storing only 25% of FIB, *RX* can serve 99% of the traffic [14] without having to use *PRX* at the cloud, which means better performance. If the lookup at partial FIB table at *RX* is a miss, the packet will either be cached or delegated to *PRX*. If the *cloud-router delay*, t_{CR}, is small (e.g., <50ms) or the local buffer is too small to temporarily keep the packets of this flow, it is best to delegate the packet to *PRX*. But, if it is imminent that many more packets will arrive on this flow and continuos delegation of it to *PRX* will make too many customer packets to experience increased delay, it may be best to resolve the miss in a manner similar to the page fault handling (OpenFlow controllers follow this approach) and still process the data traffic at *RX*. Implementation of such router-cloud integration will require a more detailed analysis involving all the key parameters, such as t_{CR}, *RX* buffer size, flow rate, cloud response time, and priority of the flow.

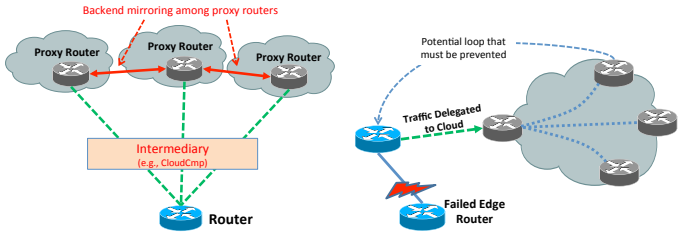
To-delegate or not-to-delegate. In general, if t_{CR} is too high, CAR designer should not delegate to the cloud and keep more routing functions at the router to achieve higher overall efficiency. For a limited hardware router, other factors such as rates of traffic flows and router's buffer size will identify which flows to delegate. The flows delegated to cloud will receive service with larger delays and higher loss; and thus, keeping a fair treatment of data flows is a challenge CAR faces.

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. This requires adaptive tuning of caching and delegation of router's functions for different traffic patterns and situations. Just like the virtual memory does not pay off if there is no locality, the benefit of CAR will highly dependent on the effectiveness of this adaptive tuning.

III. CAR Potentials

A. Establishing Robustness via Multi-Cloud Designs

Migrating routing functions to the cloud has sizable risks due to uncertainty of the cloud response times and hazy reliability to the cloud services in general. Yet, it may also help in improving the reliability of the router hardware as well. We discuss a few such scenarios.



(a) Multi-cloud design for robustness to cloud failures or delays (b) Traffic reroute using cloud-based forwarding

Fig. 3: Reliability improvements by using CAR

1) Picking the Best Cloud: Recent work by Li et al. [18] showed that it is possible to pick the best cloud provider for one's location for different application-desired metrics such as response time or service price. This presents a great opportunity to help our CAR architecture via consideration of multiple clouds to establish reliability in *PRX*. As shown in Figure 3a, we can mirror *PRX* at multiple cloud providers and dynamically select the "best" one over time and the task at hand. An intermediary similar to CloudCmp [18] could be designed to cope with the variations in the cloud service quality. For example, when a packet is to be delegated to *PRX*, such an intermediary can balance various parameters such as (i) the importance of the traffic flow the packet belongs to, (ii) closeness of the candidate clouds to the destination of the packet, and (iii) the service prices of the candidate clouds.

The intermediary could be implemented at *RX* or a computer/server in proximity of *RX*. It may even be possible to migrate this intermediary to a cloud provider that is willing to provide certain performance assurances in response time, which of course would entail costlier pricing. This type of multi-cloud framework requires efficient mirroring implementation of *PRX*s among multiple clouds, which could be done with legacy mirroring protocols. This multi-cloud approach provides opportunities beyond failure management and allows various optimizations among the backend cloud providers depending on the traffic characteristics, e.g., its destination or priority. Existing SDN implementations are looking for solutions to increase the reliability of the SDN controller and the link(s) between the hardware routers and the controller [10]. If significant portion of the control plane can be migrated to multiple third-party cloud providers over a public Internet connection, vulnerability of the controller and the router-controller link to failures and attacks can be mitigated.

