# An Overview of P4 Programmable Switches and Applications to Cybersecurity and Networks

Jorge Crichigno
College of Engineering and Computing, University of South Carolina
jcrichigno@cec.sc.edu
http://ce.sc.edu/cyberinfra

IEEE International Conference on Telecommunications and Signal Processing (TSP)

Online
July 12-14, 2023

region8 IEEE
IEEE
IEEE Czechoslovakia Section
hte
twitter
Facebook /tspconf
46th International Conference on
Telecommunications and Signal Processing
(TSP)
IEEE Conference Record: #59544
tsp
2023
July 12-14, 2023
Virtual Conference
IEEE Xplore Digital Library
THOMSON REUTERS
Google
SciVerse
Scopus
dblp
“AI for Bright Future of Humanity”

UNIVERSITY OF SOUTH CAROLINA

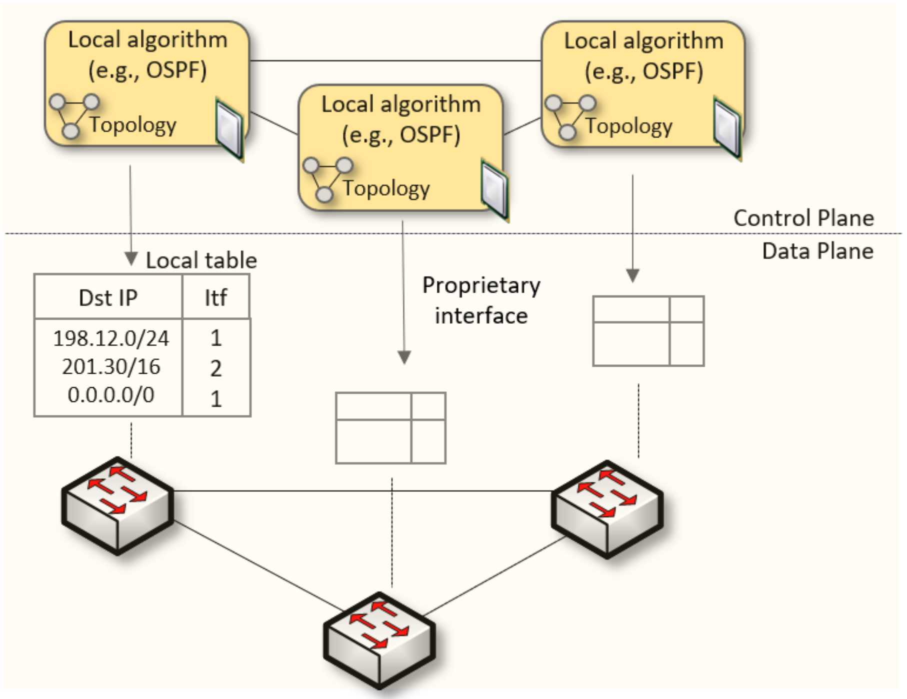
# Agenda

- Introduction to P4 Programmable Switches
  - ➢ Legacy devices and Software-defined Networking (SDN)
  - ➢ Programmable data plane (PDP) switches
- Dynamic Router’s Buffer Sizing using P4 Switches
  - ➢ Buffer sizing problem
  - ➢ A passive application using P4 switches
- DGA Family Classification using DNS Deep Packet Inspection on P4 Switches
  - ➢ Domain Generation Algorithms (DGA) used by malware’s command and control (C2)
  - ➢ An application for detection and classification of DGAs using P4 switches
- Conclusion

# Introduction to P4 Programmable Switches

# Traditional (Legacy) Networking

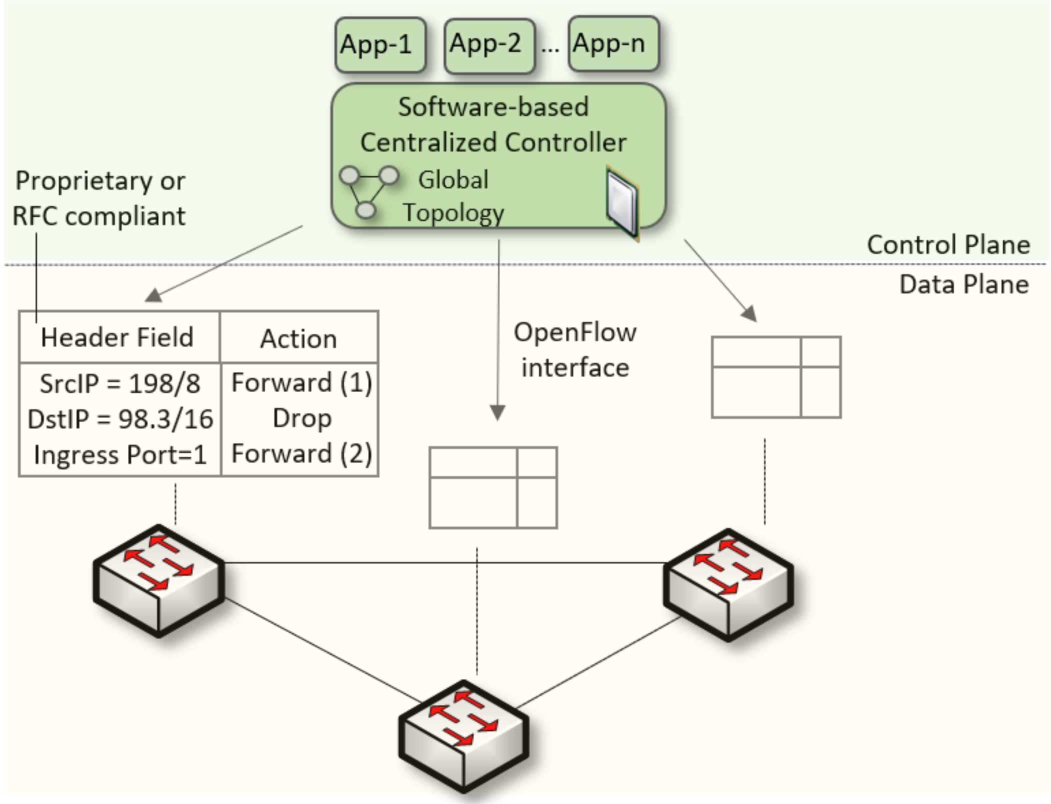
- Since the explosive growth of the Internet in the 1990s, the networking industry has been dominated by closed and proprietary hardware and software
- The interface between control and data planes has been historically proprietary
  - ➢ Vendor dependence: slow product cycles of vendor equipment, no innovation from **end programmers**
  - ➢ A router is a monolithic unit built and internally accessed by the manufacturer only


