Public Review for

Experience-Driven Research on Programmable Networks

Hyojoon Kim, Xiaoqi Chen, Jack Brassil, Jennifer Rexford

This paper presents the design and deployment of Camp4, a system based on P4 allowing for safe experimentation with network monitoring and measurement on campus networks. It describes the authors' deployment of the system, and applications they run on it, and points out the advantages of building both online and offline packet processing pipelines for research purposes.

The reviewers felt that this paper represents a valuable addition to the research community, especially when the tools will be released. It presents a viable idea and has shown it to work well in practice. The paper covers technical as well as non-technical aspects of real-life testbeds, and explains how to handle these. Also, the source code of the system and its applications will be made available, which makes it possible for the community to build on it.

Public review written by
Bhaskaran Raman
IIT Bombay, India

Experience-Driven Research on Programmable Networks

Hyojoon Kim, Xiaoqi Chen, Jack Brassil, and Jennifer Rexford
Princeton University, USA
{hyojoonk,xiaoqic,jbrassil,jrex}@cs.princeton.edu

ABSTRACT

Many promising networking research ideas in programmable networks never see the light of day. Yet, deploying research prototypes in production networks can help validate research ideas, improve them with faster feedback, uncover new research questions, and also ease the subsequent transition to practice. In this paper, we show how researchers can run and validate their research ideas in their own backyards—on their production campus networks—and we have seen that such a demonstrator can expedite the deployment of a research idea in practice to solve real network operation problems. We present *P4Campus*, a proof-of-concept that encompasses tools, an infrastructure design, strategies, and best practices——both technical and non-technical——that can help researchers run experiments against their programmable network idea in their own network. We use network tapping devices, packet brokers, and commodity programmable switches to enable running experiments to evaluate research ideas on a production campus network. We present several compelling data-plane applications as use cases that run on our campus and solve production network problems. By sharing our experiences and open-sourcing our P4 apps [28], we hope to encourage similar efforts on other campuses.

CCS CONCEPTS

• **Networks → Programmable networks**; **In-network processing**; **Network privacy and anonymity**; **Network experimentation**;

KEYWORDS

Network experiments, Network testing platform, P4

1 INTRODUCTION

For the last few years we have witnessed the steady development and maturation of programmable data planes. The Protocol Independent Switch Architecture (PISA) [7] and P4 [29]—the *lingua franca* for programming customizable pipelines—helps realize novel ideas and run them at line rate. Together, they provide deeper programmability than earlier technologies like OpenFlow [23], allowing researchers to specify packet-processing functionality unencumbered by fixed protocols or standards. More importantly, this technology has matured to the point where practitioners have confidence in commercial products, and a variety of programmable data-plane components [2, 3, 15, 25, 39] and tool chains [27] are available. With these pieces in place, the research community is better positioned to transfer research ideas into practice in production networks.

Yet, crossing the chasm from a high-level research idea to practical impact on production networks remains challenging, as illustrated in Figure 1. While research leveraging data-plane programmability is fast becoming a "cottage industry," many of today's research projects end with a software prototype (say, using the BMV2 behavioral switch model [26]) evaluated by simulation using older public packet traces. While an important step in the research pipeline, we believe researchers can and should go further, to make the ideas stronger and increase their chance of "escaping from the lab." Inspired by the success of experimental platforms like PlanetLab, we advocate *experience-driven research* [31], where researchers could build prototypes of their ideas and evaluate them in-the-wild. Researchers can gain invaluable insights that help them refine their ideas, challenge unspoken assumptions, improve their prototypes, and uncover new and interesting research problems along the way.

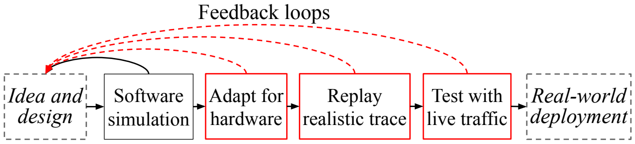

Figure 1: Conceptual research pipeline. Stages in red boxes are hard to reach, resulting in missing feedback loops (red-dashed lines) from them back to the research idea.

Experience-driven research in programmable networking would take a research idea beyond a P4 program that runs on software switches. Although P4 was envisioned as a target-independent language [7], creating packet-processing applications that can "fit" in line-rate hardware switches often requires designing new, hardware-efficient algorithms. This leads to new research challenges that our community can and should tackle (Section 2.1). A second challenge is getting access to suitable traces for evaluation. Although clearly valuable, today's publicly available traces are sometimes too old [5] to represent modern workloads. Other public data, such as CAIDA backbone traces [11], sometimes do not meet the needs of specific research questions, due to capturing only one direction of traffic or missing important header fields. We need effective ways to tailor trace collection to the research task at hand (Section 2.2). To go even further, researchers rarely have a chance to run their experiments against *live* traffic from production networks. The ability to feed live traffic directly to programmable switches running experimental P4 programs would allow running experiments over longer periods of time without incurring too much overhead when compared to capturing and storing traces beforehand (*e.g.*, disk space overhead). This also enables the long-running experiments to produce real-time traffic analytics useful for network operators (Section 2.3).