2) Failure-Triggered Traffic Delegation to PRX: In case of a major link failure, significant amount of traffic should be rerouted without deteriorating the service quality levels for the remainder of the network. There has been extensive research on analyzing link failures, redundancy/over-provisioning models and architectures to achieve greater network resilience [22], [27]. Similarly, upon a failure on a link to a neighbor router, *RX* can forward/delegate the affected traffic (which would normally go to the neighbor) to *PRX* as shown in Figure 3b. Such delegation could potentially be done in a manner

seamless to other routers. However, potential inconsistencies will have to be considered and prevented. For example, *PRX* should not be somehow forwarding the traffic back to *RX*. One simple way of assuring this does not happen is to maintain a separate lookup table at the cloud node where *PRX* is located and check if *PRX* is the next hop for the destination prefix in question – in a manner similar to BGP's poisoned reverse. Yet, this only solves one-hop inconsistencies and consideration of policy issues will make this task more complicated. However, it will be possible to manage these complications by carefully organizing which prefixes are to be allowed for such failure-triggered traffic delegation. In general, delegating data plane forwarding tasks to a remote platform is a new feature being introduced by CAR and needs reworking of some of the existing configuration and policy practices.

3) Migration of CAR Routers Via Movable States: Another interesting aspect of CAR is to decouple lower layer of network configuration with the states of upper layers by migrating some of the control and data plane functions to a third-party cloud service. This separation enables definitions of movable soft states which can define forwarding information for flow descriptors, service provisioning for differentiated services, security settings or even enhanced packet forwarding functions. One perspective of this "CAR Router state" is the mapping between these lower layer configurations and upper layer state. Currently, virtualization and tunneling technologies offer wide-range of flexibilities, as also required for achieving our CAR architecture. Telecommunication vendors also have various consortiums for developing common standards and interfaces to enable such mechanisms, e.g., [1]. Also, significant research work has been done on virtualized service mirroring and migration [13], [26], [6] especially for virtual machines. However, these technologies are still bound to the limitations of both hardware and software protocols.

Once feasible, such decoupling can be used to achieve "movable states" of provisioned services or forwarding plane among virtualized hardware. This can become especially important for Virtual Network Operators (VNOs), switching infrastructure providers and infrastructure owners to quickly recover after node failures. CAR architecture can provide quick access to the cloud storage and computation capabilities to compute, store and retrieve these movable states. Such virtualization of router states will be tremendously helpful in configuring, maintaining and optimizing large networks.

B. Intra- and Inter-Domain Routing Optimizations

When CAR attains sufficient penetration and gets deployed by many autonomous systems (ASes) and cloud providers, several routing optimization opportunities will emerge. Cloud providers will, then, be able to perform centralized optimizations as they will be offering cloud assistance services to multiple routers in the same AS and routers belonging to different ASes. We outlined a few of such cases where CAR will be very beneficial in optimizing existing routing.

1) Shortcuts Through Cloud: One obvious benefit of delegating some of the packets to *PRX* is that the packet could be easily forwarded to the next hop on its path if the next hop router is being served by the same cloud provider. This situation can easily arise among *PRX*s belonging to the same AS, since it is very likely that routers of an AS will be receiving cloud assistance service from the same provider. Inter-AS shortcuts can be taken if the cloud vendor is providing service for routers from different ASes. Recent work showed benefits of such cloud-based data transfers to higher level performance as well [25], [16], [28]. These optimizations become possible with CAR because data plane functionality is delegated to clouds. Legacy SDN designs strictly keep data plane to local router and only place control plane to a remote platform, and this clean separation of control and data planes disallows multi-hop optimizations within the data plane.

