A Case for Personal Virtual Networks

David Choffnes
Northeastern University

Abstract

Our mobile devices regularly encounter and connect to multiple networks to maintain seamless connectivity. While this enables a variety of services we increasingly rely on, these ubiquitous network connections raise a number of important concerns. Our devices regularly send traffic over networks they do not fully trust and that are not under user control, which leads to security vulnerabilities, policies that impact performance and service availability, and privacy violations.

We propose a new networking abstraction called Personal Virtual Networks, or *PVNs*, to address these issues. The key idea is to allow a device connecting to a foreign network to establish a virtual network under the device's full control, thus providing the illusion of a personal home network wherever the device roams. Devices can establish trusted network configurations, define policies for network traffic, and even deploy limited code that interposes on their traffic using a software middlebox environment. By making in-network resources available to devices via a secure and flexible interface, *PVNs* can enable more secure, private, and performant network experiences for users while potentially generating a new revenue stream for access networks that support them.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture

Keywords

Virtual networks, middleboxes, SDN

1. INTRODUCTION

Our laptops, mobile devices, and IoT devices seamlessly connect to WiFi access points and cellular networks, and sometimes both [27]. More often than not, we are connected to networks that we do not manage, control, and/or trust. There are very good reasons to take control out of the hands of users, as ISPs must protect the network from harm, *e.g.,* by preventing any single device from unfairly consuming scarce network resources [12, 41].

However, there are many cases where the lack of control can cause substantial harm to users. For example, mass surveillance (even using TLS interception), malware distribution, and policies such as shaping and content manipulation [19, 21] impact network traffic in ways that threaten privacy and security, and violate user expectations.

In addition to exposing users to misbehavior, these networks make it frustratingly difficult to deploy new protocols and policies for network traffic. For example, proxies and transcoders can optimize performance [1, 11, 42], but ISPs do not allow end systems to select and deploy them inside the network. Recent solutions to mitigate privacy leaks [30] can be efficiently deployed in carrier networks but require privileged access to traffic and network infrastructure.

Traditional approaches to address these problems fall along three axes. First, when faced with an untrusted network, a common approach to protect traffic is to securely tunnel it to one that is trusted using a virtual private network (VPN) at the cost of performance overhead. Second, ISPs deploy middleboxes to implement many useful network functions, but users do not have the flexibility to set their own policies. Third, software deployed on end hosts can address certain security, privacy, and performance issues, but there are limited resources on mobile devices to run them.

We argue for an alternative approach that leverages a new opportunity presented by networks that are increasingly endowed with network virtualization support and spare computation resources: *provide each device with its own virtual network under its control*. Such personal virtual networks, or *PVNs*, can allow devices to create a virtual network, determine the policies that apply to traffic over each link in the virtual topology, the locations of software middleboxes that interpose on the traffic, and the code that executes on that traffic. For example, a device can deploy a virtual network with software middleboxes that validate TLS certificates, selectively optimize TCP connections, and identify privacy leaks (Fig. 1(a)). In addition, such *PVNs* can enable selective routing of network traffic, leveraging path diversity from multihomed networks (Fig. 1(c)). In a sense, *PVNs* provides users and devices the illusion that they are in the same, fully controlled and customized network environment regardless of which access network they connect to.

The idea of ubiquitous personal virtual networks opens up numerous opportunities, as well as interesting challenges. For example, *PVNs* raise a number of questions that we dis-

HotNets-XV, November 09 - 10, 2016, Atlanta, GA, USA

DOI: http://dx.doi.org/10.1145/3005745.3005753.

cuss in this position paper, such as how to design and architect this service, how to trust and audit *PVN* providers, what to do when an access network does not support it, and how to incentivize deployment.

The rest of the paper is organized as follows. In the next section, we provide detailed motivation for our work and list key challenges for a *PVN* deployment. In Section 3, we present our proposed architecture and deployment models. Section 4 presents several examples of new and improved services that *PVNs* enable. We present related work in Section 5 and conclude in Section 6.

2. MOTIVATION AND CHALLENGES

The *PVN* abstraction addresses a growing problem in today's diverse mobile Internet systems: a lack of transparency or control over network traffic, particularly from mobile devices. Mobile OSes limit users to installing apps only from a curated stores subject to unilateral policies for inclusion [3, 4], limiting the ability to install software to improve network security, performance, and privacy. Mobile carriers can and do manipulate network traffic [42] (*e.g.,* via blocking and shaping), often without user awareness or consent.

