Deploying Research Ideas on Your Campus Network

Paper #51, 6 pages

ABSTRACT

Many promising networking research ideas never see the light of day. Yet, deploying research prototypes in production networks can improve networking practice, while providing the feedback necessary to make good ideas better quickly. We argue that academic networking researchers can and should demonstrate their research ideas in their own backyards—on their campus networks. We have found that using commodity programmable switches to create, deploy, and run experimental data-plane applications speeds the innovation pipeline between the conception of a novel idea and its deployment. We present our Camp4 infrastructure and several compelling applications running on our campus and solving production network problems. We also describe policies, strategies, and tactics for overcoming obstacles to working with production traffic. By sharing our experiences and open-sourcing our P4 apps, we hope to encourage similar efforts on other campuses.

1. INTRODUCTION

The challenge of introducing innovations in production networks is well known, and both researchers and practitioners miss out on opportunities that could improve network operation. To illustrate this, Figure 1 depicts a conceptual research pipeline in computer networks. Starting with an idea and design, a researcher aims to progress through the stages, and ultimately evaluate an idea through a deployment with production traffic. Most importantly, each stage presents a *feedback path* back to the first stage, which is critical for refining the idea and design. Shortening these feedback loops reduces the time to meaningful *impact* in production settings. Yet, researchers frequently hit a roadblock at the later stages, preventing their ideas from ever "escaping from the lab."

We believe that two seemingly unrelated opportunities present academic researchers with a way to overcome these roadblocks: (i) the emergence of commodity programmable networking equipment; and (ii) a timely willingness to conduct *campus-as-lab* experiments to address growing and diverse network demands. In combining these opportunities, we are informed by earlier successes in deploying emerging technologies on campus networks [15, 20].

Why programmable networking? For the last few years we have witnessed the steady development and maturation

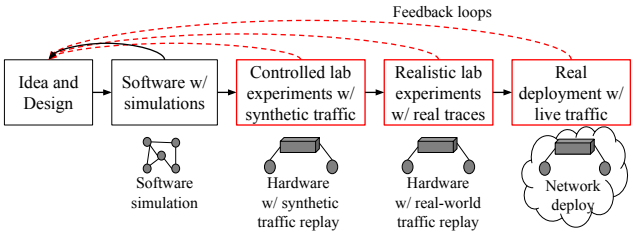

Figure 1: Research pipeline.

of programmable data planes. The Protocol Independent Switch Architecture (PISA) [6] and P4 [18]—the *lingua franca* for programming customizable pipelines—help realize novel ideas and run them at line rates. Together, they provide deeper programmability than earlier technologies like *OpenFlow*, allowing researchers to specify packet-processing pipelines unencumbered by fixed protocols or standards.

Now a variety of programmable data-plane components [2, 3, 16, 23, 10] and toolchains [17] are commercially available. This technology, once disruptive but immature, has advanced to the point where practitioners have confidence in the technology, with a variety of vendors becoming potential partners. This helped overcome prior reservations to operational deployment of research prototypes. Recognizing this shift, we believe the time is right to push for significant changes in how networking research can and should be done.

Why use campus networks? Networking researchers historically have found it difficult to deploy prototypes on campus networks. Yet, it is an ideal environment to validate ideas. Campus networks have unique characteristics that make them an attractive testbed—a microcosm of the many network settings that the research community explores.

Diverse and rich traffic types: A campus comprises many diverse units. For instance, the dormitories and faculty/staff housing generate traffic similar to residential traffic off campus. Academic departments generate "science" traffic, some characterized by big data and large bulk data transfers. Many campuses operate a local data center for high-performance computing and virtualized campus services, generating traffic similar to commercial data-center traffic. A campus also may permit visitor or public "bring your own device" wireless network access. All in all, campus network traffic is rich and

P4 Application	Benefits to Campus Network Operators
Packet anonymizer	Automated packet anonymization process
Packet collector	Automated packet capture and delivery process
Link heavy hitters	Identified top talkers inside and outside of campus
Queue heavy hitters	Identified perfSONAR as causing microbursts
Round-trip time	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 floods
Domain name join	Identified top domains visited by campus hosts
User anonymity	Tested campus IPv6 connectivity to the Internet

Table 1: Camp4 applications benefit network operators.

interesting, making a campus a valuable testing ground.

