

2020 Networked Systems Group



Rui
Maria
Roland

Romain
Edgar
Ege
Tobias

Alexander
Rüdiger
Thomas
Laurent

Roland
Coralie
Albert

ONLINE will surely be one of the adjectives of 2020. Even for our group working on computer networks, I must confess this was more of a bug than a feature. While switching to online lecturing, conferences, and meetings illustrated once more the importance of Internet connectivity (and, hence, of our research), it also made clear that online activities can only complement, not replace, in-presence ones. Teaching and research go beyond “Zoom”.

Yet, despite everything, 2020 was a good year for our group. Teaching-wise, our lectures were, once again, highly rated by the students and I was honored to receive the “Credit Suisse Award for Best Teaching”. Research-wise, we published no less than 9 papers (including 1 SIGCOMM, 3 NSDI, and 2 HOTNETS)—2 of which further went on winning awards. Group-wise, we welcomed 3 new students (Rui, Ege, and Roland) and 1 post-doc (Romain), bringing us to 14 members!

Like many, I hope 2021 will mark a return to normality. I particularly look forward to coming back to the classroom, brainstorming with students, meeting with my colleagues (who knew one can miss faculty meetings?), and travelling to conferences (arguably much less than before, but still). Whatever happens though, 2021 ought to be an exciting year for our group as 3 of our PHD students (Maria, Thomas, and Rüdiger) will graduate. Needless to say, we also have plenty of cool new research ideas in our pipeline. So . . . *stay tuned!*

Laurent VANBEVER
Professor, ETH Zurich

Teaching

Having to teach online allowed me to learn *a lot* about video streaming. I continuously refined my video setup over the weeks (inspired by various YouTubers). Among others, I got myself a green screen (enabling me to “blend myself” in my slides), a good microphone, and a decent lighting setup. As our society continues to ask for more videos (also post-COVID), I see this acquired knowledge as a positive outcome of the situation—one of the few.



Photo: Laurent Vanbever

My video setup

2020 was both a particularly busy *and* fulfilling year for us. Besides transitioning to online lecturing, we completely redesigned our “Advanced Topics in Communication Networks” course, not only revising the materials but also developing a new class-wide project. It was a *lot* of work, but we are very pleased with the results. Students seem to be happy too as our teaching evaluations were very positive, with an average rating of 4.5/5.0 and a median of 5.0/5.0.

Besides, 2020 was particularly successful as I was honored to receive the “Credit Suisse Award for Best Teaching”. This prize is awarded by ETH students to one professor each year. While we are not chasing awards, we are obviously thrilled: with our 2 “Golden Owls” in 2016 and 2018, this is already the third teaching award we receive in my 6 years at ETH!

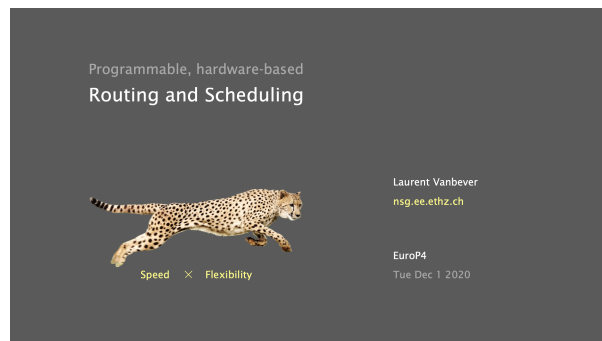


Photo: Oliver Bartenschlager

ETH Tag 2020

Research

I had a lot of fun preparing and giving the keynote at the 3rd P4 Workshop organized in Europe (EuroP4). I spoke about our recent works on making packet scheduling programmable and on offloading routing tasks to hardware. The talk is online—check it out!



EuroP4 Keynote

► [Watch online](#)

Our most common research topics this year were (still) network verification, network programmability, and routing. Regarding verification and programmability, our focus nowadays is on making the technologies more general, more practical, and more usable. We want to allow operators to verify and program more of their networks, in a user-friendly manner. In the context of (Internet) routing, we work mainly on making distributed protocols more flexible and secure.