Sometimes, these policies are put in place for the greater good, *i.e.,* protecting the users who install apps [3, 4] and use access networks [12]. However, these policies have created significant collateral damage: previous studies identified security [22, 23], privacy [10, 40], policy [41], and performance [35] issues in mobile systems. In the paragraphs below, we describe the costs of limited transparency and control to motivate the need for a new solution.

2.1 Security

The security of our day-to-day interactions on the Internet rely on trust assumptions that are easily violated. For example, we often implicitly assume that the plaintext contents of our TLS-encrypted traffic cannot be accessed or manipulated by third parties. However, many apps and browsers do not properly check certificate validity, if at all [23]—opening users and their devices to covert attacks from third-parties that man-in-the-middle (MITM) TLS connection to expose sensitive information or manipulate content.

We also generally trust that an ISP's DNS servers will provide valid mappings from names to IP addresses; however, the lack of security in today's DNS hierarchy puts users at risk of forged mappings. Combined with weaknesses in the TLS PKI, users can easily be sent to malicious domains without their knowledge.

Finally, we trust that our mobile OSes cannot easily be subverted by malicious software accessed over the network. However, there are a number of known attacks that exploit mobile OSes in the wild, and many users are indefinitely vulnerable due to a lack of vendor-supplied OS updates or user unawareness of available patches [39]. In these cases, the only way to improve user security is to detect malware in network traffic and block them, but there is no guarantee that such functionality is present in a given ISP.

These issues point to the need for additional mechanisms to improve network security: deploying trusted software running on trusted hardware to reduce the attack surface for connected devices. Because this is difficult to provide on mobile devices themselves, we propose running these mitigation techniques in the network. This not only provides a device- and app-agnostic platform to improve security, but also offloads potentially expensive computations into an environment with richer energy and computation resources.

2.2 Performance

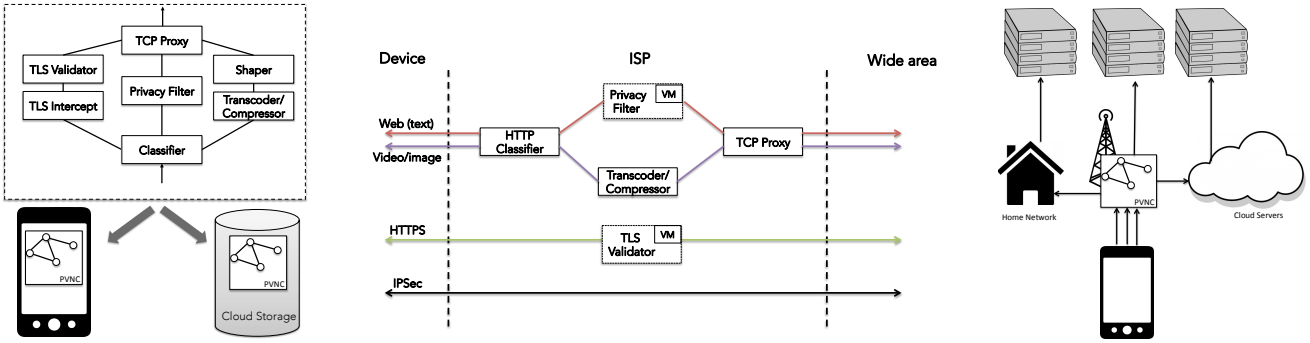
Our devices increasingly access the network via wireless (WiFi or cellular) connections, and the performance from wireless providers lags behind their fixed-line counterparts [41, 45]. This affects users in terms of poor performance (*e.g.,* latency and throughput), data consumption from monthly quota, and limited battery lifetime due to network activity. Proposals to address such problems fall along several axes: end-to-end solutions, in-network optimizaitons, and network management.

End-to-end solutions. Recent work proposes new protocols and optimizations to more efficiently use existing resources. For example, Chrome uses SPDY and QUIC protocols to improve mobile Web performance. Several browsers [25, 33] partially render pages in the cloud before delivering them to browsers. This improves performance for some pages, but a recent study calls into question the generality of this approach [34]. These solutions are also limited to traffic from browser apps, which generate a minority of total traffic (as little as 10% [43]).