2) Traffic Engineering: Having control plane of many routers located in the same cloud, it becomes possible to implement traffic engineering techniques much more advanced than today's. Existing traffic engineering involves long timescale (e.g., days or weeks) re-configurations of routers and intuitive automated responses to failures. Complex load balancing tasks are done by human admins and require heavy computations of how to set layer 2 (e.g., MPLS) paths. With CAR, the cloud provider will be able to observe many routers at the same time and perform short timescale (e.g., tens of minutes) engineering of the traffic by using abundant computation and storage of clouds. It will be reasonable to run machine learning algorithms to detect emerging patterns in the traffic at short timescales and re-configure *PRX*s (and in turn *RX*s) accordingly. Similarly, in inter-domain traffic engineering, some of the pitfalls (e.g., misconfiguration of MED) could be prevented if ASes agree to an "inter-domain manager" offered by the cloud provider.

3) Cloud as An Exchange Point or Mediator: The performance of packet forwarding depends heavily on inter-AS negotiations. Mahajan *et. al* [20] explored a negotiation framework *Nexit* based on stable and efficient routing among neighboring ISPs. It is observed, if ASes share little information with each other, optimizing the routing becomes much easier as both of them would have a more comprehensive knowledge of the network. Kotronis *et. al* [15] proposed a routing model where control plane is logically centralized and managed by a trusted third party. According to them, the more ASes will delegate their control plane tasks to this third party, who has expertise in routing functionality and has a well-trained infrastructure to offer better management, the easier it will be to provide efficient reconciliation among these ASes. As of now, we can confidently claim, cloud can be treated as the best third party to offer such a centralized scheme.

Exploiting birds eye view of multiple ASes topology, cloud provider can easily detect and resolve policy conflicts, monitor and troubleshoot inter-domain routing problems and even come up with ingenious design solutions. This habit of delegating partial control logic will essentially create clusters in cloud. At this point, we want to make sure that AS-cluster will be different from AS-confederation (defined in RFC 5065). Here, AS-cluster will compose those ASes who

are willing to trust their internal policy with cloud provider while preserving the right to manage their own privacy and integrity, business identity and policy-shaping capability. While delegating control plane functionalities to cloud, it is assured that, cloud provider will correctly map individual AS's Service Level Agreements (SLA) into policy, enforce scalability issues and monitor misconfiguration. Cloud provider will also be responsible for keeping the confidentiality of AS specific topology, network loss, delay and bandwidth utilization.

C. CAR Economics

In our earlier work [7], we have analyzed the economic viability of CAR. Exploring the price trends for DRAM (for RX storage), cloud storage (see Figure 4a) and transit cost incurred due to the delegated traffics to cloud, we propose two naive cost models:

$$C = dF \tag{1}$$

$$\hat{C} = dFl + cF + z\rho(l) \tag{2}$$

Where C is the cost for traditional routing and $\hat{C}$ is for CAR. d and c are the costs of \$/bit storage at the local DRAM and the cloud over one unit of time, respectively. z is \$/bit cost for transferring packets delegated to the cloud. Furthermore, F is the FIB size (in bits), l is the percentage of FIB size that needs to be stored in the local router to sustain an acceptable (i.e., close to traditional routers) average delay for packets forwarding time. ρ is a function of l which calculates the number of packets delegated to cloud.

We want to emphasize that second and third terms in equation 2 are negligible since the price of cloud storage is very small (price drops exponentially as shown in the Figure 4a) and the amount of traffic delegated towards cloud will be fewer as l increases. Prominent factor to be considered here is the first term that involves DRAM cost which faces some unexpected ups and downs rather following a consistent price decay as found in our earlier work [7].

We argue on the break-even point between DRAM cost and cloud storage cost depicted in Figure 4b advocating that CAR will be economically beneficial than legacy routing if the price trend of these two commodity continues to follow. As portrayed, after 2013, cloud storage is cheaper than DRAM cost which supports our claim.

