



Cybersecurity (Security+) and P4 Programmable Switches



Jorge Crichigno (jcrichigno@cec.sc.edu)
University of South Carolina
<https://research.cec.sc.edu/cyberinfra/>

Western Academy Support and Training Center (WASTC)
University of South Carolina (USC)

Cisco Systems
260 East Tasman, Building 9
San Jose, California

January 5, 2024

University of South Carolina (USC) CI Lab

- Cyberinfrastructure (CI) Lab members



Jorge Crichigno



Elie Kfoury



Jose Gomez



Ali Mazloum



Ali AlSabeH

University of South Carolina (USC) CI Lab

- USC's CI Lab develops training material for cybersecurity and computer networks
- It conducts applied research using state-of-the-art technology
 - P4 programmable switches
 - Smart Network Interface Cards (smartNICs)
 - Application of machine learning to cybersecurity and networks
- Website

<https://research.cec.sc.edu/cyberinfra/>

Lab Libraries

- Virtual Lab libraries can be installed in NDG's NETLAB+ system and NDG's Online
- Some libraries work on FABRIC, the International programmable network
- Most libraries used open-source software
- A few lab libraries require Windows, Intel agreement



<https://whatisfabric.net/>

Lab Libraries

Cybersecurity

- Virtual Labs on Cybersecurity Tools and Applications
- Virtual Labs on Zeek Intrusion Detection and Prevention Systems

Lab Libraries

SDN and P4 Programmable Data Plane Switches

- Cybersecurity Applications on P4 Programmable Data Planes
- P4 Programmable Data Planes: Applications, Stateful Elements, and Custom Packet Processing
- P4 Programmable Data Plane Switches based on BMv2
- P4 Programmable Data Plane Switches based on Intel's Tofino Chip
- Introduction to Software Defined Networking (SDN)

Lab Libraries

Routing and Switching

- Open Shortest Path First (OSPF)
- Introduction to Border Gateway Protocol (BGP)
- MPLS and Advanced BGP Topics
- Open vSwitch (OvS)

Lab Libraries

Network Monitoring and Management

- Network Management Tools (Netflow, IPFix, sFlow)
- Introduction to perfSONAR
- PerfSONAR 5.0

Lab Libraries

Network Fundamentals

- Network Tools and Protocols (NTP)

Organization of Lab Manuals

Each lab series typically consists of 10-20 guided lab experiments

Each lab experiment includes the 'theory' plus the step-by-step directions

A manual for a lab experiment has the following structure:

Overview

- Objectives
- Lab settings: passwords, device names
- Roadmap: organization of the lab

Section 1

- Background information (theory) of the topic being covered (e.g., malware fundamentals)
- Section 1 is optional (i.e., the reader can skip this section and move to lab directions)

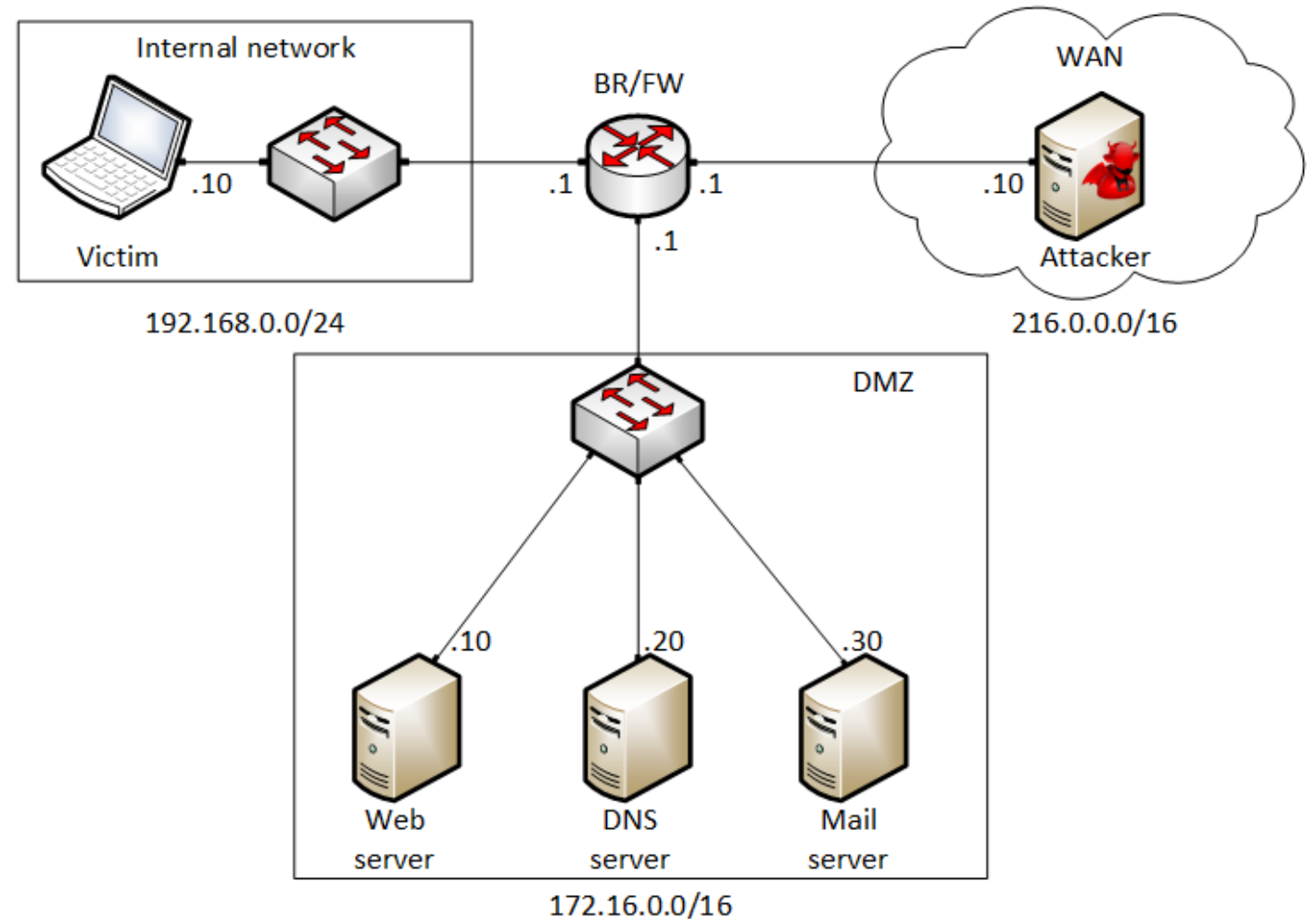
Section 2... n

- Step-by-step directions

Cybersecurity Fundamentals Lab Series

Cybersecurity Fundamentals - POD

- Attacker in the WAN running Kali
- Victim in the internal network running Windows 10
- Web, DNS, and Mail servers in the DMZ zone
- Border router interconnect the networks
- Border router implements basic security policy:
 - Attacker cannot initiate connections to devices in the internal network



Cybersecurity Fundamentals Lab Series

The labs provide learning experiences on cybersecurity topics

Lab 1: Reconnaissance: Scanning with NMAP, Vulnerability Assessment with OpenVAS

Lab 2: Remote Access Trojan (RAT) using Reverse TCP Meterpreter

Lab 3: Escalating Privileges and Installing a Backdoor

Lab 4: Collecting Information with Spyware: Screen Captures and Keyloggers

Lab 5: Social Engineering Attack: Credentials Harvesting and Remote Access through Phishing Emails

Lab 6: SQL Injection Attack on a Web Application

Lab 7: Cross-site Scripting (XSS) Attack on a Web Application

Lab 8: Denial of Service (DoS) Attacks: SYN/FIN/RST Flood, Smurf attack, and SlowLoris