In-network optimizations. An alternative is to implement performance-enhancing functionality in the mobile network via middleboxes, *e.g.,* via TCP-terminating HTTP proxies. Previous work shows that splitting TCP connections should offer better client-perceived performance (i.e., faster downloads) than direct connections if the proxy is on the same path [11, 17]. Splitting the connection reduces the RTTs between connected endpoints, which allows TCP to grow its congestion window faster, and it speeds loss detection and recovery. However, recent work shows that the impact of such proxies is mixed [44]: devices with better link quality benefited most from proxying, and the rest could receive worse performance due to proxying overheads.

Network management. To manage scarce bandwidth resources, operators historically throttled certain types of traffic (*e.g.,* P2P [9] or video [19]). While there is active discussion on whether to enforce network neutrality, recent FCC rules [12] bar *all* ISPs from commercially unreasonable practices against Internet traffic, including throttling.

Under these rules, several ISPs have explored alternative models to provide differential service. In one case, T-Mobile's Binge On program, which is enabled by default for its subscribers, zero-rates all participating video provider's



(a) PVNC example (b) Deployment example (c) Selective redirection

Figure 1: Personal virtual network components. (a) Users create personal virtual network configurations (PVNCs) and distribute them to devices and/or cloud storage. (b) In deployment, a *PVN* can use in-network devices (solid boxes) and software middleboxes (dashed boxes) to selectively interpose on traffic generated by a device. (c) A *PVN* can support selective redirection to cloud, home, or other execution environments depending on the needs of the configured services.

traffic, but also throttles it to 1.5 Mbps (often leading to sub-HD quality) [18]. One problem with this approach is that makes it difficult for users to set fine-grained policies regarding the service provided for any specific network flow. For example, users cannot decide to stream at high resolution (without zero rating) at the time the video is loaded; rather, there is one policy that applies to *all* of their video traffic. Such programs raise significant net neutrality concerns, mainly due to a lack of transparency and user choice.

The key limitations of the above approaches to network management and enhancing performance are that they are limited to specific apps, or they use a one-size-fits-all approach that does not always benefit users or their network traffic. An alternative approach that avoids net neutrality issues and can lead to improved performance is to allow users to configure the policies and performance optimizations that impact their traffic. These optimizations can potentially leverage information about network conditions provided by the access network, assuming there are guarantees that this data is not exposed to other parties.

2.3 Privacy

Most of our online activity is tracked by third parties, and our apps leak personally identifiable information (PII), *e.g.,* location, passwords, and phone numbers, over the Internet without our knowledge [30, 31, 38]. Further, users are increasingly surrounded by Internet of Things (IoT) devices equipped with sensors, microphones, and cameras, that can record and transmit our activity without consent.

A number of approaches attempt to address this issue by tracking information flows on mobile devices and analyzing software, but these are generally limited to specific app versions or OSes. Recent approaches that instead identify PII in network traffic [30] show promising results, but require either tunneling traffic to a remote network at the cost of extra delay or analyzing network traffic on a device, at the cost of battery life and network performance.

An alternative approach is to deploy in-network functionality that provides improved privacy without performance costs. Further, users could specify privacy policies that apply not only to their own network traffic, but also to traffic generated by sensors connected to the same network (*e.g.,* blurring their face in a video recording).

3. PERSONAL VIRTUAL NETWORKS

We now introduce our personal virtual network abstraction, and describe how it can address the issues raised in the previous section.

3.1 Overview

A personal virtual network is a virtual network that is configured by devices and deployed inside a physical network that they use to access the Internet. While we do not impose any requirements on the type of virtual network or the configurations it supports, we believe many of our goals can be achieved by leveraging previous work on open interfaces for specifying virtual networks, standard match/action rules to operate on each packet, and virtual machine environments for executing code. The key components are as follows.

Personal Virtual Network Configurations. First, a user must provide its device with its personal virtual network configuration (PVNC), *e.g.,* the one in Fig. 1(a). This can be created well before connecting to an access network, using high-level tools that compile user-readable configurations into low-level SDN code that is run in the network(s) where the *PVN* is deployed. The PVNC specifies a virtual network, the policies that apply to traffic each link in the virtual topology, the locations of software middleboxes that interpose on the traffic, and the code that executes on that traffic.