To this end we present *P4Campus*, an experiment infrastructure that can help academic researchers run experiments in their own production campus network. P4Campus encourages and helps researchers: (i) migrate from software-based simulation to an implementation on hardware switches, (ii) capture and replay packet traces from their own campus network, and (iii) run experiments against live production traffic. We use network tapping devices and network packet brokers to mirror and deliver production traffic to target destinations (*e.g.*, experimental P4 program on hardware). Programmable switches are primarily used for experimenting at line rate, but we also use them as a tool for anonymizing personally identifiable information (PII) and efficiently collecting production packet traces (Section 3). We showcase example P4 applications and our experiences running them on our campus network; we also explain how each experiment helped campus network operations (Section 4). We share non-technical strategies and best practices, especially on ethical data use and stewardship for protecting user privacy (Section 5).

Informed by earlier success in deploying emerging technologies on campus networks [23, 34], we built P4Campus around our production campus network, though we believe our methodology can be applied to any production network. Though initially a convenience, we discovered our campus network to be a surprisingly rich source of research challenges. First, a campus network offers diverse and rich types of traffic (*e.g.*, dorm, science, wireless). Secondly, campus networks are research-friendly. Educational institutions are inherently inclined toward supporting experiments on or against its infrastructure. With increasing support for "campus as lab" initiatives, schools have existing mechanisms and best practices for proper collection, access, and management of campus data, which we leverage.

2 EXPERIENCE-DRIVEN RESEARCH

Experiments with commodity hardware switches in production networks inform and accelerate impactful research, but at the same time introduce new practical challenges. In this section, we discuss several important steps we are taking to build the P4Campus vision.

2.1 From Software to Hardware Data Planes

Figure 1 illustrates that software simulation is an important step in networking research, but certainly not the end of it, especially for research with programmable switches. The P4 behavioral model (BMV2) simulates a programmable switch in software [26], supporting arbitrarily complex P4 programs. Yet, running a P4 program at line rate on hardware programmable switches (*e.g.*, Barefoot Tofino by Intel [2]) is much more challenging; it introduces strict constraints like fixed processing pipeline length, limited memory space, a limited number of memory accesses per packet, and so on [4].

Thus, transforming a P4 application to "fit" into a hardware switch often requires the creation of novel, hardware-efficient data structures. Furthermore, by doing so, researchers can learn valuable lessons for designing fast algorithms, a critical skill for applying other technologies to fast networks and big data. Therefore, P4Campus is designed to primarily help run experiments against hardware programmable switches with experimental P4 apps, not software simulations with BMV2. This encourages researchers to work on research ideas and corresponding P4 programs that actually fit and run in hardware. We recognize that programmable switches and NICs are not the only technologies available today to implement experiments at high rates. However, we see considerable promise in their role in many future applications that supports our focus on advancing the research community understanding of these tools.

2.2 From Generic to Specialized Traces

Packet traces from real networks are useful for demonstrating how well a solution would work against realistic traffic. As such, publicly available packet traces, like the ones from CAIDA [11], are precious resources for researchers. However, packet traces that are suitably representative of modern traffic at differing levels of aggregation are not always easily found or readily available. In some cases targeted traces can be obtained by researchers working with large companies (*e.g.*, ISPs, cloud providers) that have the ability to capture large amounts of packet traces. However, access to such datasets normally requires a privileged relationship and data use agreement.

In contrast, a university researcher's own institution is likely a rich source of diverse set of traffic types with modern workloads. Rather than "loving the data they have", researchers would benefit from having the ability to "capturing the data they need." Such an ability would allow researchers to create and use packet traces that are more tailored to their research. If the network is capable, a researcher can capture traces at multiple vantage points for analyzing a packet's journey through the network. Researchers can also include header fields of interest while removing unnecessary ones for their research. Researchers would also be able to acquire real packet traces much faster and more efficiently, and acquire more up-to-date data that better represents modern workloads. Additionally, the researcher might be able to provide interesting insights about the traffic characteristics, to the benefit of the network operations team.

2.3 From Packet Traces to Live Traffic Feeds

Capturing and replaying packet traces is a powerful research tool, but the technical problems associated with long duration, high-speed network captures are well known (*e.g.*, avoiding packet loss with network and disk I/O optimization [18, 24, 32], improving replay performance [16, 19, 33, 38], and maintaining large disk space).

To this end, there is value in the ability to run an evaluation against *live production traffic* instead of captured packet traces. In addition to avoiding the overhead of capturing, storing, maintaining, and replaying packet traces, such an approach promises better protection of data privacy. For instance, let's assume we have a packet telemetry P4 app that runs in a programmable switch, and it receives mirrored traffic from the campus network. The processing and analysis of traffic are done at line rate in the switch; it can keep limited state in the switch or report aggregated analysis results only when needed. Most importantly, working with live traffic permits deployment of online telemetry solutions that can produce traffic-analysis results in real time. It can also help open the door to running active experiments (*e.g.*, closed-loop autonomous network control).