Figure 4c demonstrates that router memory (i.e., FIB cache) size can be reduced to attain more savings by using CAR. An interesting observation here is that the reduction needed in FIB cache size is in power law relationship with the desired improvement in cost savings, i.e, to make CAR 10 times more cost-efficient, around 10 times reduction in the FIB cache size is needed. This is a very promising result and shows that more savings are possible in CAR as long as the engineering cost of reducing the FIB cache size is justified.

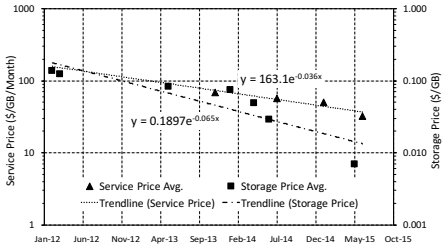
IV. Reducing Forwarding Table Size Using CAR

A well-known issue with core BGP routers is their forwarding table (FIB) and routing table (RIB) sizes. Several studies observed temporal (bursts of packets in the same flows) and spatial (few popular destinations) locality in data packet traffic [14]. This means, even though most of the destinations will be looked up very infrequently, they will keep occupying the routing table. A key motivation for CAR is to leverage these locality.

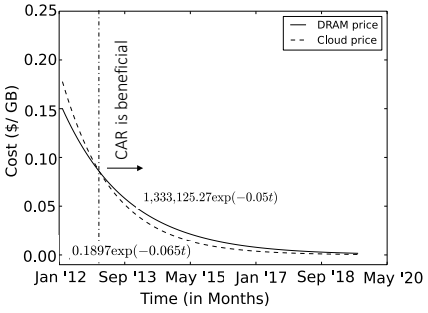
The idea is to store only partial FIB at the *RX* and delegate packets to the *PRX* if a miss occurs during the lookup at the partial FIB. The *PRX* will store the full FIB, and thus will be able to handle any misses at the *RX*. As shown in Figure 5a, one can implement this relationship between *RX* and *PRX* via tunnels or dedicated TCP sessions. CAR handles FIB lookups in a hierarchical manner, as shown in Figure 5a. For instance, some packets will be handled completely via hardware lookups (e.g., packets destined to 8.8/16 in the figure), some via software lookups at the router (e.g., destined to 72.24.10/24), and some via lookups at the cloud (e.g., destined to 72.36.10/24). Similar to traditional cache organizations, the lookup will be delegated to one level up in the hierarchy if a miss occurs. In general, the placement of prefix entries to the different levels of this CAR framework is not an easy task as explored previously in different contexts [14]. It involves several dynamic parameters such as lookup frequency of prefixes and importance of prefixes due to their contractual value. The positive factor is that high locality patterns exist in these parameters. Delegated prefixes will suffer from additional delays, and a key goal will be to establish an acceptable fairness across prefixes.

To make some initial observations, we have developed an Emulab prototype of the concept in Figure 5a using a two-level hierarchy, one is the hardware lookup at the *RX* and the other is the software remote lookup at the proxy. Although it is possible to implement existing FIB caching techniques in CAR, we prototyped a "crystal ball" approach where the router is assumed to know the future packets. At the start of each second, FIB cache is updated with the most frequently used prefix entries in the next second so that the router always knows which packets are coming. This design gives us an understanding of the best possible gain from CAR in terms of FIB size reduction.

