On the Feasibility of Cloud-Based SDN Controllers for Residential Networks

Curtis R. Taylor, Tian Guo, Craig A. Shue, and Mohamed E. Najd
Worcester Polytechnic Institute
{crtaylor, tian, cshue, menajd}@cs.wpi.edu

***Abstract*—Residential networks are home to increasingly diverse devices, including embedded devices that are part of the Internet of Things phenomenon, leading to new management and security challenges. However, current residential solutions that rely on customer premises equipment (CPE), which often remains deployed in homes for years without updates or maintenance, are not evolving to keep up with these emerging demands. Recently, researchers have proposed to outsource the tasks of managing and securing residential networks to cloud-based security services by leveraging software-defined networking (SDN). However, the use of cloud-based infrastructure may have performance implications.**

In this paper, we measure the performance impact and perception of a residential SDN using a cloud-based controller through two measurement studies. First, we recruit 270 residential users located across the United States to measure residential latency to cloud providers. Our measurements suggest the cloud controller architecture provides 90% of end-users with acceptable performance with judiciously selected public cloud locations. When evaluating web page loading times of popular domains, which are particularly latency-sensitive, we found an increase of a few seconds at the median. However, optimizations could reduce this overhead for top websites in practice.

I. Introduction

The task of managing and securing residential networks is inherently challenging. These networks are operated by end-users who often lack technical backgrounds and knowledge about how to administer their networks. In addition, the extent of these administration tasks are fundamentally limited by residential consumer-grade routers, that are often deployed for years, with little in the way of maintenance or upgrades [1].

Residential networks are increasingly home to diverse Internet-capable devices, such as gaming consoles, media centers, and Internet of Things (IoT) devices that have sensor and actuation capabilities. While these devices provide useful capabilities for residential users, they also highlight longstanding limitations in residential networks. In particular, these devices exhibit significant vulnerabilities and patching difficulties [2] and have increasingly been harnessed for use in massive Internet attacks [3]. These residential networks typically offer little resistance to attack due to their lack of security resources and limited administration.

Some researchers have proposed to address these limitations in residential networks by installing a new type of hardware device, called virtual customer premises equipment (vCPE) [4], in place of existing residential routers. These vCPE devices are designed to work with support from the user's Internet Service Provider (ISP), which must create and manage the appropriate middleboxes to offer management services and security protections. Unfortunately, the deployment of vCPE solutions are limited and residential users rarely have multiple broadband ISP options in the United States [5], leaving most residential users without access to these services.

Other approaches, including our own prior work [6], [7], have examined the feasibility of a residential software-defined network (SDN) approach. In those approaches, a cloud-hosted virtual machine acts as an OpenFlow [8] controller for commodity home routers. The controller can vet new flows and drop, modify, or approve packet forwarding out specific interfaces or tunnels. The controller can further tunnel traffic through the appropriate network function virtualization (NFV) system (generically called a *middlebox*) for analysis.

SDN and NFV applications can vary widely, yet the details of how these techniques are used can have profound impacts on the viability of the approaches. We examine different SDN and NFV application types, determine their inherent characteristics for new flows and analyze the impact such applications would have on users given different network characteristics. Specifically, our key contributions are as follows:

- We investigate the impact of controller latency on user-perceived performance. We focus on latency-sensitive web browsing traffic that dominates residential traffic. When considering top websites, we see a 2 second increase in webpage loading times for the 50th percentile of sites when using a controller at a round-trip time of 50 ms. Our results provide an upper bound on the impact on webpage loading time and could be reduced with proactive controller policies.
- We evaluate the latency impacts of cloud-hosted OpenFlow controllers on real-world residential connections. We find that roughly 90% of residential users have at least two public cloud location for a potential OpenFlow controller within a 50 millisecond round trip time (RTT).