We continued to garner a strong foothold in top venues such as ACM SIGCOMM, USENIX NSDI, and ACM HOTNETS. This year actually marked the 7th year in a row we published at least one HOTNETS paper. Two of our publications further won awards: our SIGCOMM paper “Probabilistic Verification of Network Configurations” won the best student paper award, and our CCR paper “An Open Platform to Teach How the Internet Practically Works” won the “best of CCR” award.

2020 was also busy regarding community service. Among others, I served in the program committee of USENIX NSDI (writing >20 reviews). I also served as tutorial chair for ACM SIGCOMM (with Dr. Stefano Vissicchio from UCL) and as program chair of SPIN, the first workshop on secure programmable data planes (with Prof. Ang Chen from Rice).

We have plenty of exciting research in our pipeline and look forward to sharing it with you in 2021. Among others, we are working on: network anonymity, seamless network updates, network monitoring, configuration synthesis, fast network convergence, and “machine learning meets networks”.

Network verification

Network verification is all about guaranteeing that a network enforces some important properties such as reachability or isolation. Not all properties need to hold all the time though: often what matters instead is the amount of time each property holds—that is, their probabilities. Such probabilistic properties often appear naturally in the context of Service Level Agreements. In this paper, we show how to accurately compute these probabilities.



Probabilistic Verification of Network Configurations

Samuel Steffen ¹ Timon Gehr ¹ Petar Tsankov ¹
 ETH Zurich, Switzerland ETH Zurich, Switzerland ETH Zurich, Switzerland
 samuel.steffen@ethz.ch timon.gehr@ethz.ch petar.tsankov@ethz.ch
 Laurent Vanbever ¹ Martin Vechev ¹
 ETH Zurich, Switzerland ETH Zurich, Switzerland
 lvanbever@ethz.ch martin.vechev@ethz.ch

ABSTRACT

Not all important network properties need to be enforced all the time. Often, what matters instead is the fraction of time probability these properties hold. Computing the probability of a property in a network relying on complex state-dependent routing protocols, challenging and requires determining all failure scenarios for which the property is violated. Doing so is often not accurately given by brute-force simulation of current network configurations.

In this paper, we introduce *NetDex*, the first scalable and accurate probabilistic network configuration analyzer supporting BGP, OSPF, ECMP, and static routes. Our key contribution is a reference algorithm to efficiently explore the space of failure scenarios. More specifically, given a network configuration and a property, our algorithm automatically identifies a set of links whose failure is provably guaranteed not to change whether or not the property holds across these failure scenarios. *NetDex* manages to accurately approximate *P(S)*. *NetDex* supports practical properties and expressive failure models including correlated link failures.

We implement *NetDex* and evaluate it on realistic configurations. *NetDex* is practical: it can precisely verify probabilistic properties in hours, minutes, even in large networks.

CCS CONCEPTS

• **Mathematics of computing** — Probabilistic inference problems; • **Networks** — Network properties.

KEYWORDS

Network analysis, Failure, Probabilistic inference, Cold edge

ACM Reference Format:

Samuel Steffen, Timon Gehr, Laurent Vanbever, and Martin Vechev. 2020. Probabilistic Verification of Network Configurations. In *Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '20)*, August 19–23, 2020, Virtual Event, NY, USA. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3378743.3409000>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted by ACM for non-profit organizations registered with the Copyright Clearance Center (CCC) Transactional Reporting Service, provided that the fee of \$12.00 is paid directly to CCC. For those organizations that have been granted a photocopy licence by CCC, a separate system of payment has been arranged. The fee code for users copying the copyright material is 0734-2019/20/08-15:15–23. <https://doi.org/10.1145/3378743.3409000>

Figure 4. Comparison of approaches for probabilistic network analysis in a network with 174 links and link failure probability 0.001. The confidence for sampling (Hoeffding's inequality) is $\alpha = 0.95$.

1 INTRODUCTION

Ensuring network correctness is an important problem that has received increased attention [1, 12, 16, 21, 22]. In the existing approaches have focused on verifying “hard” properties producing a binary answer of whether the property holds under all or a fixed set of failure scenarios.