Local algorithm (e.g., OSPF)
Topology
Local algorithm (e.g., OSPF)
Topology
Local algorithm (e.g., OSPF)
Topology
Control Plane
Data Plane
Local table
Proprietary interface

| Dst IP | Itf |
|---|---|
| 198.12.0/24 | 1 |
| 201.30/16 | 2 |
| 0.0.0.0/0 | 1 |

# Software-Defined Networking (SDN)

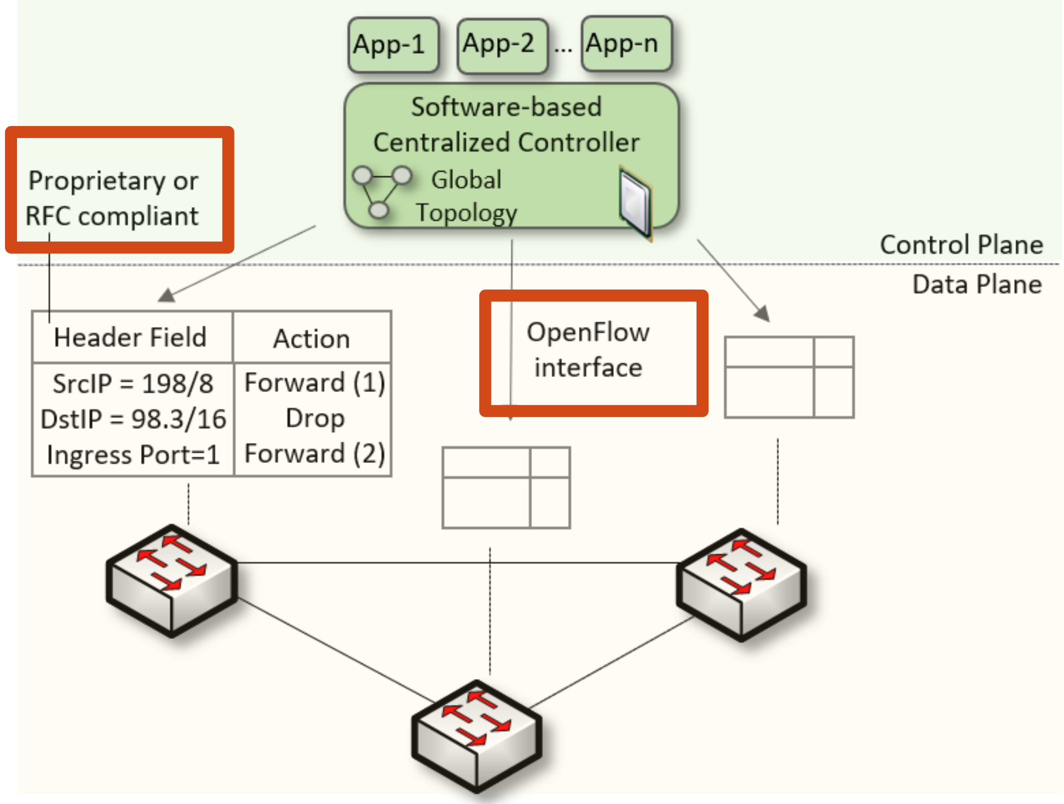
- Protocol ossification has been challenged first by SDN
- SDN (1) explicitly separates the control and data planes, and (2) enables the control plane intelligence to be implemented as a software outside the switches by **end programmers**
- The function of populating the forwarding table is now performed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



| Header Field | Action |
|---|---|
| SrcIP = 198/8 | Forward (1) |
| DstIP = 98.3/16 | Drop |
| Ingress Port=1 | Forward (2) |

# SDN Limitation

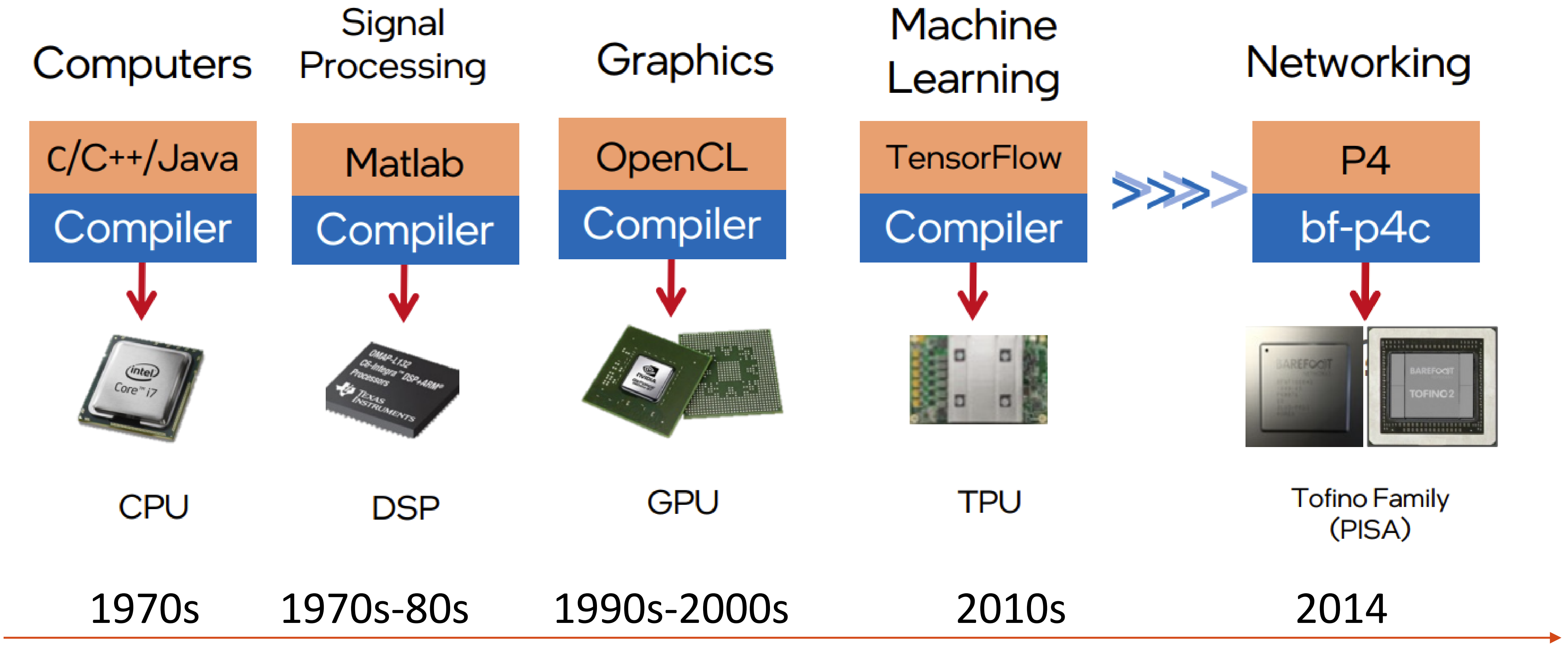
- SDN is limited to the OpenFlow specifications
  - ➢ Forwarding rules are based on a fixed number of protocols / header fields (e.g., IP, Ethernet)
- The data plane is designed with fixed functions (hard-coded)
  - ➢ Functions are implemented by the chip designer