Lab 9: Cryptographic Hashing and Symmetric Encryption

Lab 10: Asymmetric Encryption: RSA, Digital Signatures, Diffie-Hellman

Lab 11: Public Key Infrastructure: Certificate Authority, Digital Certificate

Lab 12: Configuring a Stateful Packet Filter using iptables

Lab 13: Online Dictionary Attack against a Login Webpage

Lab 14: Intrusion Detection and Prevention using Suricata

Lab 15: Packet Sniffing and Relay Attack

Lab 16: DNS Cache Poisoning

Lab 17: Man in the Middle Attack using ARP Spoofing

Lab 18: Understanding Buffer Overflow Attacks in a Vulnerable Application

Lab 19: Conducting Offline Password Attacks

Examples

Vulnerability assessment using OpenVAS

Greenbone Security Assistant

Dashboards Scans Assets Resilience SecInfo

Report: Tue, Nov 29, 2022 3:02 AM UTC Done ID: db2519f

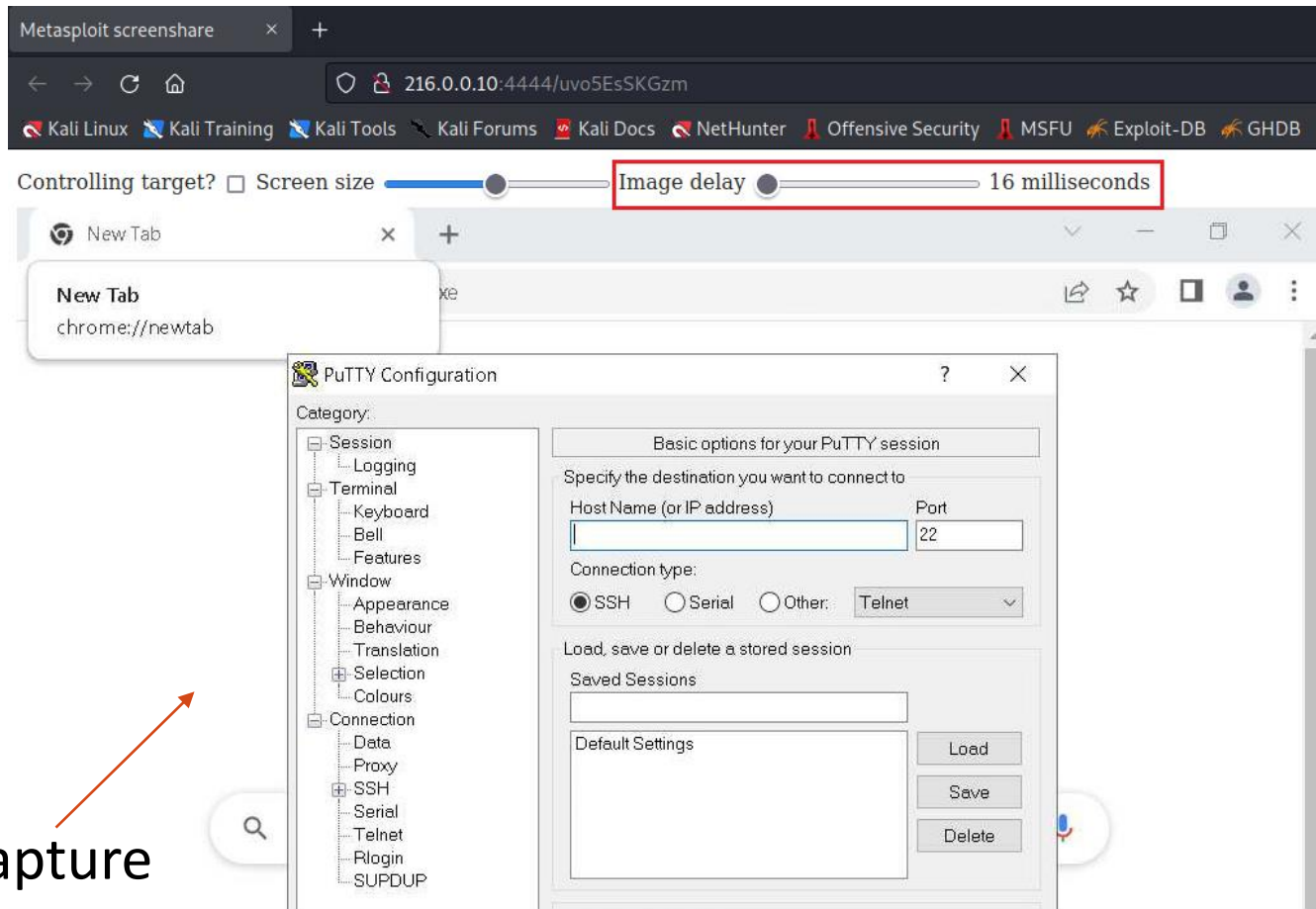
Information **Results (5 of 49)** Hosts (1 of 1) Ports (1 of 1) Applications (2 of 2) Operating Systems (1 of 1) CVEs (0 of 0) Closed CVEs (0 of 0) TLS Certificates (0 of 0) Error Messages (0 of 0) User Tags (0)

Vulnerability	Severity	QoD	Host IP
Operating System (OS) End of Life (EOL) Detection	10.0 (High)	80 %	172.16.0.10
Missing `httpOnly` Cookie Attribute	5.0 (Medium)	80 %	172.16.0.10
Backup File Scanner (HTTP) - Reliable Detection Reporting	5.0 (Medium)	80 %	172.16.0.10
Cleartext Transmission of Sensitive Information via HTTP	4.8 (Medium)	80 %	172.16.0.10
TCP timestamps	2.6 (Low)	80 %	172.16.0.10

(Applied filter: apply_overrides=0 levels=hml rows=100 min_qod=70 first=1 sort-reverse=severity)

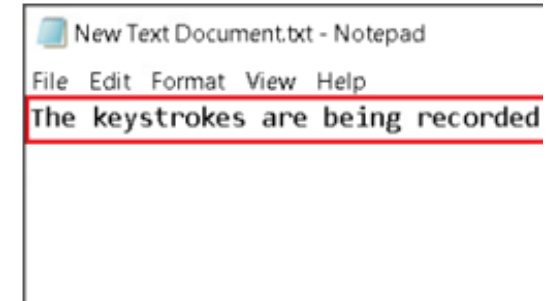
Examples

Deploying a Spyware

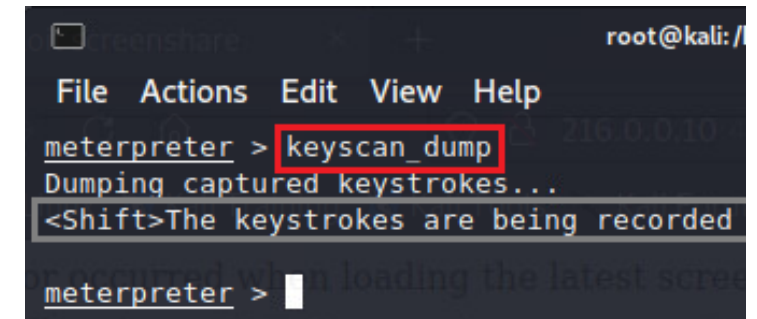


Keylogger

Victim



Attacker



Screen capture

Examples

Social engineering and phishing emails

Victim

Security Notice

From Google Support
to victim@mail-server.lab.I... No date

Dear John,

Someone used your email address to login to your account. We suspect that this activity was performed by a hijacker. Please use the link below to access your Google account settings:

<http://www.google.com/settings>

Regards,
Google Support team.

learner@email.com

.....

Sign in

Need help?

Attacker

```
POSSIBLE USERNAME FIELD FOUND: Email=learner@email.com
POSSIBLE PASSWORD FIELD FOUND: Passwd=password
PARAM: signIn=Sign+in
PARAM: PersistentCookie=yes
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A R

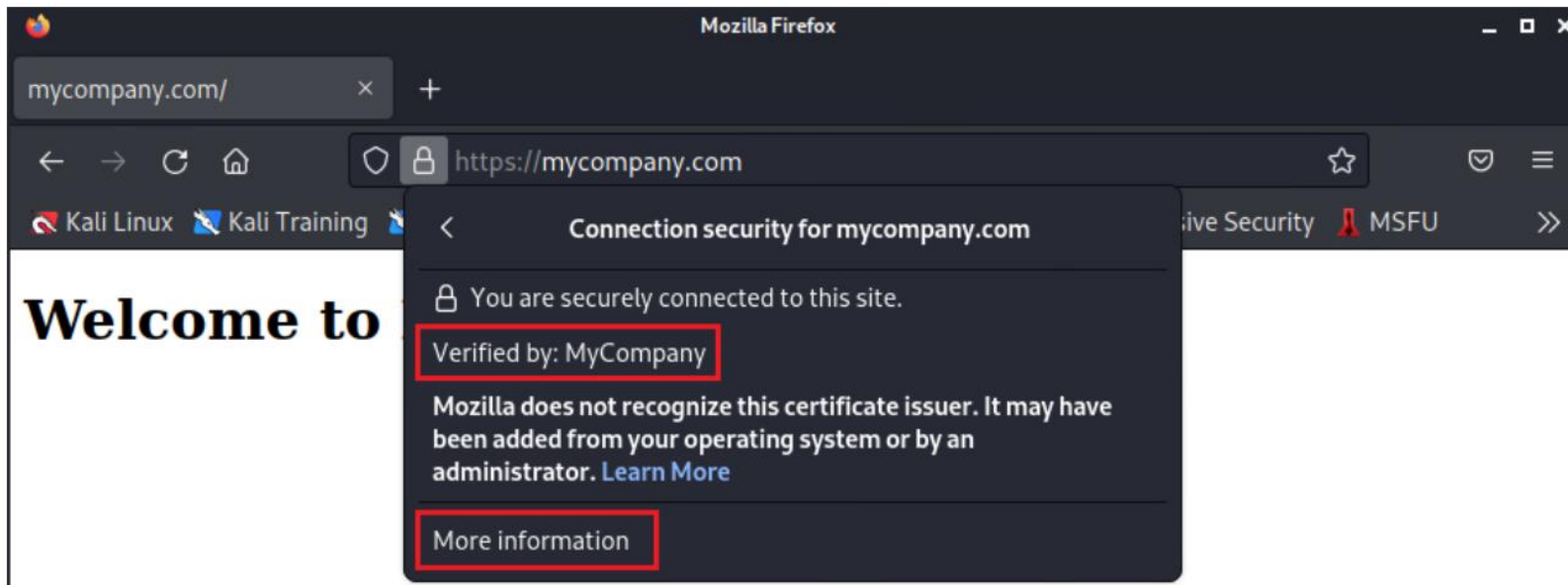
216.0.0.1 - - [08/Sep/2022 19:24:18] "POST /ServiceLogi
216.0.0.1 - - [08/Sep/2022 19:24:18] "GET / HTTP/1.1" 2
216.0.0.1 - - [08/Sep/2022 19:24:38] "GET /favicon.ico
[]
```