Indeed, hard properties, network operators often need to reason about “soft” properties which can be violated for a small fraction of time (e.g. 99.9%). Among others, allowing properties to be violated allows for cheaper network designs, e.g. by reducing over-provisioning. Soft properties typically emerge when reasoning about compliance with Service Level Agreements (SLAs). SLAs can be defined with respect to any metric (e.g. path availability, average load, capacity, latency) and are traditionally expressed as “time”-type instances, an IP VPN provider might guarantee minimal path availability between the customers for 99.99% time of the time, and two-path availability for 99.99% (five 9s).

Similarly to verifying hard properties, computing the probability of a soft property requires analyzing the network forwarding behavior from network configurations (i.e. the network control plane) in many, possibly all, environments (e.g. failure scenarios). A key difference is that verifying a hard property aims at checking the absence of a counter-example (e.g. a failure scenario in which the property is violated), not at computing how many counter-examples.

This need is exemplified by a survey on clouded-edge network operators [10] (presented in the survey) on network verification rights. Indeed, 80% of the respondents (191) have indicated that it is currently difficult to do so via App A for failure.

ACM SIGCOMM 2020

► Read online

Config2Spec: Mining Network Specifications from Network Configurations

Rüdiger Birkner¹ Dana Drachler-Cohen² Laurent Vanbever¹ Martin Vechev¹
¹ETH Zurich ²Technion

Abstract

Network verification and configuration synthesis are promising approaches to make networks safer, reliable, and secure by enforcing a set of policies. However, these approaches require a formal and precise description of the intended network behavior, imposing a major barrier to their adoption: network operators are not only reluctant to write formal specifications, but often do not even know what these specifications are.

We present *Config2Spec*, a system that automatically synthesizes a formal specification (a set of policies) of a network given its configuration and a failure model (e.g., up to two link failures). A key technical challenge is to design a synthesis algorithm which can efficiently explore the large space of possible policies. To address this challenge, *Config2Spec* relies on a careful combination of two well-known methods: data plane analysis and control plane verification.

Empirical results show that *Config2Spec* scales to mining specifications of large networks (>150 routers).

1 Introduction

Consider the task of a network operator who—instead of human-induced network downtime—decides to rely on formal methods to verify her network-wide configuration [4, 14, 22, 30] or to synthesize them automatically [5, 8, 10, 28, 29]. The operator quickly realizes that both verifiers and synthesizers require a specification of the correct intended network-wide behavior. A few generic requirements quickly come to mind: surely she wants her network to ensure reachability. At the same time, she realizes that her network does not mean just ensuring reachability. Among others, it needs to ensure load balancing for popular destinations, provide isolation between customers, drop traffic for suspicious prefixes, and receive business traffic via predicted waypoints—all these under failures and over hundreds of devices. Writing the precise specification seems daunting, especially as most of it has been

homeworked over years, by a team of network engineers (some of which do not even work there anymore).

This situation illustrates the difficulty of writing network specifications. Alas, to software specifications, formal specifications are hard to write (as hard as writing the program in the first place [20]), debug, and modify [2, 21]. Yet, without easier ways to provide network specifications, network verification and synthesis are unlikely to get widely deployed.

Config2Spec. We introduce *Config2Spec*, a system that automatically mines a network’s specification from its configurations and a failure model (e.g., up to k failures). *Config2Spec* is precise: it returns all policies that hold under the failure model (no false negatives) and only those (no false positives).

Challenge Mining precise network specifications is challenging as it involves exploring two exponential search spaces: (i) the space of all possible policies, and (ii) the space of all possible network-wide forwarding states. The challenge stems from the fact that individually exploring each of the search spaces can be prohibitive: a search for the true policies is hard since they are a small fraction of the policy space, while a search for the violated policies is hard since these require witnessing *data planes*, which are often sparse.