Research-friendly: Educational institutions are inherently research-friendly and inclined toward supporting experiments on or against its infrastructure. Schools increasingly support "campus as lab" initiatives, in a range of areas such as energy efficiency, sustainability, and public safety. Using existing mechanisms and best practices we introduce later, including proper collection, access, and management for campus data, we find campus network operators and administrative authorities (*e.g.*, the Institutional Review Board, or IRB) to be supportive partners for networking research as well.

Deploying research ideas on a production campus network is still challenging. To our delight, we find that, with the help of proper guardrails, many barriers can be cleared. In this paper, we share our successes and lessons learned. Section 2 presents experiments we have deployed so far. Section 3 describes the architecture and technical requirements for Camp4. Section 4 discusses strategies and practices for successful engagements with our campus partners, and ethical considerations for our research. Section 5 lists future plans and completes our case for using the campus as a laboratory.

2. CAMP4 APPLICATIONS

Table 1 lists Camp4 use cases that demonstrate the potential of our approach. The first two applications are also enablers that help us reach the fourth and fifth stages in Figure 1 (Section 2.1). The passive traffic-analytics applications are example use cases of the fourth and fifth stages—experiments with real traces and deployments with live production traffic (Section 2.2). Finally, the last application is a deployment example that gets involved in the live delivery of traffic between a campus host and the Internet (Section 2.3).

2.1 Production Packet Traces for Research

Access to real-world traces, recorded or live, is critical for running realistic lab experiments and real deployments (the fourth and fifth stages in Figure 1). 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 must be 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. Note that, once the Camp4 infrastructure is in place, it is easy to deploy new research ideas. For instance, we were able to deploy the packet trace collection app in the middle of the lockdown due to the COVID-19 pandemic, *without even visiting the campus*.

Line-rate traffic 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 [14]. The P4 app 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 (which is easy to configure) and keep it secret and change it periodically. The line-rate traffic anonymizer was the first P4 app we deployed. At the same time, it is also an *enabler* of the project. This application enabled us to next run multiple passive analysis programs against anonymized production traffic without first capturing and storing traffic traces, which would be expensive in terms of storage, compute, and time.

Efficient collection of packet traces: Research often requires replaying recorded traffic or doing longitudinal analysis with historical data. However, storing long periods of network traffic with libpcap [22] is prohibitively expensive even without the payload. It is also challenging to capture traces from high-speed links without any packet loss due to bottlenecks in tools or disk I/O. To this end, we deployed a P4 application that selects features from incoming packets and groups them by flow for efficient logging [21]. For example, if two packets with the same five-tuple arrive, identical information can consolidated into a single "group field" while header fields that differ are recorded separately. The application monitors and then outputs a stream of traffic traces in a much more compact data format. The application also provides flexibility on how to map packets to flows or groups. This allows us to collect traffic traces that scale to terabit rate with a single commodity switch and a server. We also integrated this application with the line-rate traffic anonymizer, merging them into a single P4 program that runs in a single programmable data plane. So far, we have collected a month of traffic traces from a moderately loaded 10 Gbps link, without sampling or losing the specified per-packet information.

2.2 Passive Traffic Analytics

The above P4 applications can offer safe access to real-world traces, captured or in real time. With this barrier gone,

we can run realistic lab experiments and even deploy research ideas in a campus network. As a next step, we tackle research ideas that perform *passive* traffic analytics because: (i) passive applications do not disrupt the production network and (ii) the analysis results provide network operators with much-needed visibility into network conditions. In addition, data-plane traffic analysis is an enabler for future work that has the network react to analysis results inline, in real time.