Examples

Creating a digital certificate and deploying it on an Apache web server

```
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State] SC
Locality Name (eg, city) [] Columbia
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MyCompany
Organizational Unit Name (eg, section) [] IT
Common Name (e.g. server FQDN or YOUR name) [] mycompany.com
Email Address []:admin@mycompany.com
```

← X.509 certificate



← Certificate deployed on a production grade web server

DEMO

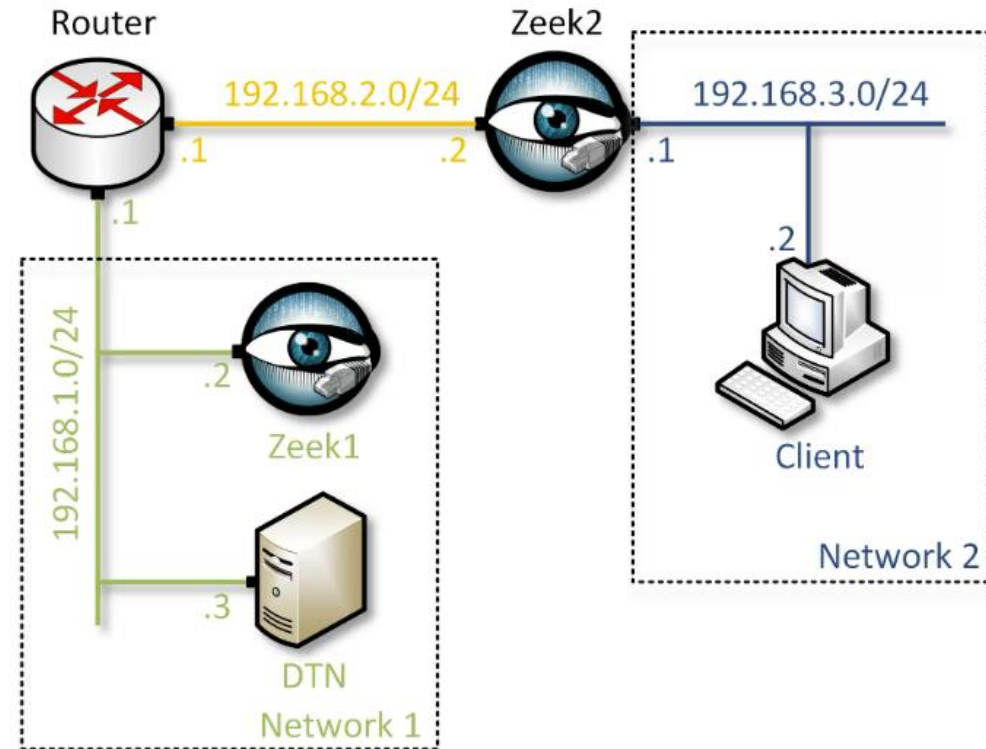
NETLAB+ Platform

Feature	NETLAB+ Platform	Public Cloud
Allocation of resources	Very granular allocation of physical resources	Not granular (access to the physical resources requires additional fees)
Custom pods	Easy to create custom pods	Difficult; hard to design complex topologies
Cost	Cost-effective when used extensively	Cost-effective for individual / small VMs; costly for large VMs over time
Presentation layer for pedagogy	Very flexible. Topology is graphically presented to the learner using a browser	Not flexible; limited to providers' interface, e.g., command-line interface
Time-sharing resource feature	The owner controls who can access resources. Easy to implement sharing policies	Cloud provider controls who can access resources (typically, a fee is required per user)
Integration of physical devices	Easy; physical hardware can be integrated into pods	Difficult; no subscription plan permits integrating customized physical devices
Flexible use of IP addresses and subnets	Each pod runs in an independent controlled environment. Pods have the same topology and IP addresses (w/o conflicts)	IP addresses are typically unique. The vLabs manuals and companion material are not identical, requiring per-learner adjustment
Target	Specially built for pedagogy (education, research, and training)	General, used by a large variety of users
Typical users	From entry-level learners to PhD researchers	More experienced professionals, students

Zeek Intrusion Detection / Prevention System

Zeek - POD

- Network 1 consists of a LAN, a data transfer node (DTN, Linux), and an Zeek-based IDS
- Network 2 consists of a LAN, a client device, and a Zeek-based IDS



Zeek - POD

- Lab experiments

Lab #	Experiment
Lab 1	Introduction to the Capabilities of Zeek
Lab 2	An Overview of Zeek Logs
Lab 3	Parsing, Reading and Organizing Zeek Log Files
Lab 4	Generating, Capturing and Analyzing Network Scanner Traffic
Lab 5	Generating, Capturing and Analyzing DoS and DDoS-centric Network Traffic
Lab 6	Introduction to Zeek Scripting
Lab 7	Introduction to Zeek Signatures
Lab 8	Advanced Zeek Scripting for Anomaly and Malicious Event Detection
Lab 9	Profiling and Performance Metrics of Zeek
Lab 10	Application of the Zeek IDS for Real-Time Network Protection
Lab 11	Preprocessing of Zeek Output Logs for Machine Learning
Lab 12	Developing Machine Learning Classifiers for Anomaly Inference and Classification

Network Tools and Protocols (NTP) Lab Series

NTP Library

- The NTP lab library provides detailed, hands-on experience with tools and protocols
- Labs describe fundamental concepts and tools needed to understand networks and measure their performance
 - Buffer sizing in routers and switches, performance impact
 - TCP congestion control algorithms, performance impact
 - The bandwidth-delay product
 - TCP buffer sizing
 - Measuring fairness between co-existing TCP connections
 - Active queue management of routers and switches

NTP Library

Lab #	Experiment
Lab 1	Introduction to Mininet
Exercise 1	Building a Basic Topology
Lab 2	Introduction to Iperf3
Lab 3	Emulating WAN with NETEM I: Latency, Jitter
Lab 4	Emulating WAN with NETEM II: Packet Loss, Duplication, Reordering, and Corruption
Lab 5	Setting WAN Bandwidth with Token Bucket Filter (TBF)
Exercise 2	Emulating a Wide Area Network (WAN)
Problem 1	Troubleshooting a WAN
Lab 6	Understanding Traditional TCP Congestion Control (HTCP, Cubic, Reno)
Lab 7	Understanding Rate-based TCP Congestion Control (BBR)
Lab 8	Bandwidth-delay Product and TCP Buffer Size
Exercise 3	Tuning TCP and Switch's Buffer Size

NTP Library

Lab #	Experiment
Exercise 4	Running tests with Competing TCP Flows and Different Congestion Control Algorithms
Lab 9	Enhancing TCP Throughput with Parallel Streams
Exercise 5	Enhancing the Aggregate TCP Throughput with Parallel Streams
Problem 2	Enhancing TCP Throughput
Lab 10	Measuring TCP Fairness
Exercise 6	RTT Unfairness
Problem 3	Minimizing the Unfairness
Lab 11	Router's Buffer Size
Lab 12	TCP Rate Control with Pacing
Exercise 7	Setting the Pacing Rate
Lab 13	Impact of MSS on Throughput
Lab 14	Router's Bufferbloat