II. Background

Software-defined networking (SDN) is a networking paradigm that allows logically centralized control of network communication by enabling switches to request forwarding information from a controller. Traditionally, controllers are placed within the same local area network (LAN) as the switch to reduce latency. Our work seeks to understand the feasibility and impact of deploying the SDN controller in the cloud. We

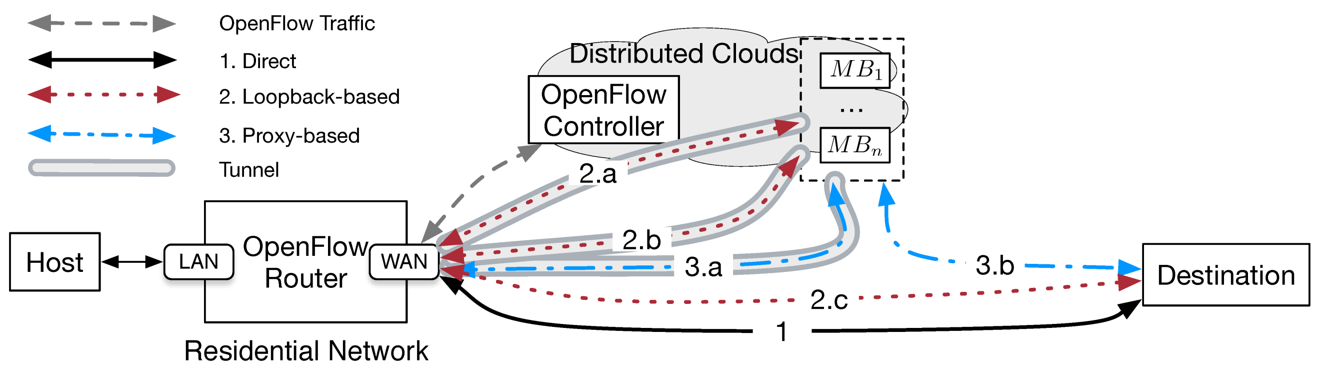


Fig. 1. **System diagram for residential SDN and middlebox solutions.** These approaches use a SDN-capable router within the home, a cloud-based OpenFlow controller, and a number of cloud-hosted middleboxes for NFV support. Paths labeled as 2.a, 2.b and 3.a use pre-established tunnels.

target the most popular SDN protocol, OpenFlow [8], in our analysis.

Before we can analyze the performance impacts of cloud-based OpenFlow controllers and middleboxes on residential networks, we must first identify how these systems are effectively used in practice. We have identified four broad classes of modules that may run on the controller. These modules characterize how SDNs and middleboxes can be used to manage and protect residential networks. Classes 1 and 2 are distinct, while classes 3 and 4 are extensions to class 2 modules. They are as follows:

- **Class 1: Proactive Flow Control Modules:** Using proactive rules, controllers can push flows to switches that will allow forwarding decisions to be made locally without consulting a controller or middlebox.
- **Class 2: Controller-Centric Modules:** For these modules, the controller only needs to see the initial packet in one or both directions of a new network connection. Such modules include stateless firewalls, DNS blacklists, and connection loggers. These modules can be implemented at an OpenFlow controller without requiring a separate middlebox that would analyze subsequent packets in the flow. The network traffic for these modules are depicted using the dashed OpenFlow traffic and the solid direct traffic lines in Figure 1.
- **Class 3: Partial Connection Middlebox Modules:** These modules must consume a relatively small portion of a connection's actual payload to function correctly, but do not need to be involved in the full connection. Such modules include deep-packet traffic classification tools or security tools that validate the initial handshaking process of a connection, such as our own TLSDeputy [7] module. These modules can use a "loopback" approach, as shown in lines 2.a and 2.b in Figure 1, in which the OpenFlow switch essentially *temporarily* redirects communication for the connection through a tunnel to a middlebox before receiving it again and delivering it to the destination, as shown in line 2.c. The tunneling allows the middlebox to inspect the payload. Once the middlebox has finished analyzing the connection, it then informs the SDN controller to remove the indirect looping process. Accordingly, the remainder of the connection proceeds directly between the end-hosts without middlebox involvement, as shown by line 2.c in Figure 1.
- **Class 4: Full Connection Middlebox Modules:** This class of modules require that all packets in the flow, in both directions, be inspected by the middlebox module for the life of the connection. Example modules include intrusion detection systems (IDSes) and anonymizing proxies, since any packets that bypass the middlebox would undermine the module's mission. The traffic pattern for this approach is shown with lines 3.a and 3.b in Figure 1.

A. *Performance Implications for Module Classes*

With this basic classification of modules, we can begin to discuss the key performance characteristics of each. Class 1 modules are independent of latency and bandwidth since they are proactively pushed and not actively involved with new flows. Class 2 and Class 3 modules are primarily concerned with latency: they must be involved with each new connection, but initial handshakes are typically low bandwidth. Class 4 modules affect the entire communication and are thus concerned with all the traditional network performance properties, including latency, bandwidth, jitter, and packet loss.