2.2.1 Traffic counting: Heavy hitter detection

As our first passive analytics use cases, we run P4 applications that efficiently identify the small number of heavy-hitter flows, on individual links and within packet queues.

Link-level monitoring: We run a P4-based heavy-hitter detection algorithm [4] to analyze the 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 via Internet2. We deployed a P4 application for fine-grain monitoring of packet queues [9] 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 [12, 19]. 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.

2.2.2 Performance monitoring

In addition to counting the traffic, P4 applications can passively monitor the network performance experienced by our campus users. Unlike active probing, the passive monitoring allows us to scrutinize the performance users *actually* experience, and does not inject additional traffic into the network. There are several metrics for monitoring traffic performance: throughput, packet loss, out-of-order packets, and so on. Here, we describe a P4 app that monitors latency via continuous round-trip-time measurements.

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. Our P4 program analyzes the RTT experienced by TCP flows in our campus traffic, by matching a TCP data packet with its corresponding acknowledgment packet using TCP SEQ and ACK numbers—taking care to avoid erroneous measurements due to delayed acknowledgments. The time elapsed between the data packet and its acknowledgment corresponds to the RTT between our vantage point (a border router) and the user. 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 (with a 90th percentile of 8 ms), compared to 1.5 ms (with a 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 is helpful for assessing the wireless performance on campus.

2.2.3 Joining with higher-level abstractions

Network operators often want to analyze traffic and apply policies based on higher-level end-point identifiers. For example, it can be easier to monitor and configure a network using domain names instead of numerical IP addresses (*e.g.*, round-trip time to Netflix.com is 5 ms on average). To achieve this, we often need to *join* two different kinds of data (*e.g.*, domain-to-IP mapping and IP-to-RTT mapping). Such *inner-joining* of streams in the data plane has an additional benefit: it is possible to remove privacy-sensitive "columns" of information before exporting the analysis results from the data plane. We present two example P4 applications that do this.

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 [24]. We then joined this information with packet counts per client IP address, which is done in the data plane. As the result, we are able to get *packet counts per OS type*. This operation also removes client IP addresses, which can be sensitive. Running this P4 application on a snapshot of our campus traffic, we found that 23% of outgoing packets (from campus to Internet) are from Linux hosts while packets from Windows and Mac OS are 35% and 42%, respectively. For incoming packets (from Internet to campus), 52%, 41%, and 6% were from Linux, Windows, and Mac OS, respectively. The rest was from others. While analyzing the incoming campus traffic, 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

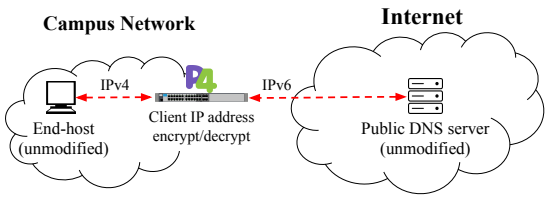

Figure 2: Active experiment with a public DNS server.

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 perform fingerprinting on packets with no TCP options, the absence of TCP options can be an indication of unwanted traffic.

Traffic analysis by server name: We developed a P4 application that counts traffic by the server domain names, rather than IP addresses. Inspired by earlier work [11], 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.

2.3 Active Traffic Experiments

Active traffic experiments are much more challenging to deploy and run on a production network, compared to passive traffic analytics. However, research efforts should not stop at supporting passive experiments; a campus network should aim to also support the deployment of network experiments that get involved in the live delivery of traffic between a campus host and the Internet. 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 a public DNS service hosted on the Internet.

Protecting user anonymity on the Internet: Campus users reasonably worry about their privacy when accessing public DNS services offered by companies like Google or Cloudflare. We implemented a lightweight in-network anonymity 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 [5], which can perform 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 2, 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 more than 3000 DNS queries (from our traces) to more than 300 public DNS servers; all DNS responses were successfully decrypted and delivered to the synthetic client.