Working with live traffic likely demands *on-the-fly* anonymization of privacy-sensitive information embedded in packets arriving at *line-rate* before they are consumed by the researcher's apparatus. Designing and demonstrating an infrastructure and pipeline for feeding live but anonymized traffic to an analytics pipeline safely while minimizing the risk to the production network has been an early yet crucial enabling contribution of P4Campus.

3 P4CAMPUS: TEST IDEAS ON CAMPUS

To overcome the challenges in the previous section and successfully deploy research ideas, a campus network needs infrastructure in place and also to follow a number of practices. In this section, we present the core technical components for making P4Campus a reality. In our initial work, we focus on extensive tapping of campus traffic to collect traces and to enable live traffic-analysis experiments, while also enabling active experiments on the campus.

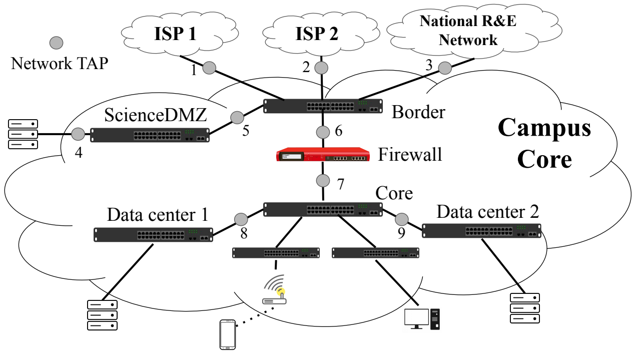

Figure 2: TAPs deployed in the campus core network.

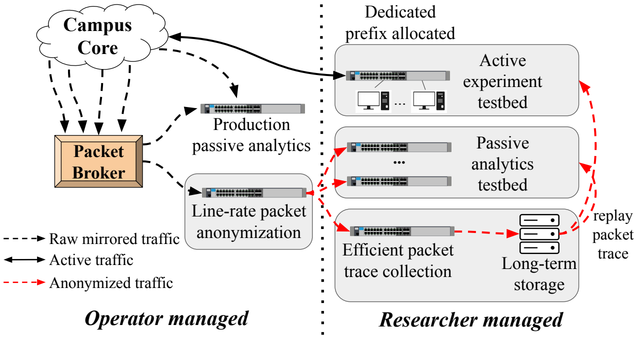
Campus Core
Dedicated prefix allocated
Active experiment testbed
Production passive analytics
Packet Broker
Passive analytics testbed
Line-rate packet anonymization
replay packet trace
Raw mirrored traffic
Active traffic
Anonymized traffic
Efficient packet trace collection
Long-term storage
Operator managed
Researcher managed

Figure 3: A network packet broker selects target tapped live traffic for injection into the experiment testbed.

3.1 Traffic Mirroring Infrastructure

There are two methods to create a mirror, or exact copy, of the traffic that transits a link or a switch port: (i) Test Access Points (TAPs) and (ii) port mirroring [21]. A TAP device is a physical optical splitter that is installed on a network link and taps traffic, creating an exact copy of the traffic transiting the link. Port mirroring is done on the switch, thus requiring switch configuration and resources. Figure 2 shows the network TAP installation in our simplified campus network. Here, we share several best practices:

Invest in TAP devices for accurate measurements: Port mirroring is supported by most, if not all, vendor switches. However, the integrity of the mirrored traffic (*e.g.*, packet arrival time, reordering, or drops) from port mirroring is distorted even under low levels of utilization in the switch [41]. Thus, it is recommended to use TAPs to ensure delivery of the clean copy of the tapped link. TAPs also do not require switch configuration or downtime; once installed, they continue to produce the exact copy of traffic on the link.

Strategically select vantage points: It is important to select vantage points strategically to monitor the traffic that is most interesting and useful. The tapping infrastructure should cover the campus network both *horizontally* and *vertically*. For instance, the link-level heavy-hitter detection use case would analyze traffic on links 1, 2, and 3 in Figure 2 while the queue-level monitoring application for the ScienceDMZ router would require link 4 and 5 (Section 4.1). The tapped link for the RTT application (Section 4.2) depends on the vantage point and the target leg. For example, for measuring the internal leg (*i.e.*, between vantage point and campus hosts), tapping link 6 gives RTT measurements with the firewall while using link 7 gives the measurements without the firewall. More interestingly, tapping both links 6 and 7 and analyzing them together will uncover the latency introduced by the firewall; such flexibility is important for troubleshooting networks with middleboxes.

3.2 Delivery of Tapped Traffic

The next task is to deliver mirrored traffic to a desired destination. One way is to utilize a Network Packet Broker (NPB) system. As shown in Figure 3, a packet broker system can ingest network traffic and forward it to one or more destinations based on a given forwarding policy. Most packet broker systems also provide a list of additional services, including packet filtering, packet deduplication, header removal (*e.g.*, VLAN tag), content masking or payload removal, and so on. However, there is no packet broker system that can perform custom and prefix-preserving anonymization of packet header fields at line rate, which we address ourselves.