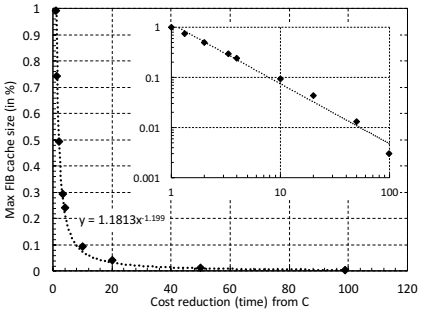
Figure 5b describes our experimental setup for the initial CAR router prototype with three next-hops. To ease the measurements, we used one destination router. We used anonymous traffic traces taken in 2014 by CAIDA [4]. For routers, we used Quagga running on Dell PowerEdge 2850s with a single 3GHz processor, 2GB of RAM, and two 10,000 RPM 146GB SCSI disks. Each router had four available gigabit net interfaces. Links were configured as duplex with standard DropTail queues of size 100 Mb and a loss probability of 0.01. To feed the routing entries, we identified the subnets of all packets and assumed them to be of /24. We are using /24 subnet prefixes based on the observation [14] from RouteView Traces during Nov 2001 to Jul 2008. /24 prefix count increased from 60*K* to 140*K*, while /16 increased 7*K* to 10*K* and /8 increased from 18*K* to 19*K* which shows a much quicker increase compared to other prefix lengths. Then we took top



(a) Cloud storage (for 3 providers) vs. Service (for 11 providers) cost

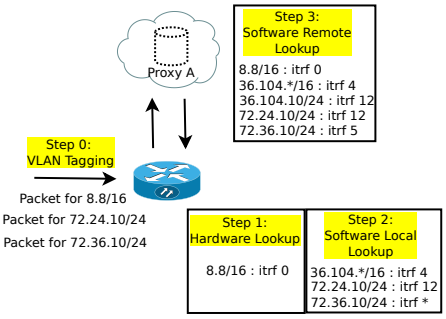


(b) Cloud storage price vs. DRAM price

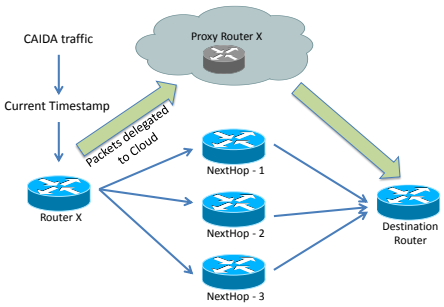


(c) CAR savings and subsequent FIB reduction

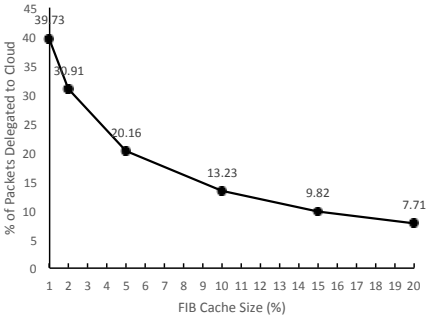
Fig. 4: Cost comparison and Break-evens



(a) Forwarding table lookup



(b) Emulab setup



(c) Packets delegated to cloud

Fig. 5: FIB table lookup scenario using CAR

x% of the /24 prefixes with the highest frequency and varied x from 1, 2, 5, 10, 15, 20. We feed these most frequent subnets to the FIB table located in *RX*, while *PRX* had the entire list. Note that the cache hiding does not exist in this case as all prefixes are /24.

Figure 5c shows the percentage of packets being delegated to the cloud proxy as the FIB cache size at the local router increases. It is clear that, the trend is going downwards and if we use 20% of entire FIB table as cache, it will miss only 7.7% of all the packets. Yet, the gain is not as high as expected since earlier measurement studies showed that 10% of FIB accounts for more than 97% of the traffic and 1% accounts for more than 90% of the traffic [14]. In comparison, our crystal ball approach could only process about 87% of the traffic when FIB cache size was 10% and 60% of the traffic when the cache was 1%. This is mainly because of the delay involved in updating the local router's FIB table. The significant insight however is that the loss in the gain because of this delay reduces significantly is FIB cache size is kept above 15%.