A user can specify the same PVNC for multiple devices, and the PVNC can be stored on the device or provided to an access network as a URI to a globally accessible PVNC object (*e.g.,* in cloud storage). In addition, PVNC components can be provided as independent entities and shared among users to facilitate PVNC creation. For example, we developers could create (and/or sell) malware-detection modules,

Web-optimizing modules, and tracker-blocking modules. To make *PVNs* accessible to a general audience instead of only networking experts, we propose building a "*PVN* Store" akin to an app- or browser-extension marketplace.

PVN Discovery and Deployment Protocol. When a user's *PVN*-enabled device connects to a network, it first conducts a network discovery operation to detect whether *PVNs* are supported. This could be done during DHCP negotiation, or afterward using protocols like UPnP. Such discovery operations can span multiple providers (using limited flooding, *e.g.,* via special anycast addresses) in case the access provider does not support *PVNs.*

The discovery message (DM) will specify a sequence number (incremented for each discovery attempt), the language and/or standards that the PVNC supports (*e.g.,* OpenFlow, Docker containers), the virtual network topology, and an estimate of the network and computational resources requested by the PVNC. A network that supports *PVNs* should respond to each DM with the location of the *PVN* deployment server, the languages/standards supported, an offered virtual network topology and resources (which may be identical to the request, or a subset), a cost per VNC module, and a time at which the offer expires.

If the requested and offered PVNC match and the cost is acceptable, the device sends an deployment request, which includes the PVNC and payment details (if necessary). If not, the device has several options. It can reject the offered topology, choosing instead to wait for a "better" offer from other *PVNs* in the discovery zone, or simply choose to eschew *PVNs* entirely. The device also can choose to send a new DM with a PVNC that includes a subset of the original configuration, to retrieve a new price. Last, the device can send a deployment request using a subset of the DM-specified configuration, *e.g.,* including only the services that are offered for free.

Upon receiving a deployment request, the *PVN*-supporting network must install the PVNC and route the device's traffic through it. Upon successfully setting up the PVNC, the network sends an acknowledgement to the device, which also triggers a DHCP refresh to obtain the new addresses. If the deployment fails for some reason, the provider replies with a NACK and failure reason.

Auditor. Given that access networks are generally untrusted, *PVNs* must include techniques to ensure that PVNCs are correctly and continuously deployed. To address part of the problem, we propose using trusted hardware/software stacks that provide client-verifiable attestations that the specified network configurations and software middleboxes were installed and executed as requested. In addition, the device will need to obtain proofs that packets sent to the *PVN* were actually routed correctly through the *PVN*. To account for adversarial actions that manifest in the physical network topology, we propose using active network measurements that reliably identify policy violations. These can include tests for service differentiation, content modification, privacy exposure, inflated/short-circuited paths, and others. Observed violations in either configurations or policies can be used as evidence in billing disputes, and to inform reputations for *PVN* providers.

3.2 Why Use a New In-Network Solution?

While the previous section makes the case for using device-configured in-network functionality to address a number of problems when connecting to access networks, the reader may question whether this is the *right* place to deploy such features. Below, we address such questions.

Why not on devices? Many of the solutions we describe can potentially be deployed as software running on end hosts. However, accessing network traffic requires special privileges on device OSes that make them either unavailable for unprivileged users or raise warnings that discourage their adoption. In addition, network functionality implemented on mobile devices can consume scarce resources such as battery life, CPU, memory, and wireless bandwidth, and lead to *worse* network performance than doing nothing at all. Last, some problems are infeasible to address on end hosts, like enforcing network policies that apply to inbound traffic *before* traversing a constrained wireless link.

Why not in the cloud or in home networks? Previous work addresses many of the described problems by redirecting traffic to a trusted server running on a cloud-based platform or a device in a user's home network, *e.g.,* via VPNs, split Web browsers, and privacy-enhancing middleboxes. *PVNs* can also be deployed in this manner, using tunneling to reach the *PVN* deployment. However, there are several disadvantages for doing so when compared to using in-network support for *PVNs.* First, there are tunneling overheads in terms of additional interdomain traffic and its associated latency; *e.g.,* 10s of ms for well connected networks [32], but potentially 100s of ms for poorly connected networks. Second, the tunneled traffic may be subject to policies (*e.g.,* shaping) that do not apply to untunneled traffic [19]. Last, port blocking and service unavailability can also impact the effectiveness of such solutions in practice.