There are many commercial NPBs in the market. To minimize our testbed capital expenditure, we rely on our campus's existing Arista's DANZ Monitoring Fabric (DMF) solution [1]. Such a monitoring fabric is a common campus network component that helps operators monitor their network (*approx.* $15,000). We supplement this with dedicated passive optical network TAPs ($500/TAP), P4 switches ($7,000/switch), and analytics servers ($5,000/server).

As previously discussed, personally identifiable information should be anonymized before traffic is delivered to researchers, following strict ethical standards. Technically, there are many available tools for the community to use. For instance, CAIDA employs a rigorous procedure for data anonymization [10], and publicly shares a taxonomy of anonymization tools [9]. In P4Campus, the line-rate traffic anonymizer is the anonymization tool (Section 3.3). We discuss additional ethical issues and non-technical strategies in Section 5.

3.3 Production Packet Traces for Research

Access to traffic traces, recorded or live, is critical for running realistic experiments. However, providing raw traffic to network researchers without first anonymizing personally identifiable information, such as MAC or IP addresses, undoubtedly violates the rights and welfare of human research subjects—the users of the campus network. Thus, all privacy-sensitive information must be first anonymized, and packet payload removed. In addition, a researcher might want to do a longitudinal analysis with historical data; therefore, data must be collected and stored, potentially for long periods of time. We present two P4 apps that address these issues.

Line-rate packet anonymizer: Anonymizing a captured packet trace offline takes a significant amount of time and effort, and existing tools that run on x86 systems cannot keep up with the speed of live traffic. Luckily, programmable data planes make it possible to anonymize traffic at line rate. We developed ONTAS [22] for this purpose. ONTAS ingests mirrored production traffic, hashes relevant header fields such as MAC and IP addresses, and outputs traffic with those fields anonymized. It is possible to customize what fields and which IP prefixes to anonymize by installing corresponding rules in the match-action tables via a control plane, without recompiling and reloading the program. The app can be easily extended to obfuscate other packet header fields if needed (*e.g.*, timestamp, TCP options). To prevent reverse engineering by researchers, the operator should add a *salt* to the hashing algorithm in this anonymizer and keep it secret and change it periodically. The line-rate packet anonymizer was the first P4 app we deployed. It plays an important role in protecting user privacy as we run more experiments on the campus network. We further discuss our efforts to protect user privacy in Section 5.

Efficient packet trace collector: Research often requires replaying recorded traffic or doing longitudinal analysis with historical data. However, storing long periods of traffic with libpcap [36] is prohibitively expensive even without the payload. It is also challenging to capture traces from high-speed links without any packet loss (*e.g.*, due to disk I/O). To this end, we deployed `*Flow` [35], a P4 app that selects features from incoming packets and groups them by flow for efficient logging. For example, if two packets with the same five-tuple arrive, identical information can consolidate into a single "group field" while header fields that differ are recorded separately. As a result, the app outputs a stream of traces in a much more compact data format. It also records the physical link ID and arrival timestamp, allowing us to replay collected traces. The app also allows customization of how to map packets to flows or groups. This enables us to collect traffic traces that scale to terabit rate with a single commodity switch and a server. We also integrated this application with ONTAS, merging them into a single P4 program that runs in a programmable data plane. So far, we have collected months of traces from a moderately loaded 10 Gbps link, without sampling or losing the specified per-packet information.

3.4 P4Campus Testbed Architecture

Passive analytics testbed: As shown in Figure 3, our P4Campus testbed has a stack of Barefoot Tofino switches, where the anonymized mirrored traffic is delivered. This testbed runs passive analytics using incoming mirrored traffic. Experiments with passive measurement and telemetry use cases are first run here. When campus network operators want to use one of the traffic-analytics applications in production, the P4 program is given to the operators so they can run it on the live, unanonymized traffic using a switch they control.

Efficient packet trace data store: For longitudinal analysis of historical data or for replaying traffic that has been previously captured, we utilize the traffic trace collecting P4 application in Section 3.3 and a long-term storage unit for collecting and storing campus traffic traces efficiently.

Active experiment testbed: Replaying passively collected traces has limitations, especially for reanimating interactive applications. To address this, although in its infancy, P4Campus has a component for running active experiments. We run these experiments with hosts controlled by researchers, using a dedicated globally-routable IP prefix allocated to us by campus operators. Having a dedicated address space gives more freedom to researchers, while reducing risk to the campus; operators simply create prefix-based policies for monitoring and controlling this special "research subnet."

4 P4CAMPUS APPLICATIONS

We now showcase a collection of P4 "apps" that run realistic experiments, or are in the process of doing so, using the P4Campus platform (Table 1); the apps in bold font run on hardware switches while others are in transition from software prototype to the hardware switches. The apps represent a progression of increasingly sophisticated hardware-efficient algorithms, including scalable traffic counting, efficient join operations, and cryptography on packet-header fields. In this section, we summarize the applications and the lessons learned from evaluating them on the campus network; the algorithmic advances necessary to fit the applications in the hardware data plane are discussed in more details in separate papers. Running these apps on production traffic gave our network operators new insights into the campus network, helping us build additional momentum for supporting our research on the campus network.