NTP Library

Lab #	Experiment
Exercise 8	Router's Bufferbloat
Lab 15	Analyzing the Impact of Hardware Offloading on TCP Performance
Lab 16	Random Early Detection
Lab 17	Stochastic Fair Queueing
Lab 18	Controlled Delay (CoDel) Active Queue Management
Lab 19	Proportional Integral Controller-Enhanced (PIE)
Lab 20	Classifying TCP traffic using Hierarchical Token Bucket (HTB)

DEMO

Publications





Computer Communications



Volume 161, 1 September 2020, Pages 212-224




An emulation-based evaluation of TCP BBRv2 Alpha for wired broadband

[Elie F. Kfoury](#)^a , [Jose Gomez](#)^a , [Jorge Crichigno](#)^a  , [Elias Bou-Harb](#)^b 

Show more 

 Add to Mendeley  Share  Cite

<https://doi.org/10.1016/j.comcom.2020.07.018> 

[Get rights and content](#) 

Understanding the Performance of TCP BBRv2 using FABRIC

Jose Gomez*, Elie Kfoury*, Jorge Crichigno*, Gautam Srivastava[†]

*Integrated Information Technology Department, University of South Carolina, Columbia, SC, USA

[†]Department of Mathematics and Computer Science, Brandon University, Canada
{gomezgaj, ekfoury}@email.sc.edu, jrichigno@cec.sc.edu, srivastavag@brandonu.ca

Abstract—This paper presents a performance evaluation of the Bottleneck Bandwidth and Round-trip Time version 2 (BBRv2) TCP congestion control. The experiments are conducted in FABRIC, a national-scale experimental network infrastructure that enables large-scale testing. Google released BBRv2 in 2019 as an improvement over its predecessor, BBRv1. Previous evaluations showed that BBRv2 demonstrates better coexistence with loss-based congestion control algorithms (CCAs), presents low retransmission rates, and produces shorter queueing delays even with large buffers. Evaluations conducted in this paper used FABRIC to reproduce the network conditions observed in Wide Area Networks (WANs). The tests presented in this paper evaluate the throughput as a function of the Round-trip Time (RTT) of BBRv2 compared to various CCAs, the RTT unfairness of BBRv1 and BBRv2, the queue occupancy, and the packet loss rate as a function of the router's buffer size. Additionally, this paper presents and discusses the influence of Active Queue Management (AQM) algorithms to mitigate performance degradation produced by the RTT unfairness and the interaction of different CCAs when sharing a bottleneck link.

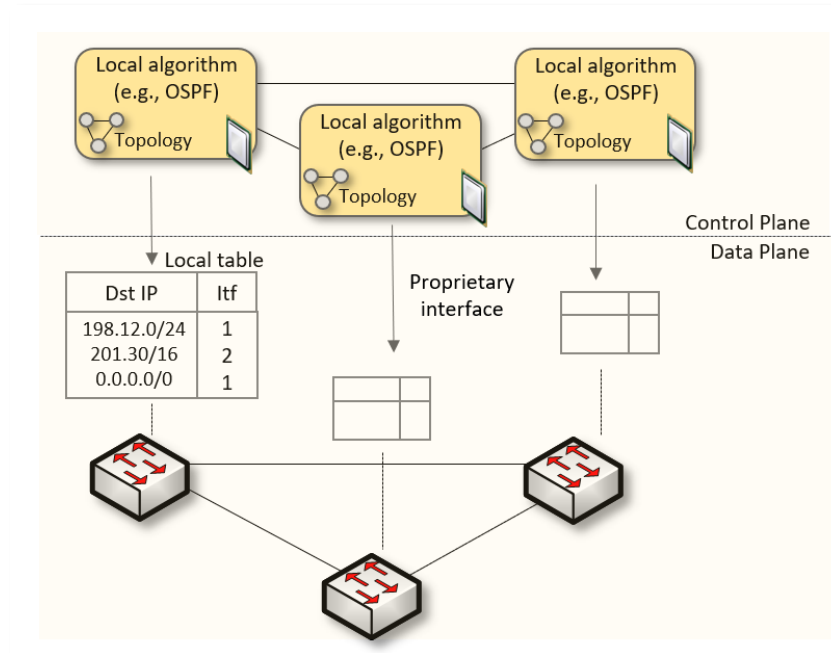
to the shortcomings of BBRv1. The approach of BBRv2 is to create a network traffic model by estimating the bandwidth, observing the RTT, measuring the loss rate, and incorporating Explicit Congestion Notification (ECN) capabilities. The literature [5–7] reported that BBRv2 tolerates higher packet loss rates than loss-based CCAs, presents lower retransmission rates than BBRv1, shows a better coexistence with CUBIC, reduces the impact of the RTT unfairness, and exhibits lower queueing delays.

While previous studies have made significant contributions to understanding the performance of BBRv2 using emulation tools and real hardware, there is a gap in the literature. Specifically, there is a need for a performance evaluation that utilizes large-scale testbeds. This paper aims to fill this gap by presenting an experimental evaluation of BBRv2 using FABRIC [8], a novel research infrastructure that supports large-scale research in various domains such as networking,

Intro to P4 Lab Series

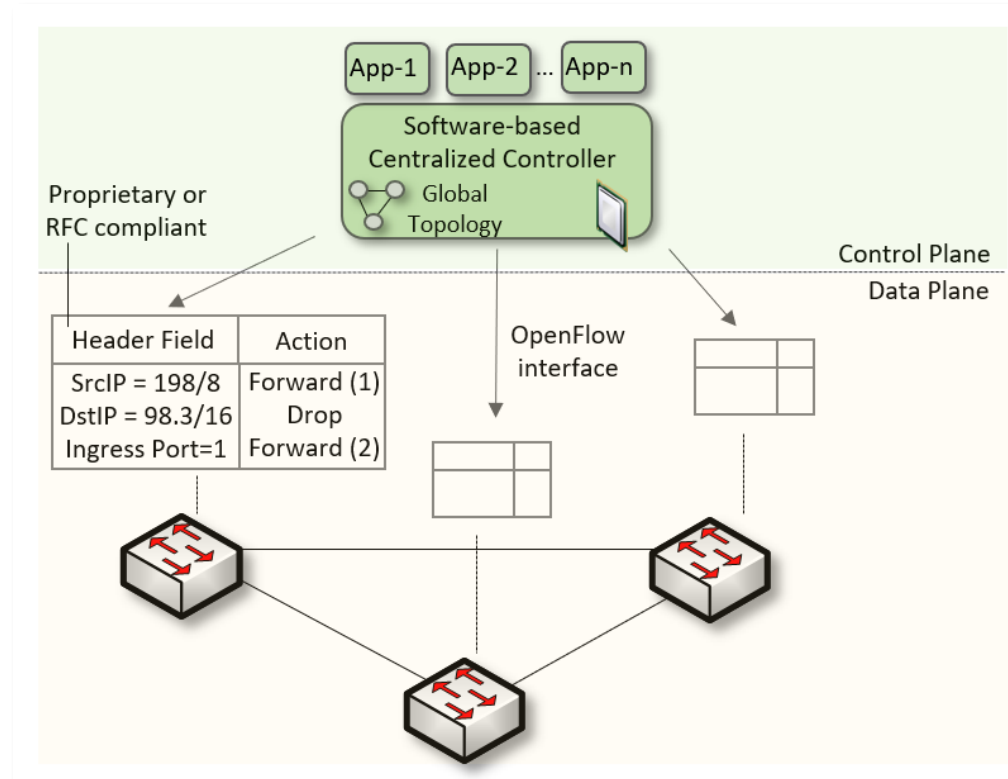
Traditional (Legacy) Networking

- In “traditional” devices, the interface between the control plane and the data plane is proprietary
 - No innovation from network owners
 - A router is a monolithic unit built and internally accessed by the manufacturer only