Why not the content providers? Servers can improve performance by optimizing traffic based on each client, and they can participate in services that improve privacy and security. However, these approaches only partially address the problem, and require users to trust each provider to protect them.

Why not use existing middleboxes? Middleboxes that improve security and performance are pervasive in ISPs, so why should we provide yet another middlebox abstraction that potentially conflicts with or reproduces functionality already present? We argue that *PVNs* are complementary to such systems, and a *PVN* deployment will in fact leverage existing in-network functionality when a device configures it (Fig. 1(b)). For example, when a device specifies a TCP proxy, the network provider can route its traffic through a

physical TCP proxy. Further, *PVNs* will leverage existing techniques to prove that any given network configuration and is valid according to important invariants, thus avoiding problems from configuration conflicts. Last, *PVNs* allow users to specify custom functionality via software middleboxes that is not present in existing physical middleboxes in a given network.

3.3 Key Challenges

Building a system that supports personal virtual networks raises a number of key challenges.

Validating that configurations and code are correctly deployed and executed. We propose using trusted hardware/software stacks to provide signed attestations that the user's configurations are deployed and code is executed as specified in the virtual network. While this provides evidence that software and configurations are not tampered with, they do not provide any proof regarding any *other* network policies or software that impact user traffic. To address this, we propose using limited active measurements to audit ISPs and check for violations of *PVN* policies.

Avoiding harm from user configurations. If users deploy arbitrary policies and code, they can potentially harm other users through unfair use of network and computational resources, or unauthorized access to other network traffic. To address this problem, we propose executing software middleboxes in secure sandboxes using a restricted development language that minimize attack surfaces. Each virtual network and its associated middleboxes can access only the traffic generated by the user who configured it.

Incentivizing access network providers. We believe that *PVNs* open a new opportunity for access providers to monetize their networks. For example, access providers can give users free limited resources and configurations in return for ads, and allow users to purchase additional resources and functionality. *PVNs* are also a way for providers to attract new subscribers, *e.g.,* by providing support for improved security, performance, and privacy compared with competitors. While such revenue models already exist in many environments (*e.g.,* "free" airport WiFi is often provided in exchange for watching ads, and users are offered premium access for a charge), a *PVN* architecture can provide users with much greater flexibility in the configurations (and thus cost) of their network connection.

Scalability and overhead. The *PVN* abstraction will be effective only if it can scale to serve potentially large numbers of subscribers with overhead that is negligible relative to non-*PVN* connections. We argue that this is feasible, *e.g.,* recent work [24] has shown that containers can be instantiated in 30 milliseconds, add only 45 microseconds of delay, and consume only 6 MB of memory.

Security and trust. Our approach uses a "trust but verify" model that assumes *PVN*-supporting networks will act honestly most of the time. Given a string of recent revelations about untrustworthy behavior from large service providers [7, 16], this assumption clearly will not always hold. We do not claim to solve the problem of ensuring that ISPs are trustworthy, nor do we argue that an arbitrary ISP is more trustworthy than a device or a VPN provider. Rather, we avoid the problem by allowing a user's device to tunnel to a nearby *PVN*-supporting network that is trusted.

To detect dishonest ISPs, we require that devices are able to audit their own *PVN* deployments. Doing so in a general way that cannot be cheated is an open problem, but previous work provides evidence that there are ways to successfully audit specific policies in practice [19, 44]. Should *PVNs* be successful, ISPs would be incentivized to act honestly or face loss of revenue from blacklisting, leading users to take their business to competing *PVN*-supporting providers.

Automating the process of negotiating access policies. We expect that many network providers may support partial *PVN* configuration of network services. In such cases, we need a way to negotiate a compromise between what the network provider allows and what the user requests. We believe a set of soft and hard constraints can inform the decision of whether a user is willing to connect to a given access network, and under what conditions.

Coping with unavailability. In the extreme case of the above issue, an access provider may not provide any *PVN* functionality. For users that insist on having the guarantees that *PVNs* provide, we propose tunneling traffic to nearby networks where *PVNs* are supported. This could be a next-hop AS, or it could be a server in the cloud. To efficiently identify and select good *PVN* deployment locations outside of the access network, we propose using active measurements to inform the costs of alternative locations.