| Header Field | Action |
|---|---|
| SrcIP = 198/8 | Forward (1) |
| DstIP = 98.3/16 | Drop |
| Ingress Port=1 | Forward (2) |

# Can the Data Plane be Programmable?

- Evolution of the computing industry



Computers
C/C++/Java
Compiler
CPU
1970s

Signal Processing
Matlab
Compiler
DSP
1970s-80s

Graphics
OpenCL
Compiler
GPU
1990s-2000s

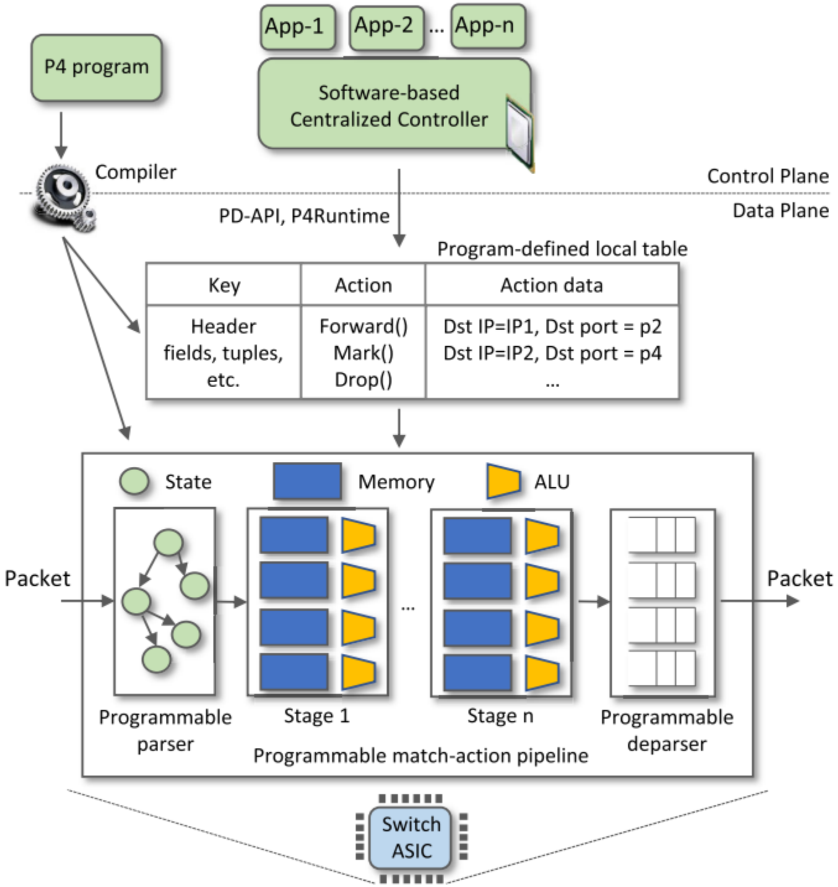
Machine Learning
TensorFlow
Compiler
TPU
2010s

Networking
P4
bf-p4c
Tofino Family (PISA)
2014

1. Vladimir Gurevich, “Introduction to P4 and Data Plane Programmability,” https://tinyurl.com/2p978tm9.

# P4 Programmable Switches

- P4[1] programmable switches permit **end programmers** 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

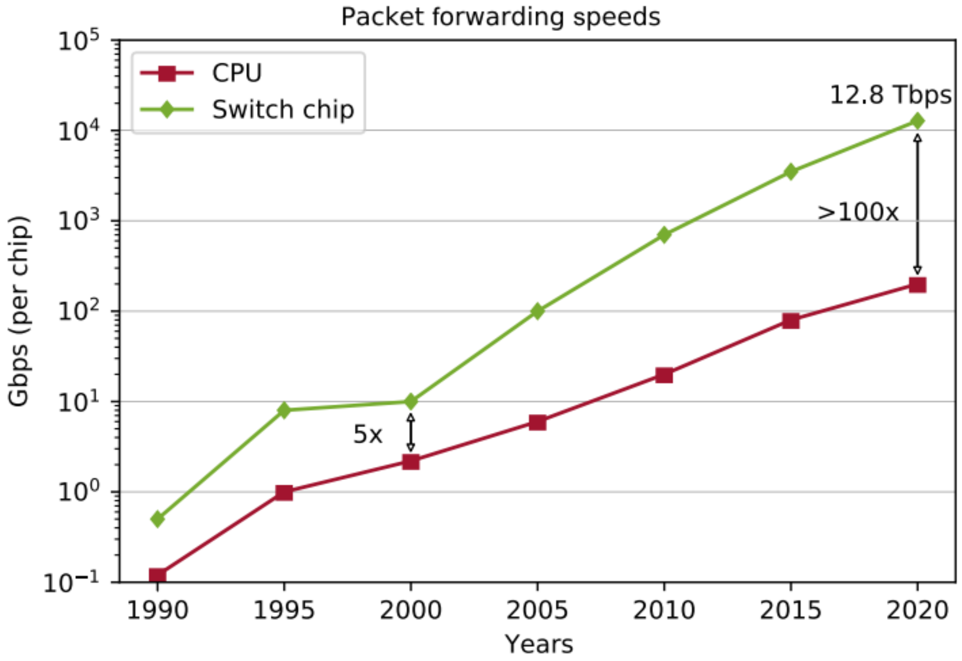


| Key | Action | Action data |
|---|---|---|
| Header fields, tuples, etc. | Forward()<br>Mark()<br>Drop() | Dst IP=IP1, Dst port = p2<br>Dst IP=IP2, Dst port = p4<br>... |