Insights *Config2Spec* addresses the above challenges by combining the strengths of data plane analysis and control plane verification. Data plane analysis enables us to compute the set of policies that hold for a single data plane, thereby providing an efficient way of pruning policies. On the other hand, control plane verification is an efficient way of validating that a search space holds for all the data planes. *Config2Spec* combines the two approaches to prune the large space of policies through sampling and data plane analysis and, then, to avoid the need of exploring all data planes, validating the remaining policies with control plane verification. The key insight is to dynamically identify the approach providing the better failures and over hundreds of devices. Writing the precise specification seems daunting, especially as most of it has been

*Work done while at ETH Zurich.

USENIX NSDI 2020

► Read online

Matha: Network Verifiers Need To Be Correct Too!

Rüdiger Birkner*, Tobias Brodmann*, Petar Tsankov, Laurent Vanbever, Martin Vechev
 ETH Zurich

Abstract

Network analysis and verification tools are often a godsend for network operators as they free them from the fear of including outages or security breaches. As with any complex software though, these tools can (and often do) have bugs. To be fair, this is not surprising: building an accurate and faithful network analysis tool is extremely difficult. Among others, one not only has to precisely capture all the different protocols’ behaviors, but also all of the quirks of their specific implementations. Unfortunately, every vendor, every OS, every device can exhibit slightly different behaviors under certain conditions. For all it takes, these behaviors might be the results of bugs themselves. And yet, failing to accurately capture these behaviors—as we show—can lead to incorrect and possibly misleading analysis results.

A fundamental and yet practical research question is therefore: *How can developers make sure that their network analysis and verification tools are correct?*

We introduce *Matha*, a system that thoroughly tests network analysis and verification tools to find subtle bugs in their network models using black-box differential testing. *Matha* automatically finds model discrepancies by generating input configurations and comparing the output of the tool under test with the output produced by the actual network software. For every discovered discrepancy, *Matha* provides a minimal configuration that helps developers pinpoint the bug. Later on, these configurations can be used to build up an adequate test suite for current and future network tools.

Challenge Precisely identifying bugs in network analyzers models is challenging for at least three reasons. First, the search space of possible configurations is gigantic: there are hundreds of configuration instances, each of which can take many possible parameters. And yet, as our analysis reveals, most of the bugs only manifest themselves when specific configuration statements/conditions are present. Second, systematically exploring the search space is highly non-trivial (as the confidence when pressing “deploy” as you have confirmed the correctness of your configuration update using a real-world on-the-air configuration verifier. A few minutes later, your phone rings: one of your customers can reach the Internet anymore.

1 Introduction

It’s Friday night and you are about to push an important (network) configuration update in production. Usually, you would feel terribly nervous doing so as there is always the possibility that you may have missed something. You are only too aware that misconfigurations happen frequently and can lead to major network outages [15, 17, 20]. Though though you feel confident when pressing “deploy” as you have confirmed the correctness of your configuration update using a real-world on-the-air configuration verifier. A few minutes later, your phone rings: one of your customers can reach the Internet anymore.

USENIX NSDI 2021

Coming soon

Network programmability

Packet scheduling is one of the last bastions standing in the way of complete data-plane programmability. Even recent programmable switches do not allow operators to reprogram their scheduling behaviors. In this paper, we enable programmable packet scheduling in existing hardware switches by approximating the behavior of Push-In First-Out (PIFO) queues. We do so by dynamically adapting how packets are mapped to strict-priority queues.

SP-PIFO: Approximating Push-In First-Out Behaviors using Strict-Priority Queues

Albert Gran Alosa
ETH Zürich

Alexander Dietmiller
ETH Zürich

Laurent Vanbever
ETH Zürich

Abstract
Push-In First-Out (PIFO) queues are hardware primitives which enable programmable packet scheduling by providing the abstraction of a priority queue at line rate. However, implementing them at scale is not easy: just hardware designs (not implementations) exist, which support only about 1k flows. In this paper, we introduce SP-PIFO, a programmable packet scheduler which closely approximates the behavior of PIFO queues using strict-priority queues—of line rate, at scale, and on existing devices. The key insight behind SP-PIFO is to dynamically adapt the mapping between packet ranks and available strict-priority queues to minimize the scheduling errors with respect to an ideal PIFO. We present a mathematical formulation of the problem and derive an adaptation technique which closely approximates the optimal queue mapping without any traffic knowledge. We fully implement SP-PIFO in P4 and evaluate it on real workloads. We show that SP-PIFO: (i) closely matches PIFO, with as little as 8 priority queues; (ii) scales to large amount of flows and ranks; and (iii) quickly adapts to traffic variations. We also show that SP-PIFO runs at line rate on existing hardware (Broadcom Tomahawk), with a negligible memory footprint.