P4 Application	Benefits to Campus Network Operators
Packet anonymizer [22]	Automated packet anonymization process
Packet trace collector [35]	Automated and improved packet capture process
Link heavy hitters [4]	Identified top talkers inside and outside of campus
Queue heavy hitters [13]	Identified perfSONAR as causing microbursts
Round-trip time [14]	Identified high intra-campus latency for WiFi users
OS fingerprinting	Identified OS types of internal and external hosts. SYN with no TCP options are portscans or SYN flood
Traffic by domain	Identified top domains visited by campus hosts
User anonymity [37]	Tested campus IPv6 connectivity to the Internet

Table 1: P4 apps run on P4Campus benefit network operators. The P4 apps in bold font run on real hardware switches.

4.1 Scalable Traffic Counting

As our first two use cases, we run P4 applications that efficiently identify the small number of heavy-hitter flows, on individual links and within packet queues. These applications estimate the contributions of heavy flows to the load on links and queues, without requiring per-flow state.

Link-level monitoring: We run a P4-based heavy-hitter detection algorithm, PRECISION [4], to analyze our campus traffic to and from the public Internet. The algorithm maintains the set of heavy-hitter flow identifiers and their corresponding sizes in the data plane, using the programmable switch's register memory—evicting flows with small counts to make space for new entries as needed. Our analysis showed that a single wired campus host dominates the campus traffic for periods of time, exchanging a lot of data with many different hosts. We have notified campus network operators as we suspected malware or a compromised host. Network operators later informed us that further analysis revealed that the host is a publicly available mirror site that hosts various Linux/UNIX distribution images (*e.g.*, Ubuntu, FreeBSD, and CentOS).

Queue-level monitoring: Our campus network operators struggled with troubleshooting a router experiencing intermittent packet losses (despite low average link utilization) for transferring large scientific datasets. We developed and deployed a P4 application for fine-grained monitoring of packet queues [13] to debug this problem, by analyzing a feed of the legacy router's ingress and egress traffic. The P4 program identifies and reports heavy-hitter flows in the queue, by calculating the time differences between the ingress and egress copies of each packet, and identifying flows with many packets queued at the same time. We found that, ironically, the heavy flows in the queue corresponded to perfSONAR, an active-monitoring tool for diagnosing performance problems [20, 30]. The router's queuing buffer is likely exhausted when many perfSONAR tests run concurrently. Since this discovery, the network operators tuned the perfSONAR configuration to decrease the number of concurrent tests. In fact, our experience with this P4 app has directly triggered interests from a large ISP, who worked with us to deploy the app to detect microbursts in their carrier network [8].

4.2 Efficient Join Operations

More sophisticated traffic analytics require computing a "join" of packets based on header fields. For example, monitoring performance often involves combining information across pairs of packets in the same TCP connection (e.g., a packet and its acknowledgment, or the sequence numbers of a pair of consecutive packets). Also, analyzing traffic by higher-level attributes, such as the domain name (rather than IP address) or end-point operating system (rather than TCP five-tuple), relies on combining information from different packets. Doing this "join" directly in the data plane avoids the overhead of exporting the raw data and protects user privacy by hiding privacy-sensitive "columns" like IP addresses. More generally, doing multiple joins can enable P4 apps that answer questions like "what is the average round-trip time to netflix.com?"

Round-trip time: TCP round trip time (RTT) directly relates to the user's experience of latency. Increased RTT indicates congestion or other abnormal behavior such as routing changes. The P4 app analyzes the continuous RTT experienced by TCP flows, by matching a TCP data packet with its corresponding acknowledgment packet using TCP SEQ and ACK numbers—taking care to minimize erroneous measurements due to delayed ACKs [14]. The time elapsed between the data packet and its acknowledgment corresponds to the RTT between our vantage point (a border router) and the host. Our preliminary results show that wireless hosts experience longer RTTs within the campus, compared to wired hosts. Wireless hosts showed an average internal RTT of 6.7 ms (90th percentile of 8 ms), compared to 1.5 ms (90th percentile of 2 ms) for wired hosts. The increased latency appears to stem from the tunneling of the data traffic through a centralized access point controller, which enables Wi-Fi roaming. Such information helps assess the wireless performance on campus and provides key insights to network operators who plan to expand the wireless infrastructure.