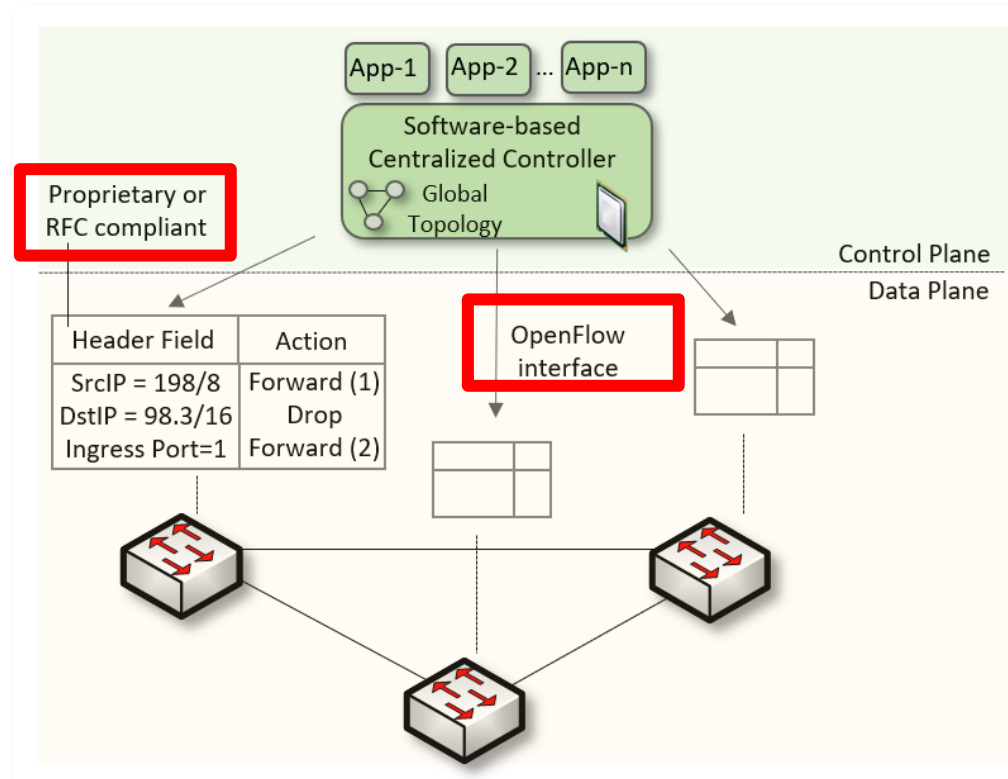
Software-defined Networking

- SDN (1) explicitly separates the control and data planes, and (2) enables the control plane intelligence to be implemented as a software outside the switches
- The function of populating the forwarding table is now performed by the controller



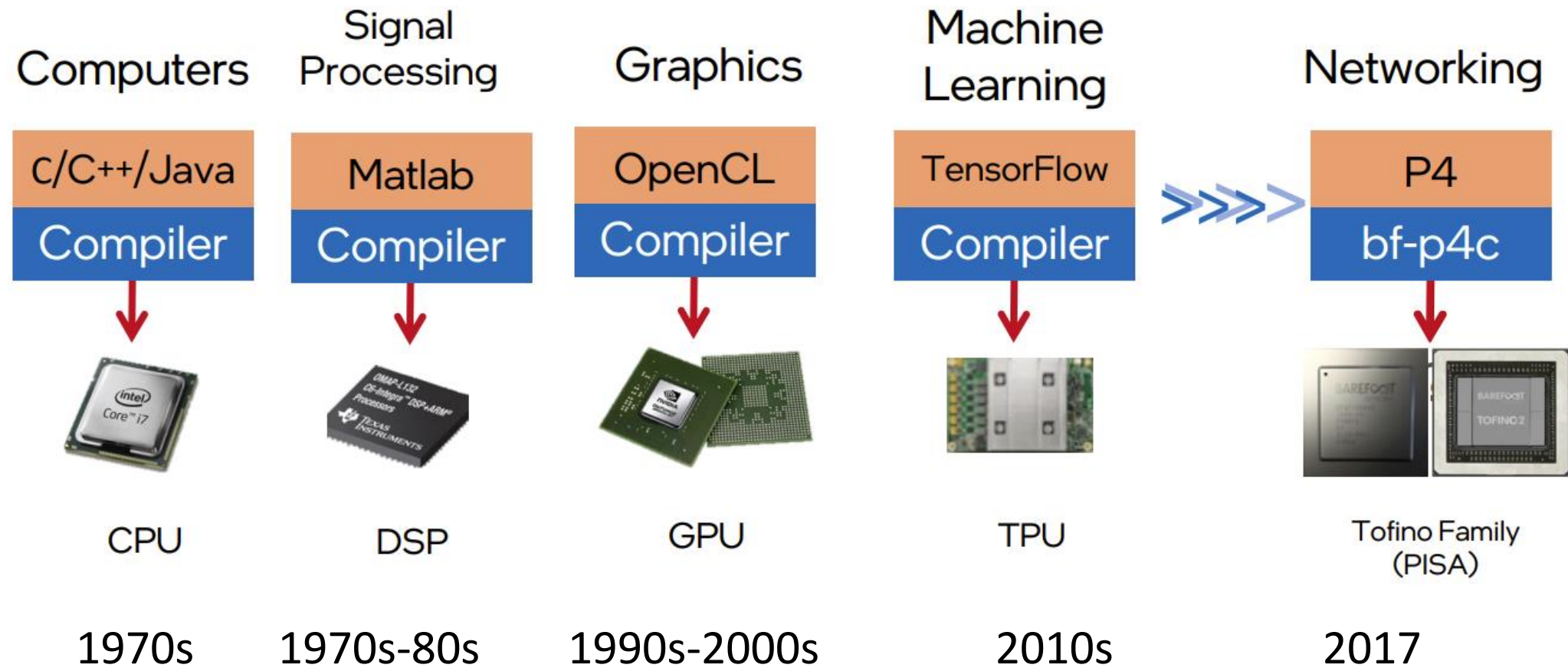
Software-defined Networking

- SDN is limited to the OpenFlow specifications
 - Forwarding rules are based on a fixed number of protocols / header fields (e.g., IP, Ethernet)
- The data plane is designed with fixed functions (hard-coded)
 - Functions are implemented by the chip designer



Can the Data Plane be Programmable?