[1] P4 stands for stands for Programming Protocol-independent Packet Processors

# P4 Programmable Switches

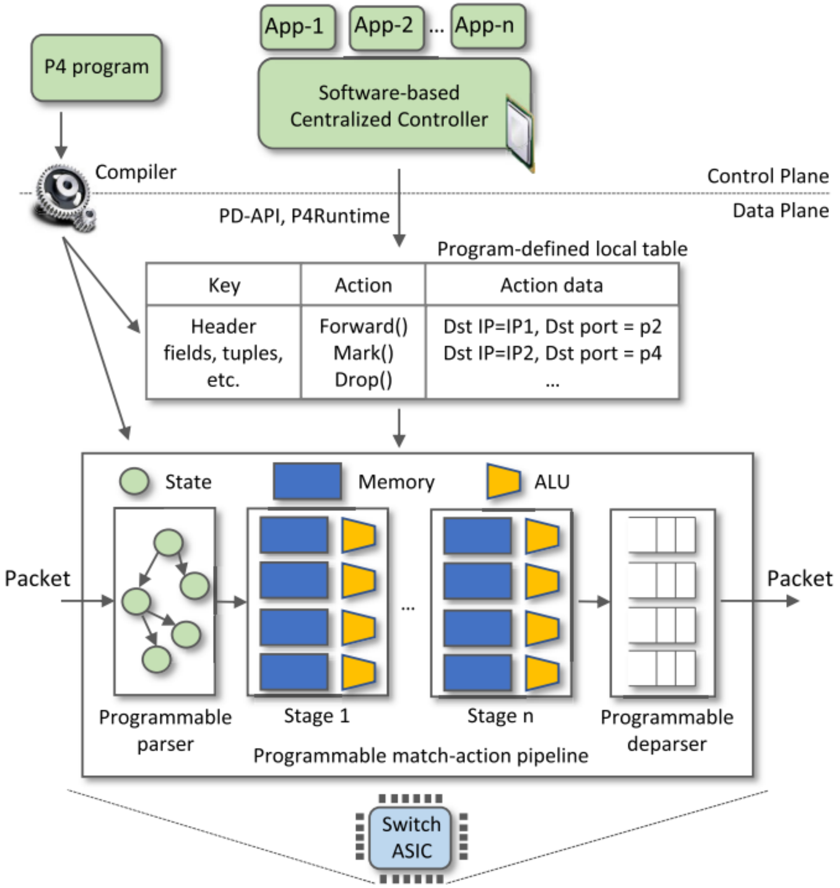
- P4[1] programmable switches permit **end programmers** to program the data plane
  - Define and parse new protocols
  - Customize packet processing functions
  - Measure events occurring in the data plane with high precision
  - Offload applications to the data plane



Reproduced from N. McKeown. Creating an End-to-End Programming Model for Packet Forwarding. Available: https://www.youtube.com/watch?v=fiBuao6YZI0&t=631s

# Generalized Forwarding: Match + Action

- Each switch contains table/s
  - ➢ Match bits in arriving packet (match phase)
  - ➢ Take action - Many header fields can determine the action  (action phase)
    - Drop
    - Copy
    - Modify
    - Forward (destination-based forwarding is just a particular case)
    - …

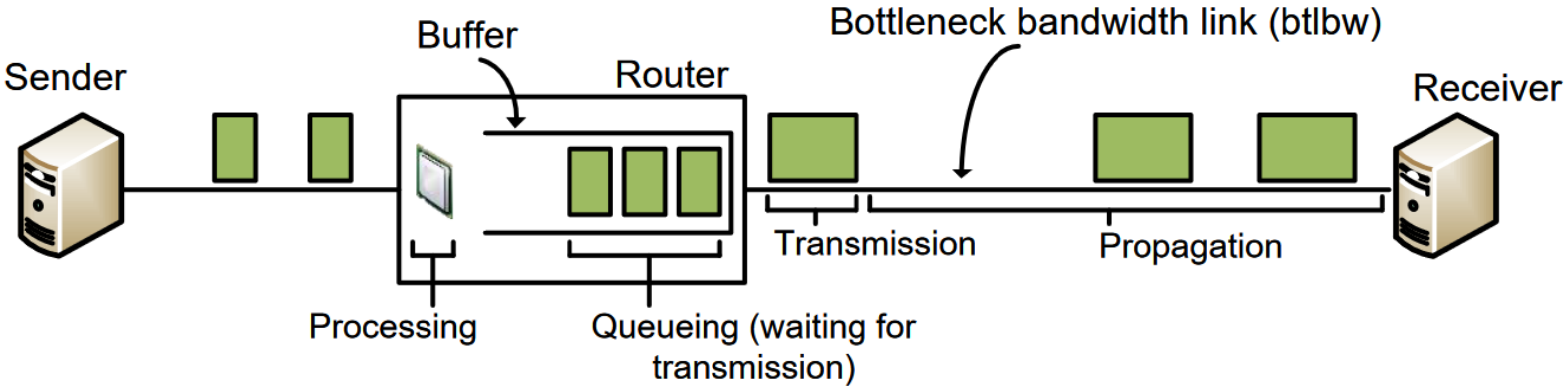


| Key | Action | Action data |
|---|---|---|
| Header fields, tuples, etc. | Forward()<br>Mark()<br>Drop() | Dst IP=IP1, Dst port = p2<br>Dst IP=IP2, Dst port = p4<br>… |

# Dynamic Router’s Buffer Sizing using Passive Measurements and P4 Programmable Switches

# Buffer Sizing Problem

- Routers and switches are designed to include packet buffers
- The size of buffers imposes significant implications on the performance of the network
- If the buffer allocated to an interface is
  - Very large, then packets may experience excessive delay (“bufferbloat”)
  - Very small, then there may be a large packet drop rate and low link utilization