OS fingerprinting: Host OS type is useful information when troubleshooting various performance and security issues. Such information can also provide statistics about the prevalence of host device types (*e.g.*, Android, Apple iOS), particularly in a BYOD setting. To this end, we created a P4 program that identifies host OS types by passively examining TCP SYN packets, a technique used in the popular p0f software tool [40]. We then joined this information with packet counts per client IP address, which is done in the data plane. As the result, we get *packet counts per OS type*, while also removing client IP addresses, which are sensitive information. We ran this P4 app on a campus traffic snapshot and found that 23% of outgoing packets (from campus to Internet) are from Linux/Android hosts while packets from Windows and Apple devices are 35% and 42%, respectively. For incoming packets (from Internet to campus), 52%, 41%, and 6% were from Linux/Android, Windows, and Apple devices, respectively, with the rest from others. We also found that the majority of incoming SYN packets contain no TCP options (77%) and, hence, do not map to a known OS; for outgoing traffic, the proportion was less than 1%. We also observe that among incoming SYN packets that see a following SYN-ACK, the proportion of packets with no TCP options is small (0.035%). This suggests that most incoming SYN packets with no TCP options are the result of adversarial behavior such as port scanning or SYN flooding. Thus, while our app may not be able to fingerprint packets with no TCP options, the absence of TCP options can be an indication of unwanted traffic—useful for defining access-control policies.

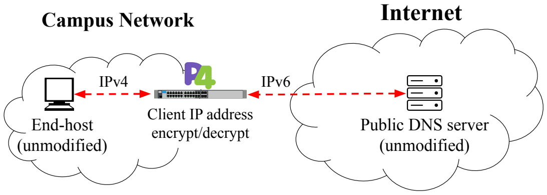

Figure 4: Active experiment with a public DNS server.

Traffic analysis by server domain name: We developed a P4 application that counts traffic by the server domain names, rather than IP addresses. Inspired by earlier work [17], our P4 program identifies and parses DNS response messages, which contain the `A` record that has the IP address information for a queried domain name. The program then joins this mapping with traffic counts per client-server IP address pair. It is also possible to specify domain names using wildcards (*e.g.*, *.edu, *.google.com), enabling the program to aggregate statistics accordingly. As the result, we can get *packet and bytes counts per domain name*. We analyzed a campus trace that captures traffic from wireless and wired campus hosts that are on subnets provisioned for consumer traffic rather than science traffic. We identified the top 20 domains (by byte count), which contribute about 90% of all traffic on campus. The top 20 domains include major sites, including Instagram, Facebook, Google, YouTube, Twitter, Twitch, Amazon, Bing, and Office365, as well as content distribution networks (CDNs) such as Limelight Networks.

4.3 Cryptographic Operations

Active traffic experiments get involved in the live delivery of traffic between a host and the Internet. Such experiments are much more challenging to deploy and run on a production network, compared to passive traffic analytics. Yet, it is important to support them. To this end, we present an active traffic experiment that we first tested with real-world traffic traces but eventually ran as a live experiment from our campus to public DNS servers hosted on the Internet.

Encrypting IP addresses in live traffic to protect user privacy: Campus users reasonably worry about their privacy when accessing public DNS services offered by companies like Google or Cloudflare. We implemented PINOT [37], a lightweight programmable in-network obfuscation solution that encrypts the campus user's IP address when communicating with public UDP-based services. As a P4 application, the service does not require any end-device software installation (*e.g.*, Tor) or coordination other than from the campus network (*e.g.*, IPSec VPN). Our current implementation encrypts the campus user's IPv4 address to one of the public IPv6 addresses the campus owns. Each packet is encrypted independently, without keeping per-flow state in the switch. To run completely in the data plane, we use the Even-Mansour encryption scheme [6], which performs encryption and decryption in a single pass through the packet-processing pipeline of a hardware switch, using only table lookups, permutation, and XORs of bits. As shown in Figure 4, we ran our P4 app to encrypt and decrypt a test client's IP address as it communicates with a public DNS server. To test our prototype with a variety of real DNS queries, we replayed around 3,000 DNS queries to more than 300 public DNS servers; all DNS responses were successfully decrypted and delivered to the synthetic client.

5 ETHICAL DATA USE AND STEWARDSHIP

Some of the key campus stakeholders for researchers to engage are (i) research ethics oversight entities and (ii) administrative data use authorities. Building a foundation of trust between these participants is crucial over the lifecycle of securing necessary permissions, advancing to getting buy-in, executing safe practices, and ultimately sharing learnings and winnings. With data use authorities, communicating that only those parties with *existing data access authority*—namely network operations—interact with unanonymized campus data. And, with research ethics authorities, establishing that research teams receive only *anonymized data or analytics against data*, avoiding contact with potentially sensitive end-user or administrative data. These principles are realized by effectively creating an organizational firewall between our academic research team and the experiment operators.

Our university performs separate reviews to obtain researcher access to administrative data, and to review research ethics and the protection of human subjects. Private and other Personally Identifiable Information (PII) is assumed present in production traffic, and we tap at network locations with varying degrees of traffic aggregation. Prior to data collection we are required to obtain approval to access campus data through the campus Office of Institutional Research, and Institutional Review Board (IRB) evaluation and approval of any activity determined human subjects research. Any use of data is required to be in compliance with applicable laws, regulations, and any restrictions imposed by the sources of the data, including the General Data Protection Regulation (GDPR). The review process is not on a one-off basis; all researchers and research projects are repeatedly reviewed, every two to three years. By leveraging a network packet broker system and its filtering policy, campus operators can also add a separate consent mechanism for end-users (*e.g.*, students, faculty, staff) for collecting anonymized traffic.