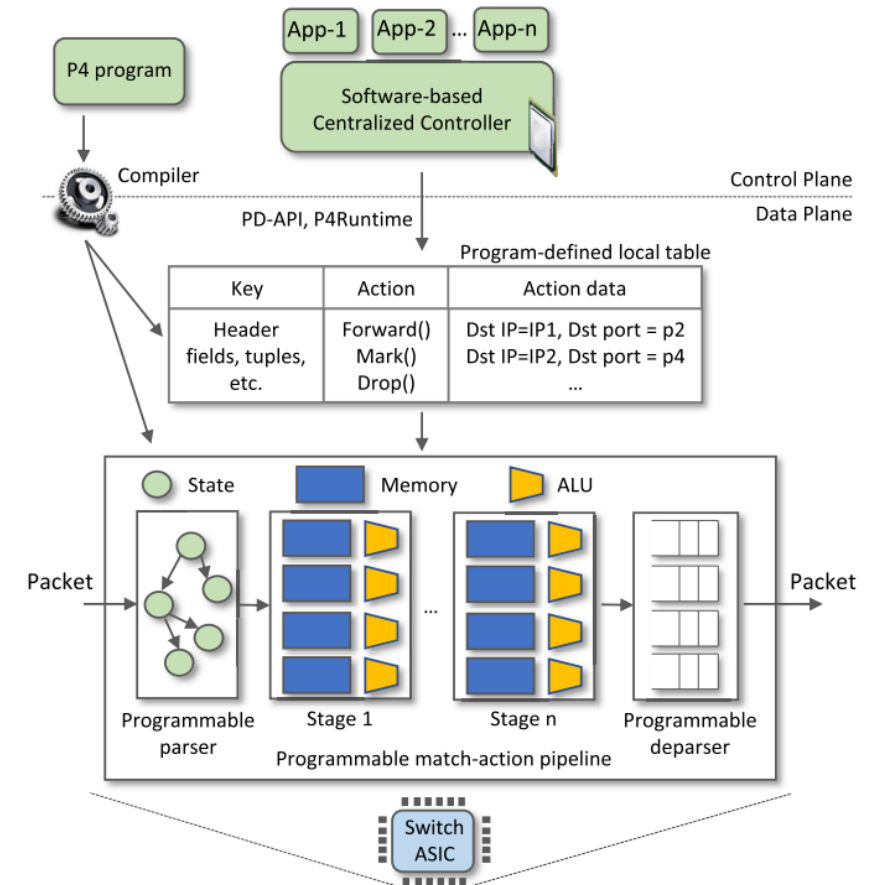
- Evolution of the computing industry



1. Vladimir Gurevich, "Introduction to P4 and Data Plane Programmability," <https://tinyurl.com/2p978tm9>.

P4 Programmable Switches

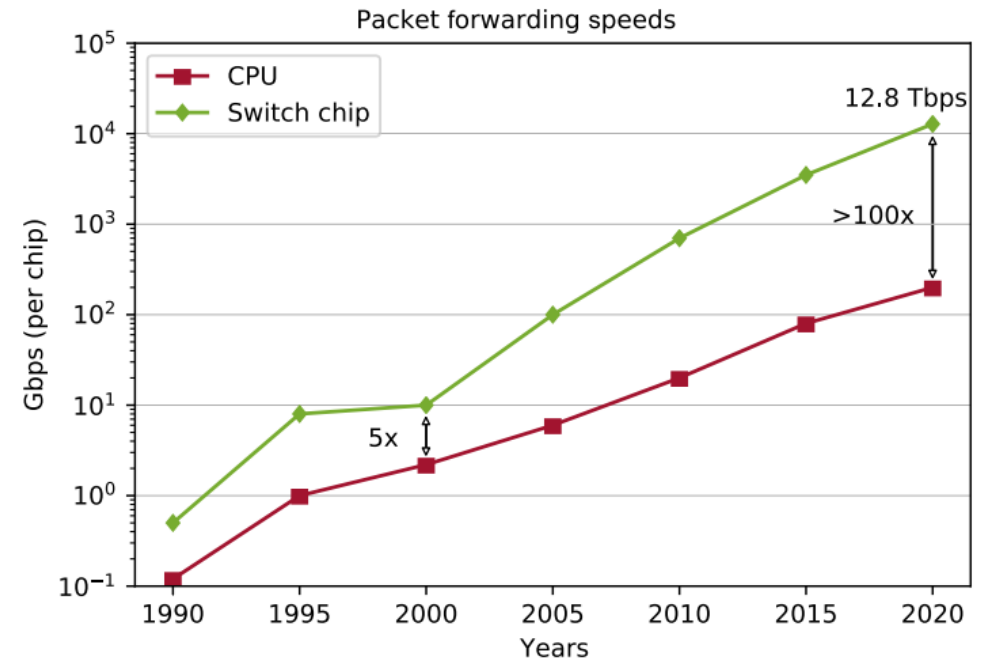
- P4¹ programmable switches permit a programmer to program the data plane
 - Define and parse new protocols
 - Customize packet processing functions
 - Measure events occurring in the data plane with high precision
 - Offload applications to the data plane



1. P4 stands for stands for Programming Protocol-independent Packet Processors

P4 Programmable Switches

- P4¹ programmable switches permit a programmer to program the data plane
 - Define and parse new protocols
 - Customize packet processing functions
 - Measure events occurring in the data plane with high precision
 - Offload applications to the data plane



Reproduced from N. McKeown. Creating an End-to-End Programming Model for Packet Forwarding.
Available: <https://www.youtube.com/watch?v=fiBuao6YZI0&t=4216s>

P4 Programmable Switches

- P4 switches permit programmer to program the data plane
- Add proprietary features; e.g., emulate RTP relay server
- Parse packet headers, including UDP packets carrying RTP traffic
- Header inspection, identifying media sessions using the 5-tuple
- Modify fields, IP addresses and ports

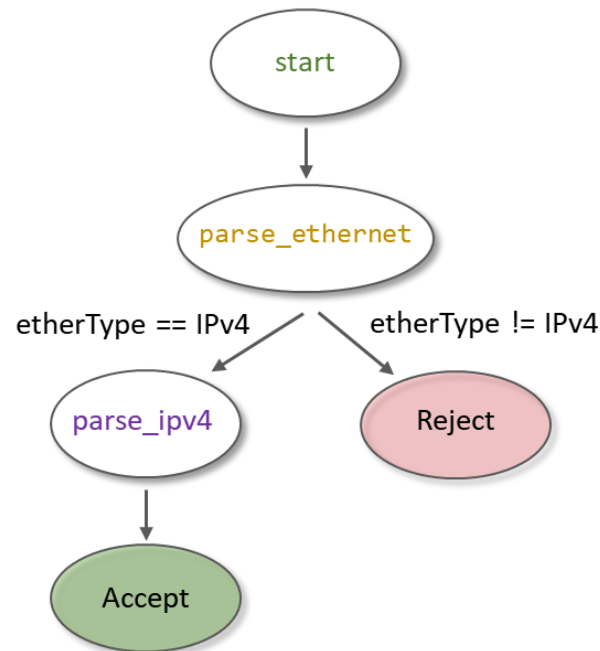
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Version				IHL				DSCP				ECN		Total Length																	
32	Identifier										Flags		Fragment Offset																			
64	Time To Live				Protocol				Header Checksum																							
96	Source IP Address																															
128	Destination IP Address																															
160	Options (if IHL > 5)																															



```
header ipv4_t {
    bit<4> version;
    bit<4> ihl;
    bit<8> diffserv;
    bit<16> totalLen;
    bit<16> identification;
    bit<3> flags;
    bit<13> fragOffset;
    bit<8> ttl;
    bit<8> protocol;
    bit<16> hdrChecksum;
    ip4Addr_t srcAddr;
    ip4Addr_t dstAddr;
}
```


P4 Programmable Switches

- P4 switches permit programmer to program the data plane
- Add proprietary features; e.g., emulate RTP relay server
- Parse packet headers, including UDP packets carrying RTP traffic
- Header inspection, identifying media sessions using the 5-tuple
- Modify fields, IP addresses and ports



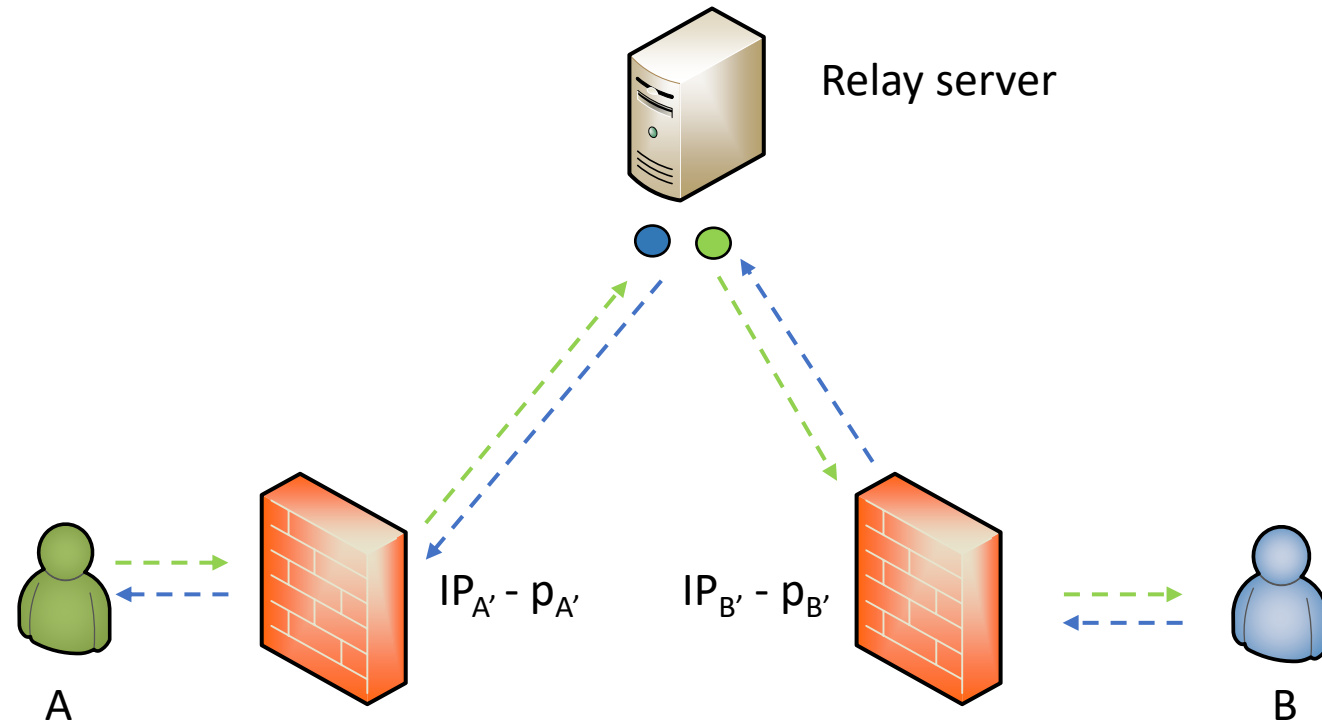
```
state start {
    transition parse_ethernet;
}
state parse_ethernet {
    packet.extract(hdr.ethernet);
    transition select(hdr.ethernet.etherType) {
        TYPE_IPV4: parse_ipv4;
        default: reject;
    }
}
state parse_ipv4 {
    packet.extract(hdr.ipv4);
    transition accept;
}
```

P4 Programmable Switches

- The relay server makes it possible for two devices behind NAT to connect with each other relays the RTP

RTP Information at relay server

	Device IP - port	Allocated IP - port
A	$IP_{A'} - P_{A'}$	$IP_R - P_{RA}$
B	$IP_{B'} - P_{B'}$	$IP_R - P_{RB}$