1 Introduction
Until recently, packet scheduling was one of the last bastions standing in the way of complete data-plane programmability. Indeed, unlike forwarding whose behavior can be adapted thanks to languages such as P4 [1] and programmable hardware [2], scheduling behaviors naturally are in sync with hardware implementations that can, at best, be configured. To enable programmable packet scheduling, the main challenge was to find an appropriate abstraction which is flexible enough to express a wide variety of scheduling algorithms and yet can be implemented efficiently in hardware [2]. In [13], Sivaraman et al. proposed to use Push-In First-Out (PIFO) queues as such an abstraction. PIFO queues allow outgoing packets to be pushed in arbitrary positions (according to the packets rank) while being drained from the head.

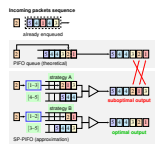


Figure 1: SP-PIFO approximates the behavior of PIFO queues by adapting how packet ranks are mapped to priority queues.

USENIX NSDI 2020

► Read online

If you have ever programmed a hardware-based programmable switch, you know that its resources are limited and come at a huge premium. And yet, it might be that your network traffic is such that your switch only use a small portion of its precious resources. In this paper, we show how making the compiler traffic-aware enables it to allocate resources in a smarter way—think profile-guided optimizations for programmable data planes.

P²GO: P4 Profile-Guided Optimizations

Patrick Wintermeyer
ETH Zürich
patrickw@ethz.ch

Alexander Dietmiller
ETH Zürich
aldietm@ethz.ch

Maria Apostolaki
ETH Zürich
apostolaki@ethz.ch

Laurent Vanbever
ETH Zürich
laurentv@ethz.ch

ABSTRACT
Programmable devices allow the operator to specify the data-plane behavior of a network device in a high-level language such as P4. The compiler then maps the P4 program to the hardware after applying a set of optimizations to minimize resource utilization. Yet, the lack of context restricts the compiler to conservatively account for all possible inputs—including unrealistic or infrequent ones—leading to sub-optimal use of the resources or even compilation failures. To address this inefficiency, we propose that the compiler leverage insights from actual traffic traces, effectively introducing a broader spectrum of possible optimizations. We present a system working alongside the compiler that uses traffic-awareness to reduce the allocated resources of a P4 program by: (i) removing dependencies that are not exercised; (ii) adjusting table and register sizes to reduce the pipeline length; and (iii) offloading parts of the program that are rarely used to the controller. Our prototype implementation on the Tofino switch automatically profiles the P4 program, detects opportunities and performs optimizations to improve the pipeline efficiency. Our work showcases the potential benefit of applying profile-guided techniques used to compile general-purpose languages to compiling P4 programs.

ACM Reference Format:
Patrick Wintermeyer, Maria Apostolaki, Alexander Dietmiller, and Laurent Vanbever. 2020. P²GO: P4 Profile-Guided Optimizations. In *Proceedings of the ACM SIGCOMM Workshop on Network Programmability*, 201–210, 10 pages. https://doi.org/10.1145/3392444.3429194

1 INTRODUCTION
Thanks to programmable data planes, network programmers can now define the forwarding behavior of their switches using programming languages such as P4 [14]. To ensure portability across platforms, these languages abstract away many hardware details and rely on a compiler to “map” programs to the available hardware resources. Typical hardware resources include the number of processing stages, the amount of resources available in each stage, as well as the number of operations available per packet. Compiling P4 programs to make digital or half custom is at the core of this work. The presented hardware is generalizable to other programmable devices, but not restricted to the specific resource allocation and that covers the entire set of the resources requested by state machines that are not modeled in practice. We then show that profiling can find opportunities for memory optimizations while verifying that they do not affect the overall behavior of the P4 program. We finally show that profiling can remove code segments that are rarely used but consume significant resources. Such segments are good candidates to be offloaded to software.