3. CAMP4 INFRASTRUCTURE

To deploy the applications mentioned in the previous section, 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 Camp4 a reality. Although we rely heavily on link tapping, our long-term goal is to run P4 applications *in the network*, or inband.

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 [13]. 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 3 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 [25]. Thus, it is recommended to use TAPs to ensure delivery of the clean copy of the tapped link. Network TAPs also do not require switch configuration or downtime; once installed, it continues 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 analyzes traffic on links `1`, `2`, and `3` in Figure 3 while the queue-level monitoring application requires link `4` and `5` (Section 2.2.1). The tapped link for the RTT application (Section 2.2.2) depends on the vantage point and the target leg. For example, for measuring the internal leg (*i.e.*, between campus and the Internet), tapping and using link `6` gives RTT measurements without the firewall while using link `7`

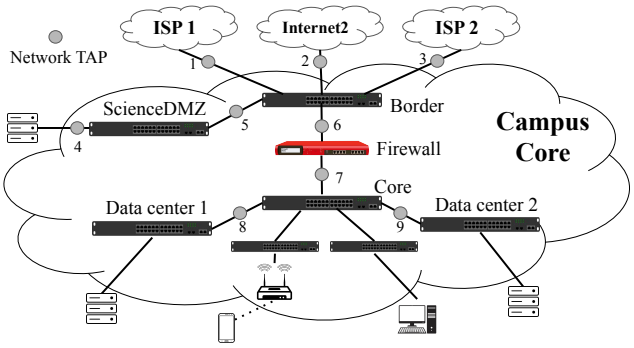

Figure 3: TAPs deployed throughout the campus core network.

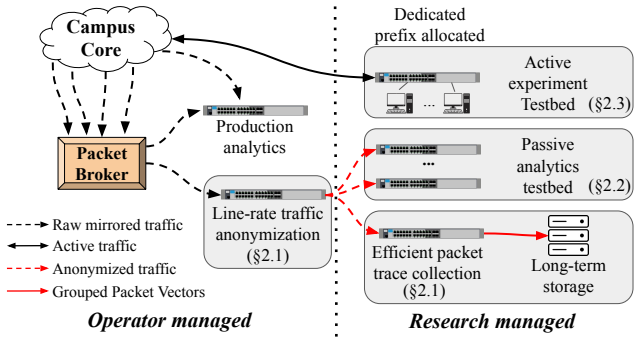

Figure 4: A network packet broker selects target tapped live traffic for injection into the experiment testbed.

gives the measurements with 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 4, 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.

There are many commercial network packet broker products in the market. In our campus, we use Arista's DANZ Monitoring Fabric (DMF) solution [1]. Note that we do not advocate any particular product, and each campus should select a network packet broker that best meets its needs.

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 [8], and publicly shares a taxonomy of anonymization tools [7]. In Camp4, the line-rate traffic anonymizer is our anonymization tool (Section 2.1). We discuss additional ethical issues and non-technical strategies in Section 4.1.

3.3 Camp4 Testbed Architecture

Passive analytics testbed: As shown in Figure 4, our Camp4 testbed has a stack of Barefoot Tofino switches, where the anonymized mirrored traffic is delivered. This testbed is dedicated to running passive analytics using incoming mirrored traffic. All passive measurement use cases in Section 2.2 are first run and tested here. When campus network operators want to use one of our traffic-analytics applications, 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 2.1 and a long-term storage unit for collecting and storing campus traffic traces efficiently.

Active experiment testbed: Although in its infancy, Camp4 has a component for running active experiments. We run these experiments with hosts controlled by researchers, utilizing a dedicated globally-routable IP prefix allocated to Camp4 by the campus operators. Having a dedicated address space gives more freedom to the researchers, while reducing risk to the campus; operators can simply create prefix-based policies for monitoring and controlling this special "research subnet."

4. FACILITATING CAMPUS DEPLOYMENT

Successfully deploying experiments on a production infrastructure takes time, effort, and special care. Nonetheless, we have found it worth the cost, and our campus partners have shown a considerable willingness to support academic partners. The key campus stakeholders for researchers to engage are (i) research ethics oversight entities; (ii) administrative data use authorities; and (iii) network operations professionals. 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.