Network latency is also a key consideration for web communication since web page retrievals often involve many short network connections to load HTML files, images, scripts, stylesheets, and other elements from many servers. These retrievals can also have dependency chains; for example, a web browser will not start a connection to load an image until it retrieves an HTML document that indicates the address of the image to load. As a result, the latency with the controller must be incurred with each of these dependent connections.

While web traffic is the biggest traffic type in residential networks [9], controller applications may optimize for popular web destinations. When considering a web request for example.com, the controller may have a pre-existing record of content providers associated with example.com and push appropriate rules to the residential router to avoid the need for future controller elevations for those requests. However, it is unlikely that a controller would have such records for less popular destinations and cumulative latency would be a concern.

Controller applications must also be careful about installing flow rules to minimize performance implications while achieving certain security goals. For example, a controller may push a rule indicating that all DNS queries can be sent directly without controller involvement, but that all responses must be vetted by the controller. Since DNS responses contain the entire query, a DNS blacklisting module, for example, could perform its work by suppressing or altering the response without needing to consult the original query. This strategic minimization of controller elevations is particularly important as the latency between the router and controller grows. We discuss controller policy decisions in more detail in Section IV.

B. *Measurement Objectives*

Class 1 modules essentially have the same performance overheads as traditional networks since only proactive rules are used. However, Classes 2, 3, and 4 each have latency concerns and thus are concerned with the same research question: What latency impacts will an OpenFlow controller running in the public cloud incur? We discuss this problem in detail in Section III.

Class 4 modules are concerned with characteristics such as throughput, packet loss, and jitter and essentially represents the middlebox placement problem. Fortunately, the middlebox placement problem allows more flexibility: as long as the middlebox has a reliable, high-throughput path to the residence, the placement is geographically independent. Further, cloud providers are routinely used to host sites and resources that consume high throughput and demand reliable connections, such as video streaming providers or VPN endpoints. Since existing systems already meet these requirements, we focus our efforts on latency concerns.

III. The Cloud Controller Placement Problem

The placement of an OpenFlow controller with respect to its controlled switches has previously been recognized as an important problem [10]. Latency is the primary consideration when placing the controller. In enterprise or data center networks, the controller can be placed in the same LAN. Unfortunately, such in-network placement is often infeasible for residential network settings. We are thus interested in understanding the feasibility of deploying a cloud-based controller for residential users.

A. *Measurement Methodology*

We now detail our methodology for measuring and understanding the feasibility of outsourcing an OpenFlow controller to a cloud server given the current *residential network* connectivity present in the continental United States (US). We focus on the continental US given its broad geographic region,

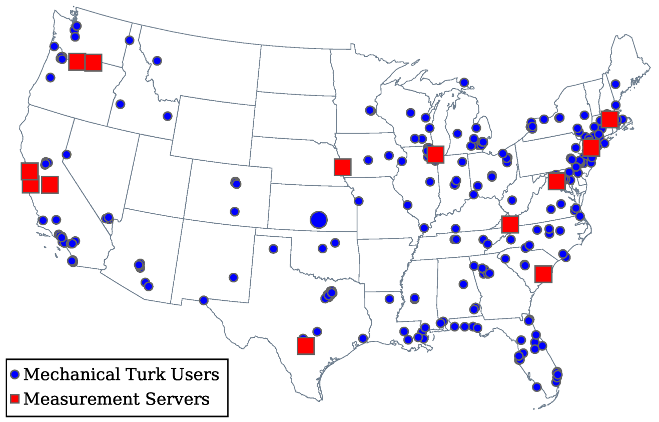


Fig. 2. Geographic map of Mechanical Turk users and our cloud measurement servers.