4. PVN-ENABLED FUNCTIONALITY

In this section, we discuss several examples of *PVN*-enabled functionality. We argue that these examples suggest substantial potential benefits to embracing the *PVN* approach today, and that the abstraction is flexible enough to support emerging needs in future networked systems.

HTTPS/TLS Enhancements. Recent studies demonstrate widespread mismanagement in the HTTPS ecosystem, from server operators who do not reissue and/or revoke vulnerable certificates, to browsers and apps that do not properly check certificate validity. Addressing these problems in their entirety may require wholesale changes to the PKI that supports them; however, using *PVNs* we can deploy interim solutions today that recover substantial security. For example, a *PVN* middlebox can perform certificate validity checks beyond those provided by mobile OSes and apps, and reject connections for (or at least present warnings for) those using invalid certificates. This protects against malicious servers spoofing as their authentic ones, and can detect and prevent unauthorized TLS interception for eavesdropping on

encrypted connections to authentic servers.

DNS Validation. Even if the ISP does not support DNSSEC, a *PVN* DNSSEC module can provide secure DNS resolution on behalf of the user. Further, when accessing name entries that are not secured, the *PVN* can use a collection of open resolvers to ensure that clients are not maliciously sent to invalid addresses for a name.

Offloading computation and communication. Previous work explored opportunities to improve performance, battery life, and data consumption by offloading computation and communication to the cloud [8, 15]. Using *PVNs*, we can provide similar functionality without the performance cost of tunneling to a cloud-based deployments. For example, the Opera Mini and Amazon Silk cloud-based Web renderers [25, 33] could be deployed as modules in a *PVN*.

To further improve performance, *PVNs* can support explicit cooperation between apps, servers and middleboxes that take advantage of the fact that the middlebox is closer to the client than the servers, and the network bandwidth to the middlebox is relatively cheap compared to mobile devices. For example, many apps pre-fetch content to reduce user-perceived delays when navigating apps, but this can be costly in terms of data quota and battery life if the pre-fetched content is not used [29]. Using *PVNs*, we can explore a middle ground, where we run code on the middlebox that prefetches content to move it closer to users, without consuming device resources. Beyond this example, *PVNs* can support rendering of Web pages and caching of static portions of pages, bitrate transcoding for streaming video/audio, and custom transformations on social-network feeds.

Detecting and Blocking PII. Recent work demonstrated that substantial user PII is exposed over the Internet via network connections from devices [30]. Some of this information is necessary for applications to function correctly (*e.g.,* username and password to log into a service) and is protected from eavesdroppers (*e.g.,* using HTTPS); however, there is a substantial amount of PII that exposed over unencrypted connections (and thus is vulnerable to eavesdroppers) and/or to third parties (*e.g.,* advertisers and analytics companies). Using *PVNs*, we can deploy network-analysis tools that automatically detect when PII is leaked and provide users the option to block or modify them. Further, given a trusted and secure execution environment (*e.g.,* SGX [2]), such system can even perform a limited type of TLS interception to identify PII leaking in encrypted connections.

Certain sensitive operations (*e.g.,* the TLS interception described in the previous paragraph) cannot be performed in a *PVN* due to lack of trust in the underlying execution environment. In these cases, *PVNs* can provide flexible tunneling options, *e.g.,* to selectively tunnel traffic needing TLS interception to trusted cloud-based VMs, *without* tunneling *all* of a device's traffic (Fig. 1(c)).

Other applications. There are a large number of other applications we envision beyond the examples above, but cannot cover into detail due to space limitations. These include malware detection, seamless encryption everywhere, personalized performance-enhancing proxies, client-assisted replica selection, and privacy controls for traffic that is generated by sensors recording information about users connected to the same network.

5. RELATED WORK

This work is inspired by three threads of networking research: software defined networking and network function virtualization, software middleboxes and secure proxies, and active networks.

SDNs have enabled a new class of applications and services [13], but typically are used in environments where the network administration configures and deploys them. With *PVNs*, we propose making this functionality available in arbitrary networks, configured by users, and deployed by their devices.