The overarching principle governing a successful partnership with IT colleagues is simply *do no harm* to the production network. With data use authorities, communicating that only those parties with *existing data access authority*—namely network operations—interact with 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.

4.1 Ethical Data Use and Stewardship

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 vary-

ing 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.

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 initially approaching our campus partners, we found that it is crucial to stress the following aspects of our research:

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 [7, 8]. Our packet tapping architecture feeds a processing pipeline with a line-rate traffic anonymizer (see Section 2.1), 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 a 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 AWS, 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 are allowed to access the data. Researchers have no access to data, and data may not be copied or moved to other locations is general.

4.2 Getting Buy-in from Network Operations

Campus IT organizations stand to benefit directly from networking research experiments deployed on campus. We have found that the following practices facilitate mutual understanding and outcomes viewed as joint successes.

Minimizing risk with passive analytics: Deploying and running a programmable data plane with a custom packet-processing pipeline in an operating network can be disruptive and risky. Network operators reasonably feel discomfort when researchers approach to ask about deploying an experimental switch in the production campus network, where their goal is to maintain 'five 9s' availability. Successful relationship building calls for initially proposing *passive measurements* with production traffic, and a willingness for researchers to use existing and familiar tools (e.g., in our case the monitoring fabric discussed in Section 3). Achieving succeess with an initial passive analytics Camp4 deployment builds trust needed for subsequent active experimentation.

Tackle timely problems for the campus: Normally, operators are fully occupied with managing the production network, which includes provisioning, upgrading, replacing, monitoring, and troubleshooting the network. As such, we initially sought to identify ways to tackle existing operational problems that could *immediately* benefit the campus network (Table 1). Passive analytic projects in Section 2.2 are notable examples of research activities that led to mutual improved understanding of campus network behavior. Even the active experiment in Section 2.3 helped our IT group to configure and test the campus' IPv6 connectivity to the Internet, which aligned with a major upcoming IT networking project. To our pleasant surprise we found this IPv6 connectivity experiment to be as challenging, interesting, rewarding, and perhaps more impactful than activities we originally envisioned.

Creating joint IT-Research roles: For initiatives such as Camp4, stakeholder communication is key. To facilitate more communication and collaboration between network operators and researchers, our campus created opportunities for joint appointment with the Office of Information Technology (OIT) and the Computer Science department. The idea is to have personnel with two hats: one as a networking researcher and the other as an IT staff member who works on real challenges faced in the production network. While a seemingly small tactical step, having the dual responsibilities of balancing research and network operations embodied in a professional staff member was a crucial step to make Camp4 real.

5. CONCLUSIONS AND FUTURE WORK

Camp4 demonstrates that academic researchers can "cross the chasm" to deploy their novel ideas on campus networks, in part by creating, deploying, and running experimental data-plane applications. Our work on Camp4 is just beginning. We plan to expand Camp4 in several directions. First, we seek to support multiple experiments concurrently, by composing multiple apps together into a single P4 program. We also expect to diversify our infrastructure with heterogeneous data-plane targets from multiple vendors. Finally, we envision expanding our testbed by working with researchers at other institutions to deploy Camp4 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.

6. REFERENCES

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

[2] Barefoot Tofino Chip. https://www.barefootnetworks.com/products/brief-tofino/, 2020.

[3] Barefoot Tofino2 chip. https://www.barefootnetworks.com/products/brief-tofino-2/, 2020.

[4] R. B. Basat, X. Chen, G. Einziger, and O. Rottenstreich. Efficient measurement on programmable switches using probabilistic recirculation. In *International Conference on Network Protocols*, 2018.

[5] 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.

[6] 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.

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

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

[9] 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 Conference on Emerging Networking Experiments and Technologies*, pages 15–29, 2019.

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

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

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

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

[20] 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.

[21] 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.

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

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

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

[25] 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.