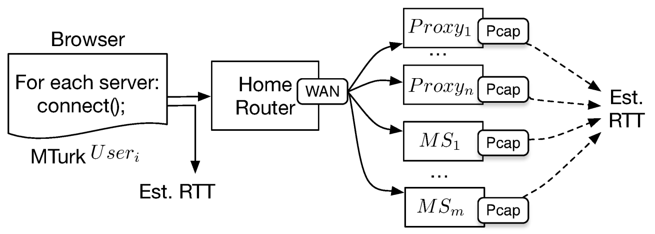
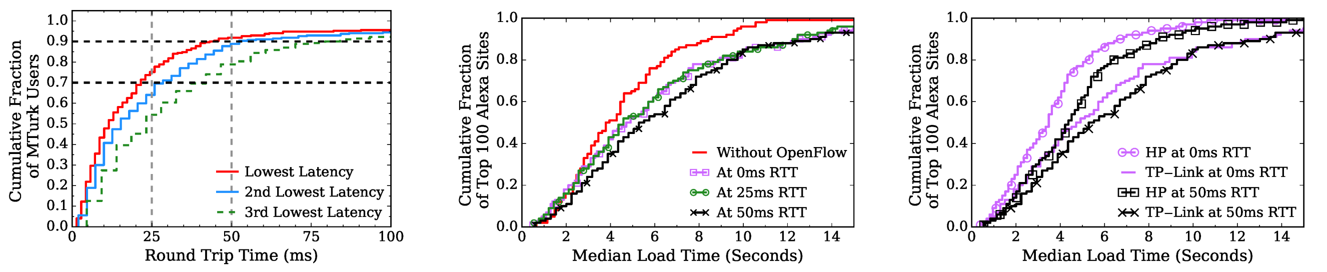


Fig. 3. This diagram describes the process an Mechanical Turk client performs to collect RTT measurements.

diverse last-mile network connectivity, and its mixture of urban and rural residences. We leverage four popular public cloud platforms: Amazon EC2, Google Cloud Platform, Microsoft Azure, and Digital Ocean. Using these services, we host a total of 12 measurement servers inside virtual machines (VMs). In addition, we also include a server running in a VM at our university. These servers are geographically spread across US, as shown in Figure 2. We then recruit residential users to perform connections to each measurement server using Javascript from their own computer as shown in Figure 3. We collect network-level data using packet captures and use the packet captures for measuring latency. This collection occurred during a two-week period in August 2016.

Using Amazon's Mechanical Turk service [11] we recruited participants and provided them with modest compensation to visit a speed testing website [1] that we hosted at our institution. Through that service, we initially recruited a total of 497 unique participants. However, we had to exclude users that did not meet our eligibility criteria, namely that the user is located in the United States and is using a residential network connection (which excludes VPNs, cellular connections, and corporate networks). We used a combination of reverse DNS, IP geolocation databases, and an examination of the IP address's associated network provider, we filtered our participants to a total of 270 eligible participants.

[1]Available at http://speedtest.wpi.edu/.



(a) RTT distribution between users and clouds (b) Page Loading Time with SDN (c) Consumer TP-Link vs. Enterprise HP Switch

Fig. 4. RTT measurements and the resulting page loading time (PLT) analysis based on those RTT measurements. We compare the PLTs by fetching top Alexa 100 websites at different controller latencies on our consumer-grade TP-Link switch from a residential network and observe how PLT is affected. Finally, we compare our consumer-grade switch to an enterprise-grade HP switch in that same residential network to determine the hardware's impact.

During the speed test, the residential user's browser first downloads a JavaScript file that contains URLs that can be used to access our distributed cloud VMs. The browser then runs the script to establish HTTP connections to all our VM servers. We calculate the round trip time (RTT) between the residential user and all cloud servers using packet captures collected at the VM servers.

B. Round Trip Times to Cloud Controllers

Figure 4(a) shows the network latency performance for US residential users with the current cloud infrastructure. We find that more than 70% of Mechanical Turk users are within a 25 ms RTT and roughly 90% are within a 50 ms RTT to two or more of our cloud servers. Our latency measurement results indicate the promise of hosting OpenFlow controllers such that a large fraction of the US-based users can be within a small RTT of at least one cloud-based controller.

C. Quantifying Performance Impact with Page Load Time

To put the controller RTT into perspective, we examine web-related traffic which is often composed of many short network flows, which are the worst-case scenario for controller latency since the latency cannot be amortized over the length of a longer connection. Further, web traffic is an important category that dominates residential traffic [9]. We quantify the latency's impact on web browsing by measuring the page load time (PLT) of popular domains from a residential network.

When performing this study, we use an unoptimized Class 2 controller module as an example. In other words, each connection to fetch remote resources, from DNS, website servers, CDN servers or advertisement networks, needs to be approved by the Floodlight SDN controller [12] twice: once for the first packet in each direction of a flow (e.g., both the TCP `SYN` packet and the matching `SYN+ACK` packet). Thus every PLT measurement requires every new flow to be independently approved by the Floodlight OpenFlow controller and results in an additional entry in the flow table.