ACM HotNets 2020

► Read online

Early 2020 we decided to release all the materials we used for our master lecture “Advanced Topics in Communication Networks” including: our lecture slides, a set of comprehensive P4 examples, documented P4 exercises (with solutions), and a complete production environment (P4-utils) which makes it easy to build, run, and debug P4 networks. Since then many people have started to use/build upon our resources. Why don’t you take a look?

master - 1 branch 0 tags

Go to file Add file + Code -

edgar-costa tcpdump f451447 on 1 Oct 41 commits

- demo BIG UPDATE: 11 months ago
- documentation tcpdump: 3 months ago
- examples BIG UPDATE: 11 months ago
- exercises fixed scapy problem introduced at 2.4.4 when building the CpuReader: 3 months ago
- slides Updated links to the website to 2019: 7 months ago
- vm small changes in the vm build scripts: 3 months ago
- gitignore updated some scripts, gitignore, cleaned a bit: 2 years ago
- LICENSE Create LICENSE: 2 years ago
- README.md Update README.md: 7 months ago

README.md

P4 Learning

This repository contains a compilation of useful resources for data plane programming, specially for the ones wanting to learn how to write P4-16 programs and test them in a virtual environment.

A big part of the materials come from the Advanced Topics in Communication Networks lecture taught at ETH Zurich. For more information visit our [website](#).

What will you find here ?

- Slides: deck of slides that go from the story of SDN and introduction to data plane programming to advanced (research level) applications.
- Documentation: list of links and documents with very useful information for P4 development.
- Examples: a collection of examples showing how to use almost all the simple switch features.
- Demos: a collection of demos with running examples.
- Exercises: a set of P4 exercises with a long description and solutions.
- Environment Installation: a guide and scripts to install the required software to start developing P4 applications in your own machine

How to start?

Clone this repository into your machine

If you want to solve the exercises, run the examples or simply download the content get a local copy of this

P4-learning repository

► Browse online

Internet routing

If you are a network operator, you are most likely frustrated by the slowness at which you can deploy new features in your network. It can literally take *years* between an initial idea and its corresponding standardization by the Internet Engineering Task Force (IETF). And while SDN promises to solve this problem, it is not a panacea either. In this paper, we propose a lightweight alternative with an API that allows to (easily) reprogram routing protocols logic.

A key element of our “Communication Networks” lecture is our routing project in which we have the entire classroom (150 students in 2020) build and operate their very own “mini-Internet”. We have found that students not only love the project but also learn a *ton* from it. In this paper, we describe our experiences and the platform we built to support class-wide assignments. We have open-sourced all our resources. Check them out: www.mini-inter.net



It is notoriously known that Internet routing is vulnerable to misconfigurations and attacks. And while efforts to secure the Internet are underway, the pace of progress has been (frustratingly) slow. In this paper, we survey how routing attacks can also compromise the security of critical applications like Tor, certificate authorities, or the bitcoin network. The good news though? Protecting an application is *much* easier than protecting the entire Internet.

xBGP: When You Can't Wait for the IETF and Vendors

Thomas Wirtgen
ICTEAM UCLouvain
Louvain-la-Neuve, Belgium
thomas.wirtgen@uclouvain.be

Quentin De Coninck¹
ICTEAM UCLouvain
Louvain-la-Neuve, Belgium
quentin.deconinck@uclouvain.be

Randy Bush²
IBM Research & Almaden
Hawthorne, CA, USA
randyb@ppc.com

Laurent Vanbergh
NSIC, ETH Zurich
Zürich, Switzerland
lvanderbergh@ethz.ch

Olivier Bonaventure³
ICTEAM UCLouvain
Louvain-la-Neuve, Belgium
olivier.bonaventure@uclouvain.be

1 INTRODUCTION

Put yourself in the shoes of a mobile application developer who needs to support old mobile platforms. Your application would clearly benefit from being able to seamlessly switch from WiFi to cellular without impacting the established TCP connections. Mobile path TCP [14] supports such handovers, but you don't have the luxury to wait until its adoption by all smartphone vendors. Instead, you will likely resort to integrating a QoS library in your software to benefit from its connection migration capabilities [23, 25].