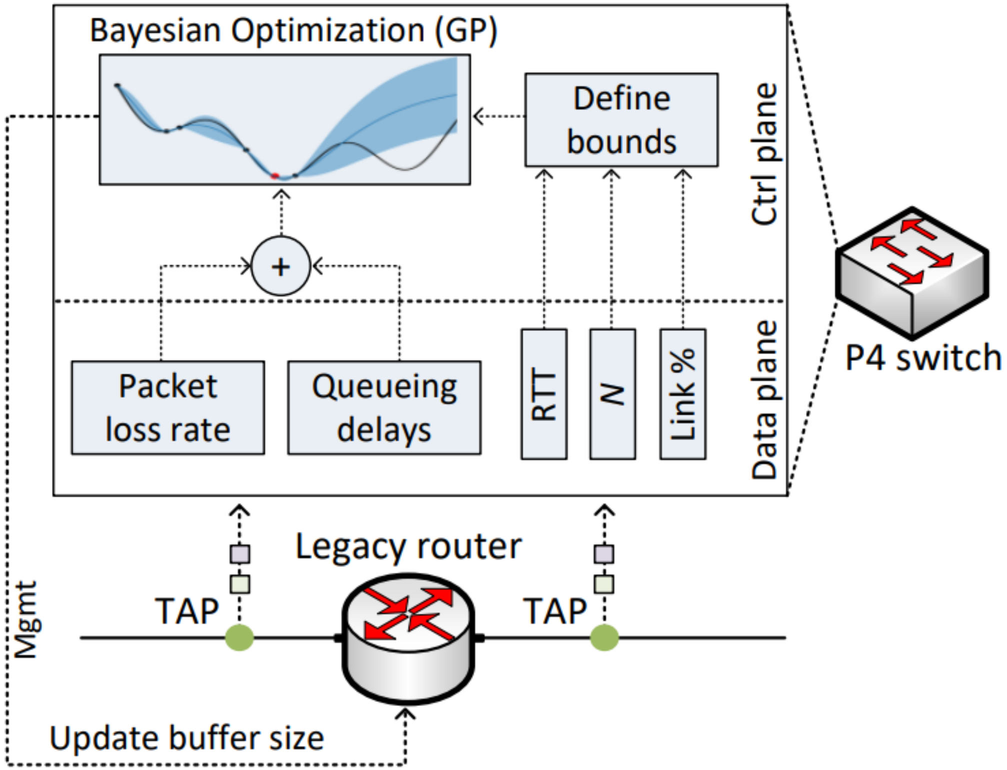

Buffer
Sender
Router
Bottleneck bandwidth link (btlbw)
Receiver
Transmission
Propagation
Processing
Queueing (waiting for transmission)

# Buffer Sizing Problem

- General rule-of-thumb in the 90s was that the buffer size must equal the Bandwidth-delay product (BDP)
  - ➢ Buffer = C * RTT
  - ➢ C is the capacity of the port and RTT is the average round-trip time (RTT)
- The “Stanford rule” corrected the previous rule
  - ➢ Buffer = $(C * RTT)$/$(\sqrt{N})$
  - ➢ N is the number of long (persistent over time) flows traversing the port
- Operator hardcodes the buffer size based on the typical traffic pattern

# Proposed System

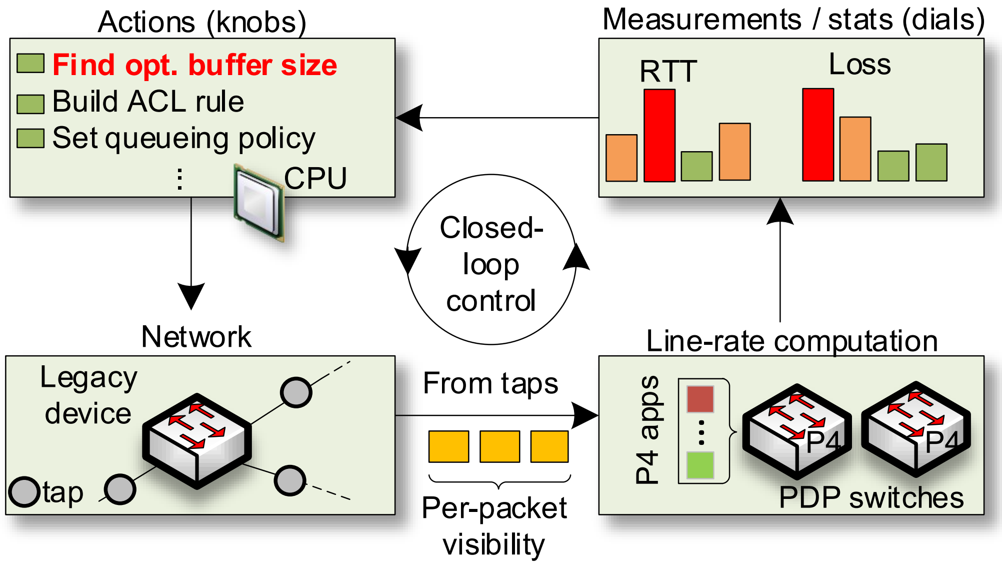
- The buffer size is dynamically modified
- A P4 switch is deployed passively to compute:
  - ➢ Number of long flows
  - ➢ Average RTT
  - ➢ Queueing delays
  - ➢ Packet loss rates
- The control plane sequentially searches for a buffer that minimizes delays and losses
- The searching algorithm is Bayesian Optimization (BO) with Gaussian Processes[1]



[1] E. Kfoury, J. Crichigno, E. Bou-Harb, "P4Tune: Enabling Programmability in Non-Programmable Networks," IEEE Communications Magazine, Vol. 61, Issue 3, Jun. 2023

# Proposed System

- The buffer size is dynamically modified
- A P4 switch is deployed passively to compute:
  - Number of long flows
  - Average RTT
  - Queueing delays
  - Packet loss rates
- The control plane sequentially searches for a buffer that minimizes delays and losses
- The searching algorithm is Bayesian Optimization (BO) with Gaussian Processes