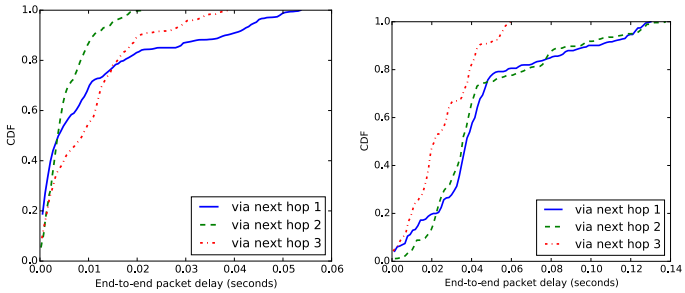
Table I shows the number of packets forwarded via individual next-hops and time it took for those packets to reach the destination router if the FIB cache size is only 1% of the entire FIB. It also shows the number of packets delegated towards the cloud. Couple of observations from this table are % of packets drop towards the next hops are quite high 4.72 to 4.96. The trend was same for rest of the experiment as well.

TABLE I: FIB Cache Size 1% (Emulab Setup)

NextHop	Packets Sent	Total Packets (%)	Packets Dropped (%)	Avg. Delay (ms)
NextHop-1	7,949,903	20.79	4.72	17.7
NextHop-2	7,670,496	20.06	4.74	13.8
NextHop-3	7,426,526	19.42	4.96	13.7
Cloud	15,191,706	39.73	-	-

This is because of the smaller size of buffer as well as the DropTail technique used in the Emulab machines. Figure 6a shows the CDF of time required for each packet to reach the destination, which shows a minimal unfairness. Most of the packets arrive in similar timings.

Figure 6b plots the CDF of packet delay for FIB cache size 10%. We observe an increase in the end-to-end delay when FIB cache size is increased, which is counter-intuitive. The reason for this increase in the end-to-end delay is the fact that the traffic experiences more delay from *RX* to the destination when there is more traffic taking that path. Delegation to *PRX* reduces the load on the path from *RX* to the destination router, and hence reduces the queuing delay on that path. This is an interesting finding and needs further exploration. In particular, in terms of the average end-to-end delay, the delay from *RX* to *PRX*, t_{CR}, may be absorbed by the reduction in the delay from *RX* to the destination. Depending on the actual value of



(a) FIB cache size 1% (b) FIB cache size 10%

Fig. 6: End-to-end delay in CAR

t_{CR} and the queuing delay on the path via *PRX*, the optimum FIB cache size can be tuned.

V. Summary and Discussion

Following the RaaS concept [17], we introduced a new architecture, CAR, to address the growing complexity of routers and outlined its principles and key components, while navigating through possible benefits and implied challenges. We also presented our initial prototype on how CAR can help reduce FIB tables routers. CAR advocates *on-demand seamless integration of the cloud's computational and storage capabilities* into the Internet backbone.

Beyond bridging the gap between router hardware and software-based routing services, CAR allows opportunities to improve 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.

Deploying CAR at inter-AS level will entail some changes to the existing policy routing practices. For a smooth transition, an AS could initiate a *consulting phase* by discussing about the best practice and informing CAR provider about its own requirements [15]. Based on the observation of traffic flow, the CAR provider will first optimize the routing and deploy intelligent traffic handling in client's network. Later, in *hand-over phase*, AS will allow CAR provider to access its own FIB and finalize the process. Further models could be developed integrating CAR providers into the inter-AS routing architecture. Various implementation challenges of CAR (e.g., potential loops and reliability via multiple clouds) need to be explored.

References

[1] TM Forum IPsphere Framework. http://www.tmforum.org/ipsphere.

[2] BGP routing growth in 2011. http://labs.apnic.net/blabs/?p=25, October 2011. APNIC.

[3] Cloud's Commodity Pricing Squeezes Service Providers, Creates Opportunities. www.cio.com/article/714864/Cloud_s_Commodity_Pricing_Squeezes_Service_Providers_Creates_Opportunities, 2012. CIO.

[4] The CAIDA UCSD Anonymized Internet Traces 2014. www.caida.org/data/passive/passive_2014_dataset.xml.

[5] 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*, 40:183–194, August 2010.

[6] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proc. of USENIX NSDI*, pages 273–286, 2005.