Consider this situation with the one of network operators whose networks could really benefit from new network-level features such as improved traffic engineering protocols [1], faster convergence mechanisms [3], or improved DDoS protection [14]. Like our programmer, network operators may highly heterogeneous environments, composed of a wide variety of device types (routers, switches, middle boxes), coming from distinct vendors, and running distinct operating systems. This heterogeneity is actually necessary, not only to avoid vendor lock-in [5], but also for reason of reliability (cheap or vulnerable devices tend not to affect all users at once).

Unlike our programmer, though, network operators often need the entire network to support the features (not only the endpoints) and so wish some alternative but to wait for the request feature to be (i) standardized by the IETF, (ii) implemented by all vendors, (iii) widely tested and (iv) widely deployed on their network. The standardization process alone often takes years. As an illustration, Fig. 1 depicts the timeline of the standardization of the BGP routing protocol responsible for the BGP routing protocol (BGP) started in 1989 and finished in 2005. BGP is the de facto standard for the Internet and Vendors, by providing the de facto BGP implementation on the Internet as Network Director [20]. November 6–8, 2005, Virtual Internet, USA, New York, NY, USA. <https://www.ietf.org/proceedings/05nov/>

Quentin De Coninck is a Ph.D. (PhD) Postdoctoral researcher.

Permission is made to distribute and reuse this work, provided that the original author and source are credited. This work is licensed under a Creative Commons Attribution 4.0 International License. For more information, see <http://creativecommons.org/licenses/by/4.0/>. This work is licensed under a Creative Commons Attribution 4.0 International License. For more information, see <http://creativecommons.org/licenses/by/4.0/>. This work is licensed under a Creative Commons Attribution 4.0 International License. For more information, see <http://creativecommons.org/licenses/by/4.0/>.

© 2020 ACM. This work is licensed under a Creative Commons Attribution 4.0 International License. For more information, see <http://creativecommons.org/licenses/by/4.0/>.

An Open Platform to Teach How the Internet Practically Works

Thomas Holterbach
ETH Zurich
thomashol@ethz.ch

Tobias Bühler
ETH Zurich
buehler@ethz.ch

Timo Reilstab
ETH Zurich
timre@ethz.ch

Laurent Vanbergh
ETH Zurich
lvanderbergh@ethz.ch

ABSTRACT

Each year at ETH Zurich, around 100 students collectively build and operate their very own Internet infrastructure composed of hundreds of routers and dozens of Autonomous Systems (ASes). The fun that all networks need to work for the Internet as a whole to work only helps to bring together the entire classroom.

Over the years, the mini-Internet project has become a flagship piece of our networking lecture, one that the new students look forward to. This fun feedback we received from the students has been extremely positive, with comments such as: “it really allows us to apply the theoretical concepts”, “I am quite confident about many things on the Internet now”, and “it is a unique project”.

In this paper, we describe the overall design of our teaching platform, how we use it, and interesting lessons we have learnt over the years. We also make our platform openly available [2].

CCS CONCEPTS

Networks → Network design principles, Network protocols, Public Internet.

1 INTRODUCTION

Most undergraduate networking courses, including ours [2], aim at teaching “how the Internet works”. For the instructor, this typically means painstakingly going through the TCP/IP protocol stack, one layer at a time, following a bottom-up [1] or top-down approach [15]. At the end of the lecture, students (hopefully) have learnt concepts such as switching, routing, and reliable transport, together with the corresponding protocols.