There is along history of using VPNs and middleboxes to improve security, performance [1], and privacy [36]. Recent work explores the potential to combine VPNs and software middleboxes to address these issues [28]. *PVN* builds on these ideas to support device-configured software middleboxes without needing to tunnel to another network. VPMNs [6] provide the ability to create virtual mobile networks. Other work [14, 20] proposed decentralized techniques for naming, securing, and routing connections between a set of personal devices that form an overlay personal network. In contrast and orthogonal to this work, *PVNs* enable users to specify how an ISP should route and/or interpose on their traffic via in-network functionality.

Active networks [37] proposed network traffic that carries computation. *PVN* can be seen as a one-hop active network with additional flexibility and control.

A variety of previous work has identified important privacy [5, 30], security [22, 26], and performance [11, 34, 41, 44] issues that arise for end hosts in various networking environments. Our work uses these studies as motivation and provides in-network solutions to these problems.

6. CONCLUSION

We proposed a new networking abstraction, personal virtual networks (*PVNs*), that allows a device connecting to an access network to establish a fully configurable virtual network under its control. We identified several security, performance, and privacy problems in today's networks that motivate our design, and discussed how recent trends in software defined networking and in-network computation will facilitate deployment. We provided a set of example applications built atop *PVNs* that address many important problems when devices roam across different access networks today, and several problems that we believe will become prominent in the near future. As part of our ongoing work, we are building prototype *PVN* networks to better understand implementation trade-offs and guide future deployments.

7. REFERENCES

[1] Data Compression Proxy. developer.chrome.com/multidevice/data-compression.

[2] Intel® Software Guard Extensions (Intel® SGX). software.intel.com/en-us/sgx.

[3] App review. developer.apple.com/app-store/review/, June 2014.

[4] Policy guidelines and practices. support.google.com/googleplay/android-developer/answer/113474, June 2014.

[5] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel. FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps. In *Proc. of PLDI*, 2014.

[6] A. Baliga, X. Chen, B. Coskun, G. de los Reyes, S. Lee, S. Mathur, and J. E. Van der Merwe. VPMN: Virtual private mobile network towards mobility-as-a-service. In *Proc. of MCS*, 2011.

[7] T. Chung, D. Choffnes, and A. Mislove. Tunneling for Transparency: A Large-Scale Analysis of End-to-End Violations in the Internet. In *Proc. of IMC*, 2016.

[8] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In *Proc. of MobiSys*, 2010.

[9] M. Dischinger, M. Marcon, S. Guha, K. P. Gummadi, R. Mahajan, and S. Saroiu. Glasnost: Enabling end users to detect traffic differentiation. In *Proc. of USENIX NSDI*, 2010.

[10] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *Proc. of USENIX OSDI*, 2010.

[11] V. Farkas, B. Héder, and S. Nováczki. A Split Connection TCP Proxy in LTE Networks. In *Inf. Comm. Tech.*, 2012.

[12] FCC. Protecting and promoting the open internet. www.fcc.gov/general/open-internet, April 2015.

[13] N. Feamster, J. Rexford, and E. Zegura. The road to SDN: An intellectual history of programmable networks. *SIGCOMM Comput. Commun. Rev.*, 44(2), Apr. 2014.

[14] B. Ford, J. Strauss, C. Lesniewski-Laas, S. Rhea, F. Kaashoek, and R. Morris. Persistent personal names for globally connected mobile devices. In *Proc. of USENIX OSDI*, 2006.

[15] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen. COMET: Code offload by migrating execution transparently. In *USENIX OSDI*, 2012.

[16] J. Hoffman-Andrews. Verizon injecting perma-cookies to track mobile customers, bypassing privacy controls. www.eff.org/deeplinks/2014/11/verizon-x-uidh, November 2014.

[17] M. Ivanovich, P. Bickerdike, and J. Li. On TCP performance enhancing proxies in a wireless environment. *IEEE Comm. Mag.*, 46(9), 2008.

[18] A. M. Kakhki, F. L. D. R. Choffnes, A. Mislove, and E. Katz-Bassett. Bingeon under the microscope: Understanding T-Mobile's zero-rating implementation. In *SIGCOMM Internet-QoE Workshop*, 2016.

[19] A. M. Kakhki, A. Razaghpanah, A. Li, H. Koo, R. Golani, D. R. Choffnes, P. Gill, and A. Mislove. Identifying traffic differentiation in mobile networks. In *Proc. of IMC*, 2015.