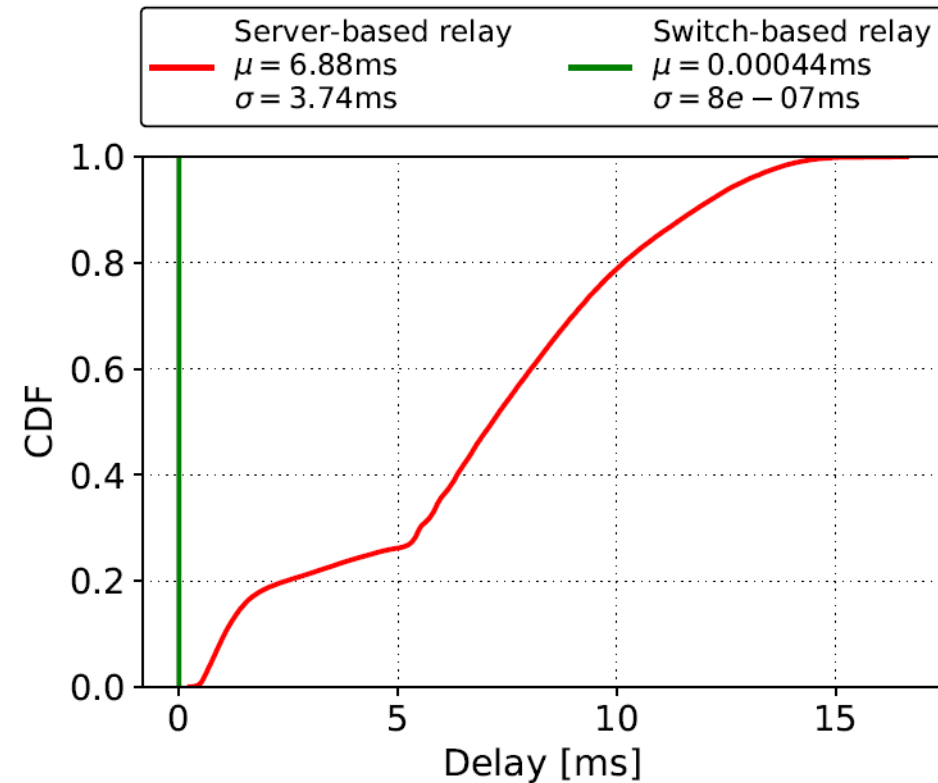


P4 Programmable Switches

- P4 switches permit programmer to program the data plane
- Add proprietary features; e.g., emulate RTP relay server
- Parse packet headers, including UDP packets carrying RTP traffic
- Header inspection, identifying media sessions using the 5-tuple
- Modify fields, IP addresses and ports

Application example: media (voice) relay server

	Programmable Switch	General-purpose CPU
Cost	\$6,000	\$ 10,000 - 25,000
Capacity	~35,000,000 connections per switch	~500 connections per core
Latency	400 nanoseconds	Tens to hundreds of milliseconds



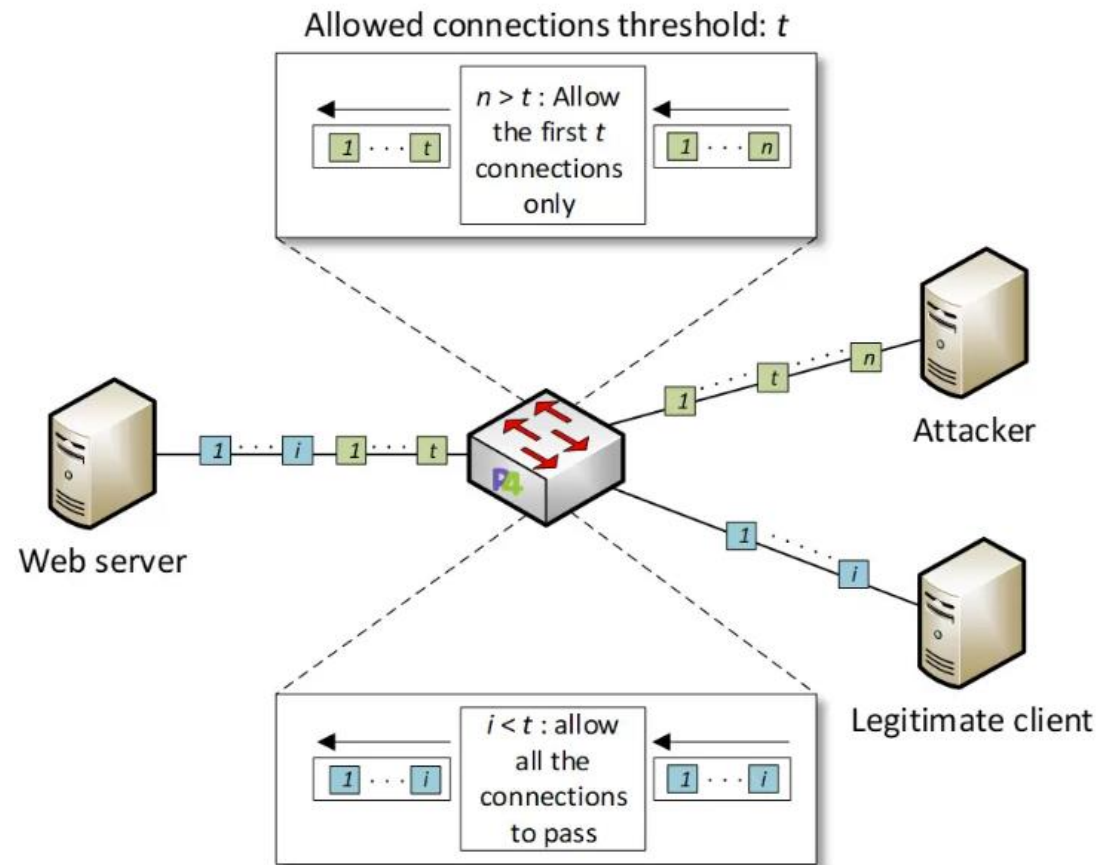
Library on Security Applications with P4

Security Applications with P4

- These labs provide a hands-on experience on implementing cybersecurity applications on P4 programmable data planes
- The lab library explains topics such as implementing stateful packet filters, devising mitigation schemes for TCP SYN flood, DNS amplification, and others
- This library uses the BMv2 software switch (open source)

Security Applications with P4

- Example: DoS detection



Library on Security Applications with P4

Experiments

- Lab 1: Introduction to Mininet
- Lab 2: Introduction to P4 and BMv2
- Lab 3: P4 Program Building Blocks
- Lab 4: Parser Implementation
- Lab 5: Introduction to Match-action Tables
- Lab 6: Implementing a Stateful Packet Filter for the ICMP protocol
- Lab 7: Implementing a Stateful Packet Filter for the TCP protocol
- Lab 8: Detecting and Mitigating the DNS Amplification Attack
- Lab 9: Identifying Heavy Hitters using Count-min Sketches (CMS)
- Lab 10: Limiting the Impact of SYN Flood by Probabilistically Dropping Packets
- Lab 11: Blocking Application Layer Slow DDoS Attack (Slowloris)
- Lab 12: Implementing URL Filtering through Deep Packet Inspection and String Matching

P4 Programmable Data Plane Switches based on Intel's Tofino Chip

P4 PDP Switches based on Intel's Tofino Chip

- These labs provide a hands-on experience on P4 programming running on a production chip
- The lab library describes the architecture of the “Tofino” chip, the software development environment (SDE), and how to use them
- The lab library presents several real examples

```
136 /*****
137 ***** P A R S E R *****/
138 /*****/
139
140 state parse_ethernet {
141     packet.extract(hdr.ethernet);
142     transition select(hdr.ethernet.etherType) {
143         TYPE_IPV4: parse_ipv4;
144         default: accept;
145     }
146 }
147
148 state parse_ipv4 {
149     packet.extract(hdr.ipv4);
150     verify(hdr.ipv4.ihl >= 5, error.IPHeaderTooShort);
151     transition select(hdr.ipv4.ihl) {
152         5 : accept;
153         default : parse_ipv4_option;
154     }
155 }
```

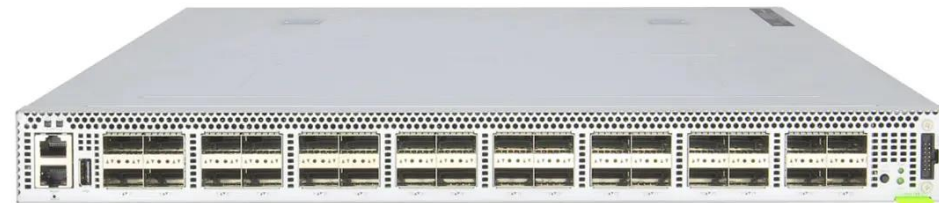
P4 code



Programmable chip

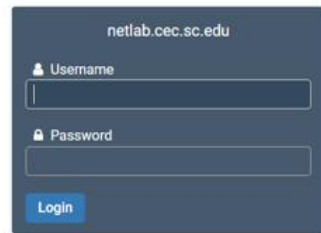
P4 PDP Switches based on Intel's Tofino Chip