Learning these concepts is not sufficient to understand how the Internet infrastructure works or, alternatively, why it does not work. For this, we think one also needs to understand the ins and outs of how the Internet is operated, which includes topics such as network design, network configuration, network monitoring, and network debugging. Understanding these topics is important as network operators tend to have a hard time diagnosing issues of the Internet, sometimes even due to human-related errors [18].

We argue that an effective way to teach students about Internet operations—one that we have successfully used for the last four years—is simply to let students operate their own mini-Internet.

Forming students into operators. Each year for the last four years, around 100 ETH students have been configured and managed an actual Internet infrastructure composed of hundreds of routers split across 10 Autonomous Systems (ASes). Each group of 2–3 students is responsible for administering, from scratch, one AS composed of multiple hosts, layer-2 switches and layer-3 routers. Each network “operator” works either using BGP, either directly or through Internet Exchange Points (IXPs), which we use the instructor maintains. The students’ goal is identical to the one of actual operators: enabling Internet-wide connectivity between any pair of

Securing Internet Applications from Routing Attacks

Yixin Sun
University of Virginia

Maria Apostolaki
ETH Zurich

Henry Birge-Lee
Princeton University

Laurent Vanbergh
ETH Zurich

Jennifer Rexford
Princeton University

Mung Chiang
Princeton University

Prateek Mittal
Princeton University

ABSTRACT

Attacks on Internet routing are typically viewed through the lens of availability and confidentiality, assuming an adversary that either disrupts traffic or performs eavesdropping. Yet, a strategic adversary can use routing attacks to compromise the security of critical Internet applications like Tor, certificate authorities, and the bitcoin network.

In this paper, we survey such application-specific routing attacks and argue that both application-layer and network-layer defenses are essential and uniquely needed. The good news is that, while deployment challenges have hindered the adoption of network-layer defenses (i.e., secure routing protocols) thus far, application-layer defenses are much easier to deploy in the short term.

1 INTRODUCTION

The Internet is a “network of networks” that interconnects tens of thousands of geographically-administrated networks. The Border Gateway Protocol (BGP) is the glue that holds the Internet together by propagating information about how to reach destinations in remote networks. However, BGP is notoriously vulnerable to misconfiguration and attack. The consequences range from making destinations unreachable (e.g., Google’s routing incident caused widespread Internet outage in Japan [7]), to misdirecting traffic through sensitive locations (China Telecom due to improper routing announcements from a BGP neighbor [26]), to impersonating legitimate services (e.g., traffic to an Amazon DNS server redirected to attacker-controlled DNS queries via hijacked, spoofed IP addresses [8]). Efforts to secure the Internet routing system have been extensive (see survey paper [29, 22, 30, 28]).

In the past few years, we have seen many parties invest in securing routing, not only in the distributed system space (i.e., the network, nodes and users) but also in the user-space to secure web services, including applications like Tor, certificate authorities, and the bitcoin network. Given the slow adoption of secure routing defenses, we show diverse, low-application-specific defenses can be used in the near-term to mitigate routing attacks without requiring global coordination. We believe that application-layer and network-layer solutions and interventions should both be essential to secure Internet applications. Furthermore, by demonstrating the distributed management for users, we hope to motivate the community to re-evaluate efforts to tackle BGP’s many security problems once and for all.

2. ROUTING ATTACKS

Routing attacks occur in the wild, and we get into increasingly prevalent and more sophisticated. An attacker can launch recently-crafted attacks to achieve desired goals,

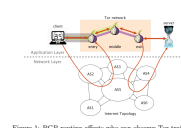


Figure 1: BGP routing affects who can observe Tor traffic.

Looking forward to *see* you in 2021!

Our upcoming lectures

Spring	Communication Networks Seminar in Communication Networks
Fall	Advanced Topics in Communication Networks Discrete Event Systems

Our upcoming PhD graduations

Spring	Maria Apostolaki Thomas Holterbach
Fall	Rüdiger Birkner

Our incoming PhD students

Spring	Tibor Schneider <i>You?</i>
--------	--------------------------------

Zurich, Mon 11 Jan 2021

Visit our webpage at <https://nsg.ee.ethz.ch>