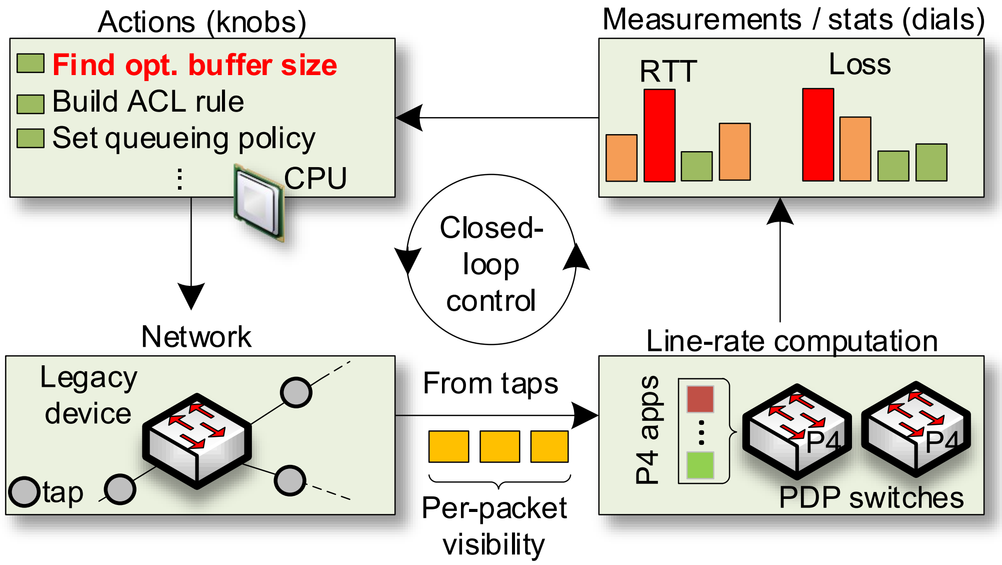


Closed-loop control system

[1] E. Kfoury, J. Crichigno, E. Bou-Harb, “P4Tune: Enabling Programmability in Non-Programmable Networks,” IEEE Communications Magazine, Vol. 61, Issue 3, Jun. 2023

# Proposed System

- Note that the system incorporates
  - ➢ Customized packet processing
  - ➢ Nanosecond resolution measurements
  - ➢ Per-packet visibility
- The P4 apps run on the PDP chip at line rate

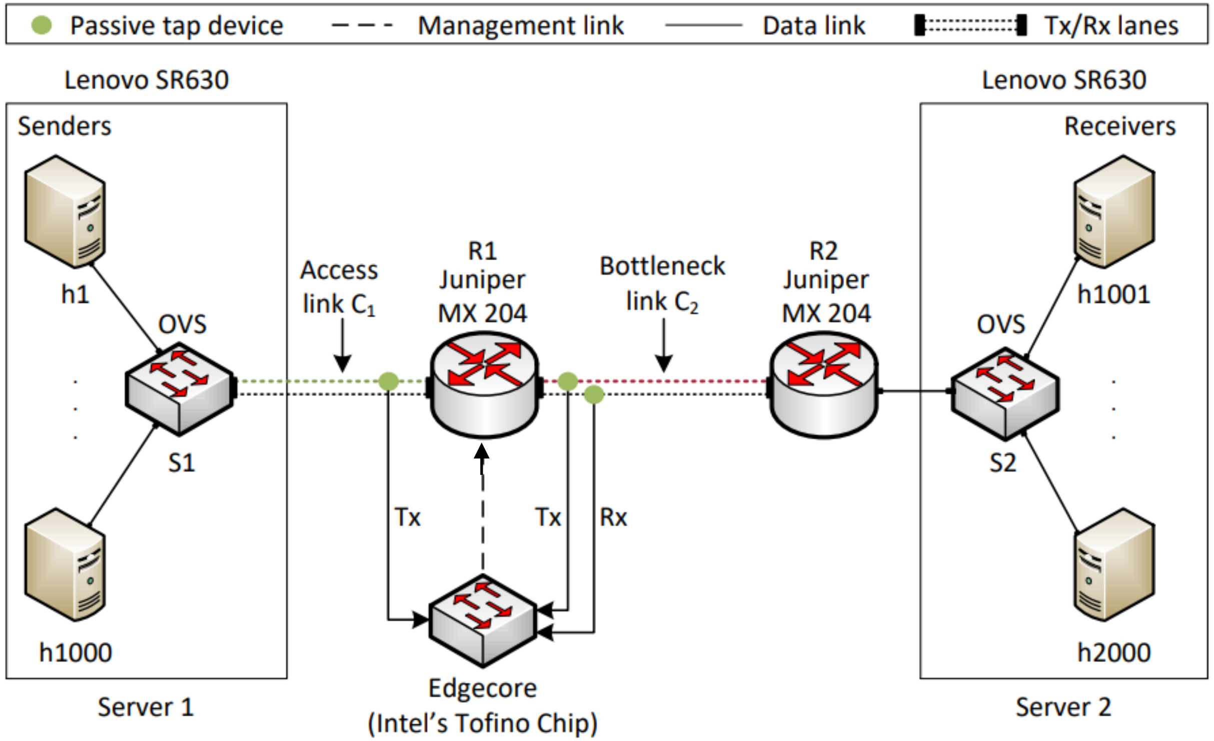


Closed-loop control system

[1] E. Kfoury, J. Crichigno, E. Bou-Harb, “P4Tune: Enabling Programmability in Non-Programmable Networks,” IEEE Communications Magazine, Vol. 61, Issue 3, Jun. 2023