- The switch model is Wedge 100BF-32X from Edgecore
- This switch has 32 x 100G QSFP28 switch ports

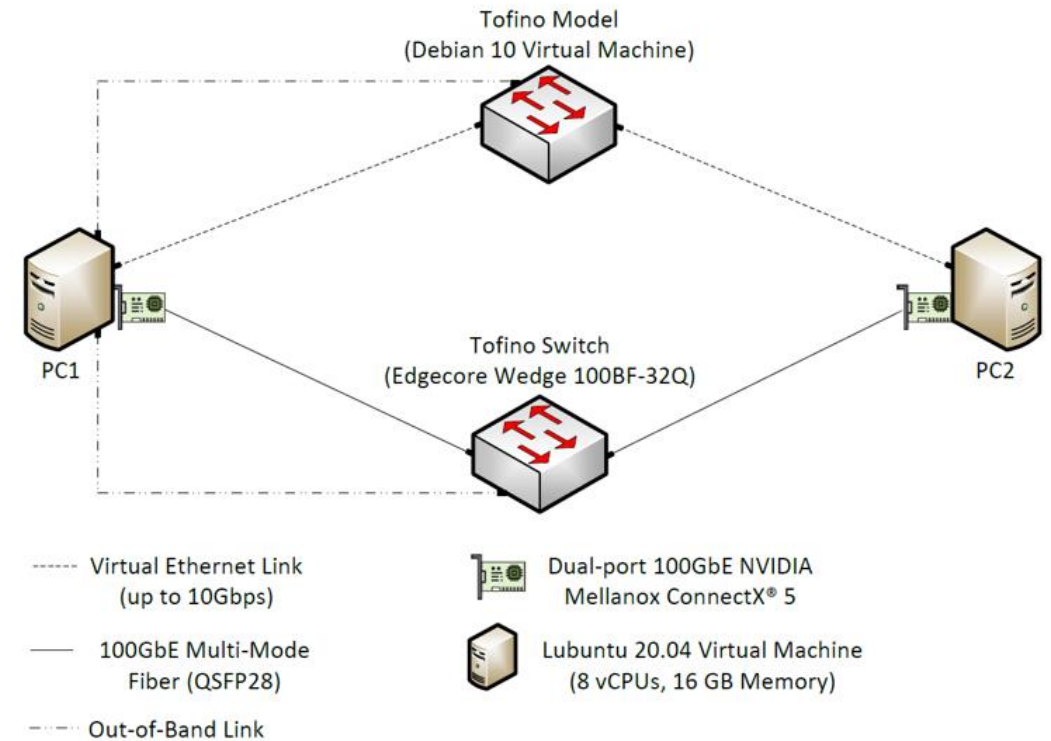


P4 PDP Switches based on Intel's Tofino Chip

- POD design



Cyberinfrastructure
Lab @ UofSC



P4 PDP Switches based on Intel's Tofino Chip

Lab experiments

- Lab 1: Introduction to P4 and BMv2
- Lab 2: P4 Program Building Blocks
- Lab 3: Parser Implementation
- Lab 4: Introduction to Match-action Tables (Part 1)
- Lab 5: Introduction to Match-action Tables (Part 2)
- Lab 6: Populating and Managing Match-action Tables
- Lab 7: Checksum Recalculation and Packet Deparsing

Exercises

- Exercise 1: Compiling and Testing a P4 Program
- Exercise 2: Parsing UDP and RTP
- Exercise 3: Building a Simplified NAT
- Exercise 4: Configuring Tables at Runtime
- Exercise 5: Building a Packet Reflector

DEMO 1

DEMO 2

P4 PDP Switches based on Intel's Tofino Chip

Lab experiments

- Lab 1: Introduction to P4 and BMv2
- Lab 2: P4 Program Building Blocks
- Lab 3: Parser Implementation
- Lab 4: Introduction to Match-action Tables (Part 1)
- Lab 5: Introduction to Match-action Tables (Part 2)
- Lab 6: Populating and Managing Match-action Tables
- Lab 7: Checksum Recalculation and Packet Deparsing

Exercises

- Exercise 1: Compiling and Testing a P4 Program
- Exercise 2: Parsing UDP and RTP
- Exercise 3: Building a Simplified NAT
- Exercise 4: Configuring Tables at Runtime
- Exercise 5: Building a Packet Reflector

Other P4 Libraries

Introduction to P4 Lab Series

Lab Experiments

- Lab 1: Introduction to Mininet
- Lab 2: Introduction to P4 and BMv2
- Lab 3: P4 Program Building Blocks
- Lab 4: Parser Implementation
- Lab 5: Introduction to Match-action Tables (Part 1)
- Lab 6: Introduction to Match-action Tables (Part 2)
- Lab 7: Populating and Managing Match-action Tables
- Lab 8: Checksum Recalculation and Packet Deparsing

Lab Exercises

- Exercise 1: Building a Basic Topology
- Exercise 2: Compiling and Testing a P4 Program
- Exercise 3: Parsing UDP and RTP
- Exercise 4: Building a Simplified NAT
- Exercise 5: Configuring Tables at Runtime
- Exercise 6: Building a Packet Reflector

Publications

SPRINGER LINK

Find a journal

Publish with us

Track your research

Search



European Symposium on Research in Computer Security

↳ ESORICS 2022: [Computer Security – ESORICS 2022](#) pp 551–569 | [Cite as](#)

[Home](#) > [Computer Security – ESORICS 2022](#) > Conference paper

INC: In-Network Classification of Botnet Propagation at Line Rate

[Kurt Friday](#) , [Elie Kfoury](#), [Elias Bou-Harb](#)  & [Jorge Crichigno](#)

Conference paper | [First Online: 25 September 2022](#)

Offloading Media Traffic to Programmable Data Plane Switches

Elie F. Kfoury*, Jorge Crichigno*, Elias Bou-Harb[†], Vladimir Gurevich[‡]

*Integrated Information Technology, University of South Carolina, USA

[†]The Cyber Center For Security and Analytics, University of Texas at San Antonio, USA

[‡]Barefoot Networks, an Intel Company, USA

Abstract—According to estimations, approximately 80% of Internet traffic represents media traffic. Much of it is generated by end users communicating with each other (e.g., voice, video sessions). A key element that permits the communication of users that may be behind Network Address Translation (NAT) is the relay server.

This paper presents a scheme for offloading media traffic from relay servers to programmable switches. The proposed scheme relies on the capability of a P4 switch with a customized parser to de-encapsulate and process packets carrying media traffic. The switch then applies multiple switch actions over the packets. As these actions are simple and collectively emulate a relay server, the scheme is capable of moving relay functionality to the data plane operating at terabits per second. Performance evaluations show that the proposed scheme not only produces optimal results regarding Quality of Service (QoS) parameters (no packet loss, minimum delay, negligible delay variation, high Mean Opinion Score) but also scales much better than current solutions. Evaluations conducted with up to 35Gbps of media traffic or its equivalent of 400,000 simultaneous G.711 media sessions (limited only by the traffic generator rather than by the switch) show an ideal operation of the switch-based solution (using ~1% of the switching capacity). In contrast, a relay server with a modern CPU model used for evaluations can process up

results [8] reveal that CGN has a widespread adoption and that over half of operators have deployed or will deploy CGN. NAT introduces issues such as violation of the end-to-end principle, scalability and reliability concerns, and traversal of end-to-end sessions. The latter is a problem that severely affects media traffic. For example, for an end user to be reachable for an end-to-end media session (voice, video), the user must wait and accept incoming connections at a well-known port. With NAT, the user is not reachable because it is assigned a private IP address. Furthermore, port numbers are also allocated dynamically. Moreover, these dynamic allocations interfere with the operation of signaling protocols, as end devices rely on opening ephemeral ports during the session establishment to send and receive media traffic [6].

Technical solutions to these problems include Session Traversal Utilities for NAT (STUN) [9], Traversal Using Relays around NAT (TURN) [10], Interactive Connectivity Establishment (ICE) [11], Port Mapping Protocol (PMP) [12], and Port Control Protocol (PCP) [13]. Essentially, the general solution of the NAT traversal problem requires the use of a

Summary

- Several lab libraries for networks and cybersecurity
- Lab libraries cover topics from introductory to advanced levels
- Opportunity for students to go beyond the basics
 - Advanced courses
 - Capstone projects
 - Undergraduate research
- The material is available through NETLAB+, NETLAB Online, and FABRIC