Though administrative approvals may differ between campuses, we have found that all stakeholders are best fully informed of all aspects of data handling to ensure best practice adoption and ensure the integrity of the research activity. When approaching our campus partners, we found that it is crucial to stress the following aspects:

Scrubbing private and sensitive data: Our research goals rarely require examination of packet payload data; storing it is time consuming and costly, so this data is discarded. Packet headers containing PII such as a campus user's source MAC and IP addresses are anonymized. The adequacy of our approaches is supported by using well-known tools and published best practices for anonymizing traffic from recognized experts including CAIDA [9, 10]. Our packet tapping architecture feeds a processing pipeline with a line-rate packet anonymizer (see Section 3.3), ensuring sensitive, unanonymized headers are not stored, even briefly.

Handling remote destination IP addresses: Our processing pipeline anonymizes all local (campus prefix-based) IP addresses, and remote (or off-campus) IP addresses where possible. However, in some studies remote IP addresses are used for application or service identification. This is a murky area, as an unhashed IP address potentially can be used to pinpoint a remote recipient on an off-campus network. A workable strategy in this case is to establish explicitly that identifying the individual owner of a remote address: (i) is orthogonal to the research goals, thus will not be attempted, and (ii) is sufficiently challenging to do correctly in general. The reasons for this are plentiful but include that most IP addresses on the Internet belong to ISPs (*e.g.*, Verizon) or large corporations or services (*e.g.*, Amazon, Google), and even non-phantom destinations require considerable effort to confidently identify a particular recipient.

Securely managing data: Our experiment data is typically managed by professional IT staff and kept in a secure location, and only authorized personnel and researchers are allowed to access the data. Data may not be copied or moved to other locations in general. We acknowledge that the organizational coordination required to manage secure data and operate a jointly researcher-managed and campus network operator-managed testbed (Figure 3) is challenging. On our campus we helped bridge this gap by establishing cross-functional campus cyberinfrastructure roles [12] that we envision will support a variety of future Campus-as-a-Lab projects.

6 CONCLUSIONS AND FUTURE WORK

P4Campus demonstrates that academic researchers can "cross the chasm" to evaluate their novel ideas on campus networks, in part by creating, deploying, and running data-plane applications. Our work on P4Campus is just beginning. We plan to expand P4Campus in several directions. First, we seek to support multiple experiments concurrently, by composing multiple apps together into a single P4 program (*e.g.*, [42]). We also expect to diversify our infrastructure with heterogeneous data-plane targets from multiple vendors, and support building a testbed with different network topologies. We plan to improve and expand our mechanism for running more active experiments. Finally, we envision expanding our testbed by working with researchers at other institutions to deploy P4Campus on their campuses and jointly build a larger suite of open-source P4 apps. We have begun this inter-campus tested at regional scale with a first direct fiber connection to a collaborating university with the support of a major regional Research & Education network. We will continue to share our research infrastructure framework and project artifacts with the research community [28].

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers as well as John Sonchack, Liang Wang, Mary Hogan, Praveen Tamanna, and Satadal Sengupta for their helpful feedback. We want to thank Princeton University's Office of Information Technology, Office of Institutional Research, and the Institutional Review Board for reviewing our work and helping us deploy our infrastructure on campus. This material is supported by NSF under Grant No. CNS OAC-2018308 and DARPA under Dispersed Computing No. HR0011-17-C-0047.

REFERENCES

[1] Arista DANZ monitoring fabric. https://www.arista.com/en/products/danz-monitoring-fabric, 2020.

[2] Barefoot (Intel) Tofino Chip. https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series/tofino.html, 2021.

[3] Barefoot (Intel) Tofino2 Chip. https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-2-series.html, 2021.

[4] R. B. Basat, X. Chen, G. Einziger, and O. Rottenstreich. Designing heavy-hitter detection algorithms for programmable switches. *IEEE/ACM Transactions on Networking*, 28(3), June 2020.

[5] T. Benson, A. Akella, and D. A. Maltz. Data set for IMC 2010 data center measurement. http://pages.cs.wisc.edu/~tbenson/IMC10_Data.html, 2010.

[6] A. Bogdanov, L. R. Knudsen, G. Leander, F.-X. Standaert, J. Steinberger, and E. Tischhauser. Key-alternating ciphers in a provable setting: Encryption using a small number of public permutations. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 45–62. Springer, 2012.

[7] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, 2014.

[8] S. Buccapatnam, X. Chen, K. Duell, S. L. Feibish, K. Meier-Hellstern, Y. Koral, S. A. Monetti, A. Raghuram, J. Rexford, J. Stango, et al. Fine-grained P4 measurement toolkit for buffer sizing in carrier grade networks. In *Stanford University. Workshop on Buffer Sizing*, 2019.