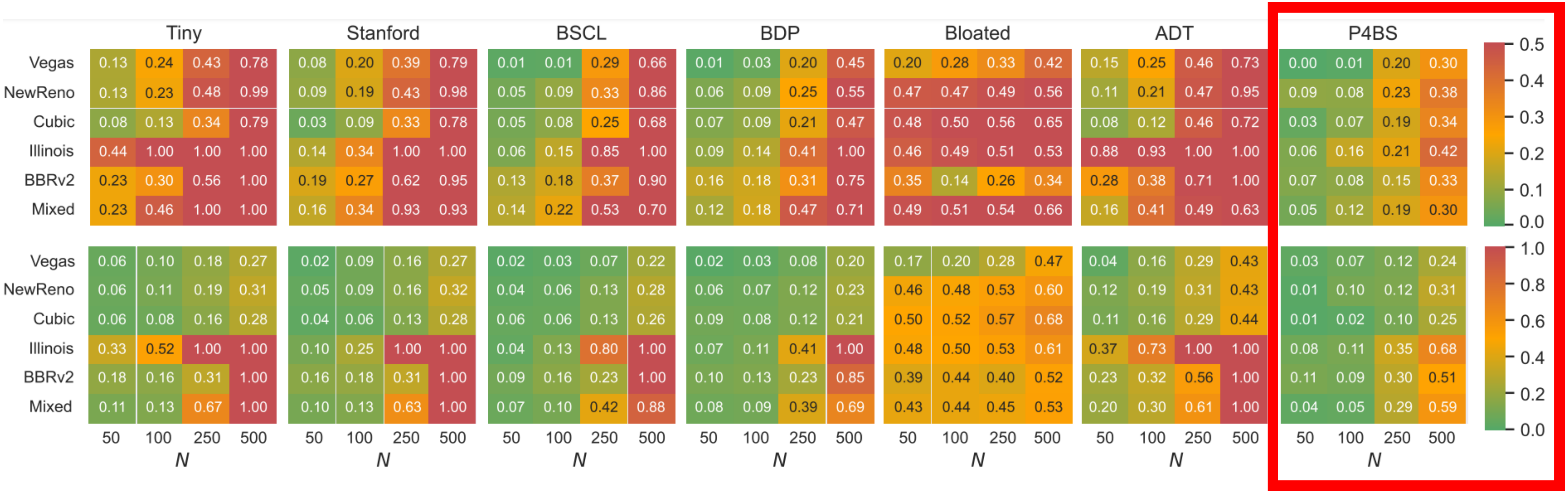
# Evaluation

- 1000 senders
- P4 switch: Wedge100BF-32X with Intel's Tofino ASIC
- Legacy router: Juniper router MX-204
- Different congestion control algorithms
- Access network:
  - ➢ $C_1$ = 40Gbps, $C_2$ = 1Gbps
- Core network:
  - ➢ $C_1$ = 10Gbps, $C_2$ = 2.5Gbps



# Results

- Combined metric accounting for packet loss and delay [0, 1] (the lower, the better)
- Top heatmaps: access network
- Bottom heatmaps: core network
- The Mixed scenario combines multiple congestion control algorithms[1]



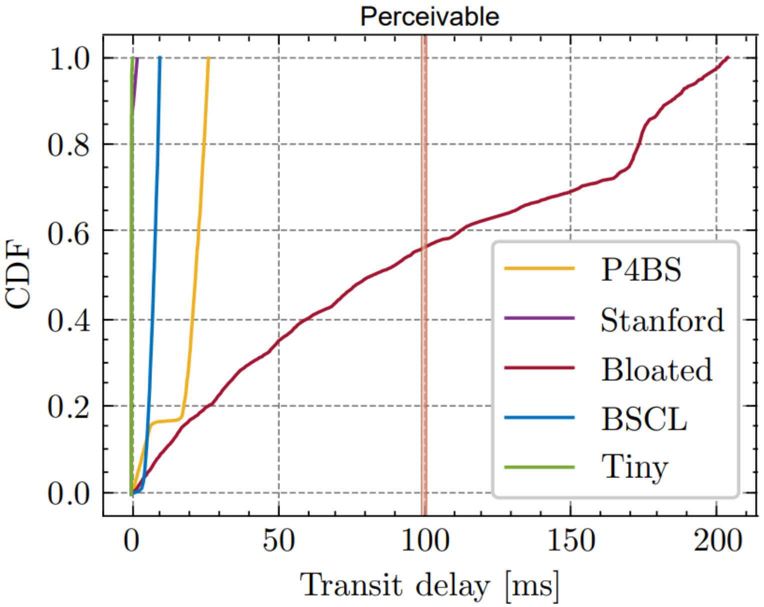
| | Tiny | | | | Stanford | | | | BSCL | | | | BDP | | | | Bloated | | | | ADT | | | | P4BS | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 50 | 100 | 250 | 500 | 50 | 100 | 250 | 500 | 50 | 100 | 250 | 500 | 50 | 100 | 250 | 500 | 50 | 100 | 250 | 500 | 50 | 100 | 250 | 500 | 50 | 100 | 250 | 500 |
| Vegas | 0.13 | 0.24 | 0.43 | 0.78 | 0.08 | 0.20 | 0.39 | 0.79 | 0.01 | 0.01 | 0.29 | 0.66 | 0.01 | 0.03 | 0.20 | 0.45 | 0.20 | 0.28 | 0.33 | 0.42 | 0.15 | 0.25 | 0.46 | 0.73 | 0.00 | 0.01 | 0.20 | 0.30 |
| NewReno | 0.13 | 0.23 | 0.48 | 0.99 | 0.09 | 0.19 | 0.43 | 0.98 | 0.05 | 0.09 | 0.33 | 0.86 | 0.06 | 0.09 | 0.25 | 0.55 | 0.47 | 0.47 | 0.49 | 0.56 | 0.11 | 0.21 | 0.47 | 0.95 | 0.09 | 0.08 | 0.23 | 0.38 |
| Cubic | 0.08 | 0.13 | 0.34 | 0.79 | 0.03 | 0.09 | 0.33 | 0.78 | 0.05 | 0.08 | 0.25 | 0.68 | 0.07 | 0.09 | 0.21 | 0.47 | 0.48 | 0.50 | 0.56 | 0.65 | 0.08 | 0.12 | 0.46 | 0.72 | 0.03 | 0.07 | 0.19 | 0.34 |
| Illinois | 0.44 | 1.00 | 1.00 | 1.00 | 0.14 | 0.34 | 1.00 | 1.00 | 0.06 | 0.15 | 0.85 | 1.00 | 0.09 | 0.14 | 0.41 | 1.00 | 0.46 | 0.49 | 0.51 | 0.53 | 0.88 | 0.93 | 1.00 | 1.00 | 0.06 | 0.16 | 0.21 | 0.42 |
| BBRv2 | 0.23 | 0.30 | 0.56 | 1.00 | 0.19 | 0.27 | 0.62 | 0.95 | 0.13 | 0.18 | 0.37 | 0.90 | 0.16 | 0.18 | 0.31 | 0.75 | 0.35 | 0.14 | 0.26 | 0.34 | 0.28 | 0.38 | 0.71 | 1.00 | 0.07 | 0.08 | 0.15 | 0.33 |
| Mixed | 0.23 | 0.46 | 1.00 | 1.00 | 0.16 | 0.34 | 0.93 | 0.93 | 0.14 | 0.22 | 0.53 | 0.70 | 0.12 | 0.18 | 0.47 | 0.71 | 0.49 | 0.51 | 0.54 | 0.66 | 0.16 | 0.41 | 0.49 | 0.63 | 0.05 | 0.12 | 0.19 | 0.30 |