[20] D. N. Kalofonos, Z. Antoniou, F. Reynolds, M. V. Kleek, J. Strauss, and P. Wisner. Mynet: A platform for secure P2P personal and social networking services. In *Proc. of PerCom*, 2008.

[21] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating the edge network. In *Proc. of IMC*, 2010.

[22] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. van der Veen, and C. Platzer. Andrubis - 1,000,000 Apps Later: A View on Current Android Malware Behaviors. In *Proc. of BADGERS*, 2014.

[23] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson. An End-to-End Measurement of Certificate Revocation in the Web's PKI. In *Proc. of IMC*, 2015.

[24] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici. ClickOS and the art of network function virtualization. In *Proc. of USENIX NSDI*, 2014.

[25] Opera mini browser. www.opera.com.

[26] R. Perdisci, W. Lee, and N. Feamster. Behavioral clustering of HTTP-based malware and signature generation using malicious network traces. In *Proc. of USENIX NSDI*, 2010.

[27] Project fi. fi.google.com.

[28] A. Rao, J. Sherry, A. Legout, W. Dabbout, A. Krishnamurthy, and D. Choffnes. Meddle: Middleboxes for increased transparency and control of mobile traffic. In *Proc. of CoNEXT 2012 Student Workshop*, 2012.

[29] L. Ravindranath, S. Agarwal, J. Padhye, and C. Riederer. Procrastinator: pacing mobile apps' usage of the network. In *Proc. of MobiSys*, 2014.

[30] J. Ren, A. Rao, M. Lindorfer, A. Legout, and D. R. Choffnes. ReCon: Revealing and controlling privacy leaks in mobile network traffic. In *Proc. of MobiSys*, 2016.

[31] F. Roesner, T. Kohno, and D. Wetherall. Detecting and Defending Against Third-Party Tracking on the Web. *Proc. of USENIX NSDI*, 2012.

[32] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: Network processing as a cloud services. In *Proc. of ACM SIGCOMM*, 2012.

[33] Amazon silk browser. www.amazon.com/gp/help/customer/display.html?nodeId=200775440.

[34] A. Sivakumar, V. Gopalakrishnan, S. Lee, S. Rao, S. Sen, and O. Spatscheck. Cloud is not a silver bullet: A case study of cloud-based mobile browsing. In *Proc. of ACM HotMobile*, 2014.

[35] J. Sommers and P. Barford. Cell vs. WiFi: On the Performance of Metro Area Mobile Connections. In *Proc. of IMC*, 2012.

[36] Y. Song and U. Hengartner. PrivacyGuard: A VPN-based Platform to Detect Information Leakage on Android Devices. In *Proc. of ACM SPSM*, 2015.

[37] D. L. Tennenhouse and D. J. Wetherall. Towards an active network architecture. *ACM SIGCOMM Computer Communication Review*, 26(2), April 1996.

[38] The Wall Street Journal. What They Know - Mobile. blogs.wsj.com/wtk-mobile/, December 2010.

[39] D. R. Thomas, A. R. Beresford, and A. Rice. Security metrics for the android ecosystem. In *Proc. of ACM SPSM*, 2015.

[40] N. Vallina-Rodriguez, J. Shah, A. Finamore, H. Haddadi, Y. Grunenberger, K. Papagiannaki, and J. Crowcroft. Breaking for Commercials: Characterizing Mobile Advertising. In *Proc. of IMC*, 2012.

[41] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang. An Untold Story of Middleboxes in Cellular Networks. In *Proc. of ACM SIGCOMM*, 2011.

[42] N. Weaver, C. Kreibich, M. Dam, and V. Paxson. Here Be Web Proxies. In *Proc. PAM*, 2014.

[43] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman. Identifying diverse usage behaviors of smartphone apps. In *Proc. of IMC*, 2011.

[44] X. Xu, Y. Jiang, T. Flach, E. Katz-Bassett, D. Choffnes, and R. Govindan. Investigating transparent web proxies in cellular networks. In *Proc. PAM*, 2015.

[45] K. Zarifis, T. Flach, S. Nori, D. Choffnes, R. Govindan, E. Katz-Bassett, Z. M. Mao, and M. Welsh. Diagnosing path inflation of mobile client traffic. In *PAM*, 2013.