[9] CAIDA: Anonymization Tools Taxonomy. https://www.caida.org/tools/taxonomy/anonymization.xml.

[10] CAIDA: Summary of Anonymization Best Practice Techniques. https://www.caida.org/projects/predict/anonymization/.

[11] CAIDA: Data Collection, Curation and Sharing. https://www.caida.org/data/.

[12] USA National Science Foundation: Campus cyberinfrastructure. https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=504748, 2021.

[13] X. Chen, S. L. Feibish, Y. Koral, J. Rexford, O. Rottenstreich, S. A. Monetti, and T.-Y. Wang. Fine-grained queue measurement in the data plane. In *ACM SIGCOMM CoNEXT*, pages 15–29, 2019.

[14] X. Chen, H. Kim, J. M. Aman, W. Chang, M. Lee, and J. Rexford. Measuring TCP round-trip time in the data plane. In *ACM SIGCOMM Workshop on Secure Programmable Network Infrastructure*, August 2020.

[15] Cisco Silicon One. https://www.cisco.com/c/en/us/solutions/service-provider/innovation/silicon-one.html, 2020.

[16] Disk2n: A 10 gigabit network traffic (re)player. https://www.ntop.org/products/traffic-recording-replay/disk2n/, 2020.

[17] S. Donovan and N. Feamster. Intentional network monitoring: Finding the needle without capturing the haystack. In *ACM SIGCOMM HotNets Workshop*, 2014.

[18] DPDK, Data Plane Development Kit. https://www.dpdk.org/, 2018.

[19] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle. MoonGen: A scriptable high-speed packet generator. In *ACM SIGCOMM Internet Measurement Conference*, pages 275–287, 2015.

[20] A. Hanemann, J. W. Boote, E. L. Boyd, J. Durand, L. Kudarimoti, R. Łapacz, D. M. Swany, S. Trocha, and J. Zurawski. PerfSONAR: A service oriented architecture for multi-domain network monitoring. In *International Conference on Service-Oriented Computing*, pages 241–254. Springer, 2005.

[21] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras. Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFiX. *IEEE Communications Surveys & Tutorials*, 16(4):2037–2064, 2014.

[22] H. Kim and A. Gupta. ONTAS: Flexible and scalable online network traffic anonymization system. In *ACM SIGCOMM Workshop on Network Meets AI & ML*, pages 15–21, 2019.

[23] 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*, 38(2):69–74, 2008.

[24] N2disk: 10/40 gbit network traffic recorder with indexing capabilities. https://www.ntop.org/products/traffic-recording-replay/n2disk/, 2020.

[25] Netronome Agilio SmartNIC. https://bit.ly/2UOGfi8, 2016.

[26] P4 language consortium behavioral model (bmv2). https://github.com/p4lang/behavioral-model, 2020.

[27] P4-NetFPGA. https://github.com/NetFPGA/P4-NetFPGA-public/wiki, 2020.

[28] P4Campus: framework, applications, and artifacts. https://p4campus.cs.princeton.edu, 2021.

[29] P4 Specification [Online]. https://p4.org/specs/, 2019.

[30] perfSONAR. https://www.perfsonar.net, 2020.

[31] L. Peterson and V. S. Pai. Experience-driven experimental systems research. *Communications of the ACM*, 50(11):38–44, 2007.

[32] PFRING: High-speed packet capture, filtering and analysis. https://www.ntop.org/products/packet-capture/pf_ring/.

[33] L. Rizzo. Netmap: A novel framework for fast packet I/O. In *USENIX Security Symposium*, pages 101–112, 2012.

[34] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. M. Parulkar. Can the production network be the testbed? In *OSDI*, volume 10, pages 1–6, 2010.

[35] J. Sonchack, O. Michel, A. J. Aviv, E. Keller, and J. M. Smith. Scaling hardware accelerated network monitoring to concurrent and dynamic queries with *Flow. In *USENIX Annual Technical Conference*, pages 823–835, 2018.

[36] TCPDUMP and Libpcap. https://www.tcpdump.org, 2020.

[37] L. Wang, H. Kim, P. Mittal, and J. Rexford. Programmable in-network obfuscation of DNS traffic. In *NDSS: DNS Privacy Workshop*, 2021.

[38] K. Wiles. Pktgen-dpdk. http://github.com/Pktgen/Pktgen-DPDK/, 2020.

[39] Xilinx Netcope P4. https://www.xilinx.com/products/intellectual-property/1-pcz517.html, 2020.

[40] M. Zalewski. p0f v3 (version 3.09b). http://lcamtuf.coredump.cx/p0f3/, 2014.

[41] J. Zhang and A. Moore. Traffic trace artifacts due to monitoring via port mirroring. In *Workshop on End-to-End Monitoring Techniques and Services*, pages 1–8. IEEE, 2007.

[42] P. Zheng, T. Benson, and C. Hu. P4visor: Lightweight virtualization and composition primitives for building and testing modular programs. In *ACM SIGCOMM CoNEXT*, pages 98–111, 2018.