The PLT is defined as the time interval between the start of the first connection and when the browser signals the onload event, which we capture using events triggered in the Chrome browser as discussed by Bell *et al.* [13]. Intuitively, assuming the residential user's request to a domain is fulfilled by the same set of end resources, PLT can be impacted by (1) the network latency between residential SDN router and the OpenFlow controller, (2) the maximum length of dependent network connections, and (3) the residential router's inherent ability to process OpenFlow traffic.

From a residential network, we measure PLT for the top 100 Alexa US domains. To measure the PLT of a particular domain, we modified a popular open source Chrome extension [14] to record the time it takes for the page load event to occur [13], [15]. We repeat the PLT measurement 25 times for each site, each time with a clean browser cache. We then report the median PLT for that site.

To study the impact of RTTs on our residential SDN architecture, we used an in-LAN controller and artificially added latency to the controller's reply, allowing us to explore the impact of a network RTT ranging from roughly 0 ms to 50 ms. Based on our results (shown in Figure 4(a)), 90% of users can reach a cloud-based controller within a 50 ms RTT. Therefore, we believe the performance degradation observed with a 50 ms RTT is a reasonable upper bound for our experiments. In Figure 4(b), we plot the median PLT for all top 100 Alexa domains. Our results show that redirecting all new connections to an OpenFlow controller increases the PLT from approximately 4 seconds without OpenFlow to 6 seconds with OpenFlow at the 50th percentile of Alexa sites. In all, we conclude that the performance degradation in the form of median PLTs is mostly attributed to the existence of the controller and the flow elevation process.

Importantly, this exploration focuses on an unoptimized controller module. With optimizations, such as only examining a single direction in the flow or by proactively allowing traffic to known content providers, the PLT could be dramatically reduced. We discuss this in greater detail in Section IV.

D. Impact of Router Hardware on Page Load Time

We also investigated whether the hardware of consumer-grade routers would be a factor for this approach. We compared an enterprise-grade HP 2920-24G OpenFlow switch with a consumer-grade TP-Link Archer C7 router running Open vSwitch. As one might expect, the enterprise switch nearly always outperforms the consumer router in the 0 ms and 50 ms latency environments, as shown in Figure 4(c). The HP switch has multiple advantages, including more memory and hardware flow tables. However, our measurement process also gives the HP switch a built-in advantage: the TP-Link router supports TLS and was enabled in our experiments, since a practical deployment requires TLS for the OpenFlow connection. Since the HP switch does not support TLS connections, it was not responsible for the inherent encryption and decryption operations for each message. The overheads associated with TLS are unclear without further experimentation.

E. Controller Latency Summary

Our measurement results indicate there is a 2 second increase in median PLTs in the worse case scenario when using a cloud-hosted controller at a 50 ms RTT. As the cloud becomes more distributed, we expect the median PLTs to drop proportionally to the minimal cloud latency. Further, as consumer router hardware improves, the overall latency may also improve.

IV. DISCUSSION

When considering the deployment of cloud-based OpenFlow controllers, there are a few different possibilities for improving performance in residential networks. First, there is the possibility to deploy more cloud data centers in order to reduce latency to homes but potentially at higher computing and infrastructure costs. Second, we could consider improving the hardware deployed in the home as we have shown a commodity router's ability to process OpenFlow packets is limited. Finally, a more immediately and practical approach to improving residential SDN performance would be to fine-tune policy at the controller to limit switch requests while also meeting security goals. We now expand more on controller policy considerations.

OpenFlow controllers can be configured with different types of policy and rule creation strategies which directly affect the number of elevations required for a new connection. One type of policy, using coarse-grain rules, employs wildcards for network addresses or transport layer ports. With such wildcarding, it is possible for multiple network connections to use the same policy rule. This allows the OpenFlow switch, such as a residential router, to manage subsequent connection matching the policy rule without an additional elevation. In our analysis, we focused on fine-grained flow control policies, in which no wildcards are used for network addresses or ports. This means each new connection requires an elevation to the controller. As a result, our analysis essentially captures the most conservative estimate of the impact of a cloud-hosted OpenFlow controller. The performance of coarse-grain rules would essentially blend our analysis with that of direct connections, with cache hits incurring no additional latency and cache misses having latency similar to our observations.