[7] P. K. Dey and M. Yuksel. On the breakeven point between Cloud-Assisted and legacy routing. In *2016 IEEE 5th International Conference on Cloud Networking (CloudNet)*, Pisa, Italy, Oct. 2016.

[8] 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 ACM SOPS*, pages 15–28, 2009.

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

[10] N. Feamster, J. Rexford, and E. Zegura. The road to SDN: An intellectual history of programmable networks. *ACM Queue*, Dec. 2013.

[11] B. Ford. Icebergs in the clouds: The other risks of cloud computing. In *Proc. of USENIX HotCloud*, pages 453–466, Boston, MA, June 2012.

[12] H. T. Karaoglu and M. Yuksel. Offloading routing complexity to the cloud(s). In *Proc. of IEEE ICC Workshop*, June 2013.

[13] E. Keller, S. Ghorbani, M. Caesar, and J. Rexford. Live migration of an entire network (and its hosts). In *Proceedings of HotNets*, 2012.

[14] C. Kim, M. Caesar, A. Gerber, and J. Rexford. Revisiting route caching: The world should be flat. In *Proceedings of the 10th International Conference on Passive and Active Network Measurement*, pages 3–12, Seoul, Korea, 2009.

[15] V. Kotronis, X. Dimitropoulos, and B. Ager. Outsourcing the routing control logic: Better internet routing based on sdn principles. In *Proc. of Hotnets*, pages 55–60. ACM, 2012.

[16] A. Kuzmanovic. Skynet: A cloud-hopping data transfer architecture. In *Proceedings of IEEE Annual Computer Communications Workshop (CCW)*, Cape Cod, MA, October 2011.

[17] K. K. Lakshminarayanan, I. Stoica, S. Shenker, and J. Rexford. Routing as a service. Technical Report UCB/EECS-2006-19, Feb 2006.

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

[19] T. A. Limoncelli. OpenFlow: A radical new idea in networking. *ACM Queue*, June 2012.

[20] R. Mahajan, D. Wetherall, and T. Anderson. Negotiation-based routing between neighboring ISPs. In *Proc. of 2nd NSDI*, pages 29–42. USENIX Association, 2005.

[21] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, April 2008.

[22] M. Menth, M. Duelli, R. Martin, and J. Milbrandt. Resilience analysis of packet-witched communication networks. *IEEE/ACM Trans. Netw.*, 17:1950–1963, December 2009.

[23] G. Rétvári, J. Tapolcai, A. Kőrösi, A. Majdán, and Z. Heszberger. Compressing IP forwarding tables: Towards entropy bounds and beyond. In *Proc. of ACM SIGCOMM*, pages 111–122, 2013.

[24] D. Rosenthal. Cloud storage pricing history. blog.dshr.org/2012/02/cloud-storage-pricing-history.html, 2012.

[25] 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 *Proceedings of ACM SIGCOMM*, Helsinki, Finland, August 2012.

[26] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford. Virtual routers on the move: Live router migration as a network-management primitive. In *Proc. of ACM SIGCOMM, 231–242*, 2008.

[27] Y. Wang, H. Wang, A. Mahimkar, R. Alimi, Y. Zhang, L. Qiu, and Y. R. Yang. R3: Resilient routing reconfiguration. *SIGCOMM Comput. Commun. Rev.*, 40:291–302, August 2010.

[28] T. Wood, A. Gerber, K. K. Ramakrishnan, P. Shenoy, and J. Van der Merwe. The case for enterprise-ready virtual private clouds. In *Proc. of conference on Hot topics in cloud computing*, pages 4–4, 2009.

[29] H. Yuan, T. Song, and P. Crowley. Scalable NDN forwarding: Concepts, issues and principles. In *Proceedings of IEEE ICCCN*, 2012.

[30] M. Zec, L. Rizzo, and M. Mikuc. DXR: Towards a billion routing lookups per second in software. *ACM SIGCOMM CCR*, 42(5):29–36, 2012.