OpenFlow controllers may choose how they install rules in the OpenFlow switch. Upon receiving an elevation for the first packet in a new flow, the OpenFlow controller may choose to proactively approve the flow bi-directionally or only uni-directionally (essentially, just approving the connection initiator to reach the responder). If the controller installs only a uni-directional rule, any response would also be elevated to the controller, incurring a second round of latency to the controller. In our experiments, our controller installed rules uni-directionally and thus required two elevations to the controller for each new connection (e.g., both the `SYN` packet and for the `SYN+ACK` packets in the TCP handshake). Accordingly, our results are again conservative. A controller installing bidirectional rules would essentially incur half the number of elevations and could be twice as far away while obtaining reasonable performance. Such controllers may be usable for over 90% of residential users since they are less latency sensitive.

V. RELATED WORK

Project BISmark [16] is the most well-known body of work in residential networks. Project BISmark extends Feamster's [17] position paper on outsourcing home network security to third-party experts who can better manage the home networks. BISmark has been used for extensive performance characteristics of residential network ISP's [18] and wireless performance [19]. However, BISmark has not examined connectivity to public cloud infrastructure or the implications of hosting a cloud-based controller or NFV middleboxes. Instead, the approach measures connections to M-Lab servers, which are often hosted at academic institutions and may not be representative of commercial clouds. BISmark's core system uses locally installed applications on the router rather than using OpenFlow. A BISmark extension [20] does use a remote OpenFlow controller, but not for actively managing individual flows. Instead, that project queries a set of static rules on the OpenFlow router to gather statistics to help users enforce network usage caps.

Additional research has focused on deploying SDN within the home. The Homework project [21] uses a PC acting as a router running an OpenFlow controller within the LAN to understand HCI aspects of LAN network management. Yiakoumis *et al.* [22] proposed using the FlowVisor [23] tool to allow an ISP to facilitate the management of certain home devices by outside service providers, such as utility companies. They measured the impact of an OpenFlow controller that was within 16ms of 7 homes. Their limited study does not provide enough data on viability for nation-wide deployments of SDN and NFV solutions. HomeVisor [24] considers the variation in packet delay when slicing home networks. Lee *et al.* [25] suggest using cloud-based SDNs for auto-configuration and identification of devices, but only considers the overhead of the configuration and identification protocol rather than all network traffic. Finally, our past work of a whole-home proxy solution [6] and a cloud TLS verification and revocation approach for the home [7] did not explore the performance of the approaches on a broad scale.

In enterprise networks, researchers have considered outsourcing network functionality through cloud or third-party services [26], [27]. However, enterprise networks differ greatly from home networks, including far different network connectivity [18], leaving unresolved questions about the feasibility of such an approach in residential networks. One large scale attempt to outsource in the enterprise is APLOMB [28]. APLOMB outsources network functionality to the cloud using a specialized network gateway. Further, this approach requires DNS modifications to support redirection to cloud MBes from clients. This strategy that is not applicable to most residential networks as they do not host services that require DNS.

Finally, Persico *et al.* [29] explored the intra-cloud throughput performance of virtual machines at the Amazon EC2 cloud provider. Their experiments showed high throughput for VMs, confirming the cloud has adequate networking resources. Our work focuses on communication between cloud VMs and residential networks, exploring potential bottlenecks between clouds and their home users.

VI. Conclusion

In this work, we characterize residential network connections to cloud infrastructure. Using Amazon's Mechanical Turk, we recruit 270 participants across the United States and use in-browser instrumentation to direct participants to connect to various cloud instances hosted by 4 major providers in different geographical location. We characterize the connections using packet captures on the servers we controlled. With this data, we examine the OpenFlow controller placement problem for residential SDNs and found that 90% of users were within 50 ms of a cloud instance. While this latency is most likely to affect the web browsing experience, due to its interdependent objects and connection characteristics, optimizations can reduce this effect. With these results, we conclude that residential SDN and middleboxes are feasible for roughly 90% of US users even when limited to publicly available cloud VMs. In order to improve residential SDN performance in the future, we can consider multiple avenues including: (1) more data center deployments to provide lower latency connections to homes, (2) higher performance hardware in residential routers, or (3) fine-tune OpenFlow controller policy to reduce flow requests to the controller while maintaining operational control of network communication.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1422180.

References

[1] J. Valentino-DeVries, "Rarely patched software bugs in home routers cripple security," http://www.wsj.com/articles/rarely-patched-software-bugs-in-home-routers-cripple-security-1453136285, 2016.

[2] "The internet of things is not always so comforting," http://blog.talosintel.com/2016/02/trane-iot.html.

[3] "Largest ddos attack ever delivered by botnet of hijacked iot devices," http://www.networkworld.com/article/3123672/security/largest-ddos-attack-ever-delivered-by-botnet-of-hijacked-iot-devices.html.

[4] "Residential cord," https://wiki.opencord.org/display/CORD/Residential+CORD.

[5] "Most of the US has no broadband competition at 25mbps, FCC chair says — ars technica," http://arstechnica.com/business/2014/09/most-of-the-us-has-no-broadband-competition-at-25mbps-fcc-chair-says/.

[7] C. R. Taylor and C. A. Shue, "Validating security protocols with cloud-based middleboxes," in *IEEE Conference on Communications and Network Security (CNS)*, 2016.

[6] C. R. Taylor, C. A. Shue, and M. E. Najd, "Whole home proxies: Bringing enterprise-grade security to residential networks," in *IEEE International Conference on Communications (ICC)*, May 2016.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, 2008.

[9] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, 2009.

[10] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN, 2012.

[11] G. Paolacci, J. Chandler, and P. G. Ipeirotis, "Running experiments on amazon mechanical turk," *Judgment and Decision making*, 2010.

[12] "Floodlight openflow controller," http://www.projectfloodlight.org/floodlight/.

[13] O. Bell, M. Allman, and B. Kuperman, "On browser-level event logging," Tech. Rep., 2012.

[14] "Google chrome extension to measure page load time and display it in the toolbar," https://github.com/alex-vv/chrome-load-timer.

[15] X. S. Wang, A. Krishnamurthy, and D. Wetherall, "Speeding up web page loads with shandian," in *13th USENIX Symposium on Networked Systems Design and Implementation*, 2016.

[16] S. Sundaresan, S. Burnett, N. Feamster, and W. De Donato, "Bismark: a testbed for deploying measurements and applications in broadband access networks," in *USENIX Annual Technical Conference*, 2014.

[17] N. Feamster, "Outsourcing home network security," in *Proceedings of the 2010 ACM SIGCOMM Workshop on Home Networks*, ser. HomeNets '10. New York, NY, USA: ACM, 2010, pp. 37–42. [Online]. Available: http://doi.acm.org/10.1145/1851307.1851317

[18] S. Sundaresan, W. De Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè, "Broadband internet performance: a view from the gateway," in *ACM SIGCOMM computer communication review*. ACM, 2011.

[19] S. Sundaresan, N. Feamster, and R. Teixeira, "Measuring the performance of user traffic in home wireless networks," in *International Conference on Passive and Active Network Measurement*, 2015.

[20] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, February 2013.

[21] R. Mortier, T. Rodden, P. Tolmie, T. Lodge, R. Spencer, A. crabtree, J. Sventek, and A. Koliousis, "Homework: Putting interaction into the infrastructure," in *ACM Symposium on User Interface Software and Technology*, 2012.

[22] Y. Yiakoumis, K.-K. Yap, S. Katti, G. Parulkar, and N. McKeown, "Slicing home networks," in *ACM SIGCOMM workshop on Home networks*, 2011.

[23] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," 2009.

[24] T. Fratczak, M. Broadbent, P. Georgopoulos, and N. Race, "Homevisor: adapting home network environments," in *European Workshop on SDN*, 2013.

[25] M. Lee, Y. Kim, and Y. Lee, "A home cloud-based home network auto-configuration using SDN," in *IEEE International Conference on-Networking, Sensing and Control (ICNSC)*, 2015.

[26] 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 *SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2012.

[27] G. Gibb, H. Zeng, and N. McKeown, "Outsourcing network functionality," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, 2012, pp. 73–78. [Online]. Available: http://doi.acm.org/10.1145/2342441.2342457

[28] 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," *ACM SIGCOMM Computer Communication Review*, 2012.

[29] V. Persico, P. Marchetta, A. Botta, and A. Pescapé, "Measuring network throughput in the cloud: the case of Amazon EC2," *Computer Networks*, vol. 93, pp. 408–422, 2015.