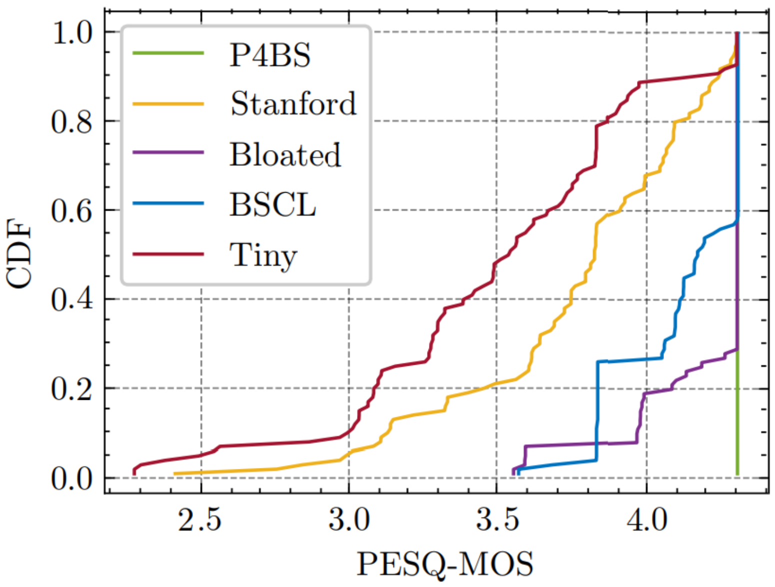
| | Tiny | | | | Stanford | | | | BSCL | | | | BDP | | | | Bloated | | | | ADT | | | | P4BS | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 50 | 100 | 250 | 500 | 50 | 100 | 250 | 500 | 50 | 100 | 250 | 500 | 50 | 100 | 250 | 500 | 50 | 100 | 250 | 500 | 50 | 100 | 250 | 500 | 50 | 100 | 250 | 500 |
| Vegas | 0.06 | 0.10 | 0.18 | 0.27 | 0.02 | 0.09 | 0.16 | 0.27 | 0.02 | 0.03 | 0.07 | 0.22 | 0.02 | 0.03 | 0.08 | 0.20 | 0.17 | 0.20 | 0.28 | 0.47 | 0.04 | 0.16 | 0.29 | 0.43 | 0.03 | 0.07 | 0.12 | 0.24 |
| NewReno | 0.06 | 0.11 | 0.19 | 0.31 | 0.05 | 0.09 | 0.16 | 0.32 | 0.04 | 0.06 | 0.13 | 0.28 | 0.06 | 0.07 | 0.12 | 0.23 | 0.46 | 0.48 | 0.53 | 0.60 | 0.12 | 0.19 | 0.31 | 0.43 | 0.01 | 0.10 | 0.12 | 0.31 |
| Cubic | 0.06 | 0.08 | 0.16 | 0.28 | 0.04 | 0.06 | 0.13 | 0.28 | 0.06 | 0.06 | 0.13 | 0.26 | 0.09 | 0.08 | 0.12 | 0.21 | 0.50 | 0.52 | 0.57 | 0.68 | 0.11 | 0.16 | 0.29 | 0.44 | 0.01 | 0.02 | 0.10 | 0.25 |
| Illinois | 0.33 | 0.52 | 1.00 | 1.00 | 0.10 | 0.25 | 1.00 | 1.00 | 0.04 | 0.13 | 0.80 | 1.00 | 0.07 | 0.11 | 0.41 | 1.00 | 0.48 | 0.50 | 0.53 | 0.61 | 0.37 | 0.73 | 1.00 | 1.00 | 0.08 | 0.11 | 0.35 | 0.68 |
| BBRv2 | 0.18 | 0.16 | 0.31 | 1.00 | 0.16 | 0.18 | 0.31 | 1.00 | 0.09 | 0.16 | 0.23 | 1.00 | 0.10 | 0.13 | 0.23 | 0.85 | 0.39 | 0.44 | 0.40 | 0.52 | 0.23 | 0.32 | 0.56 | 1.00 | 0.11 | 0.09 | 0.30 | 0.51 |
| Mixed | 0.11 | 0.13 | 0.67 | 1.00 | 0.10 | 0.13 | 0.63 | 1.00 | 0.07 | 0.10 | 0.42 | 0.88 | 0.08 | 0.09 | 0.39 | 0.69 | 0.43 | 0.44 | 0.45 | 0.53 | 0.20 | 0.30 | 0.61 | 1.00 | 0.04 | 0.05 | 0.29 | 0.59 |

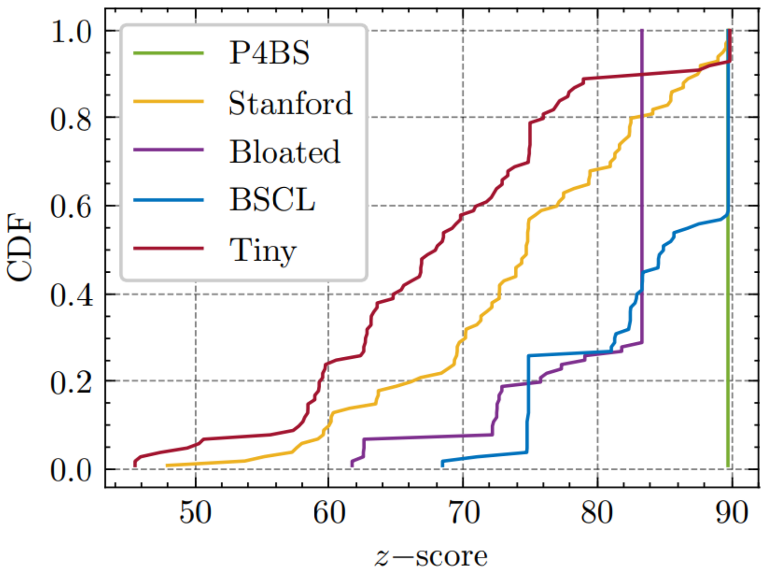
[1] Mishra et al. "The great Internet TCP congestion control census," ACM on Measurement and Analysis of Computing Systems, 2019

# Results

- 100 VoIP calls playing 20 reference speech samples (G.711.a)
- The Perceptual Evaluation of Speech Quality (PESQ) compares an error-free audio signal to a degraded one (the higher, the better)
- The z-score considers both the delay and the PESQ (the higher, the better)

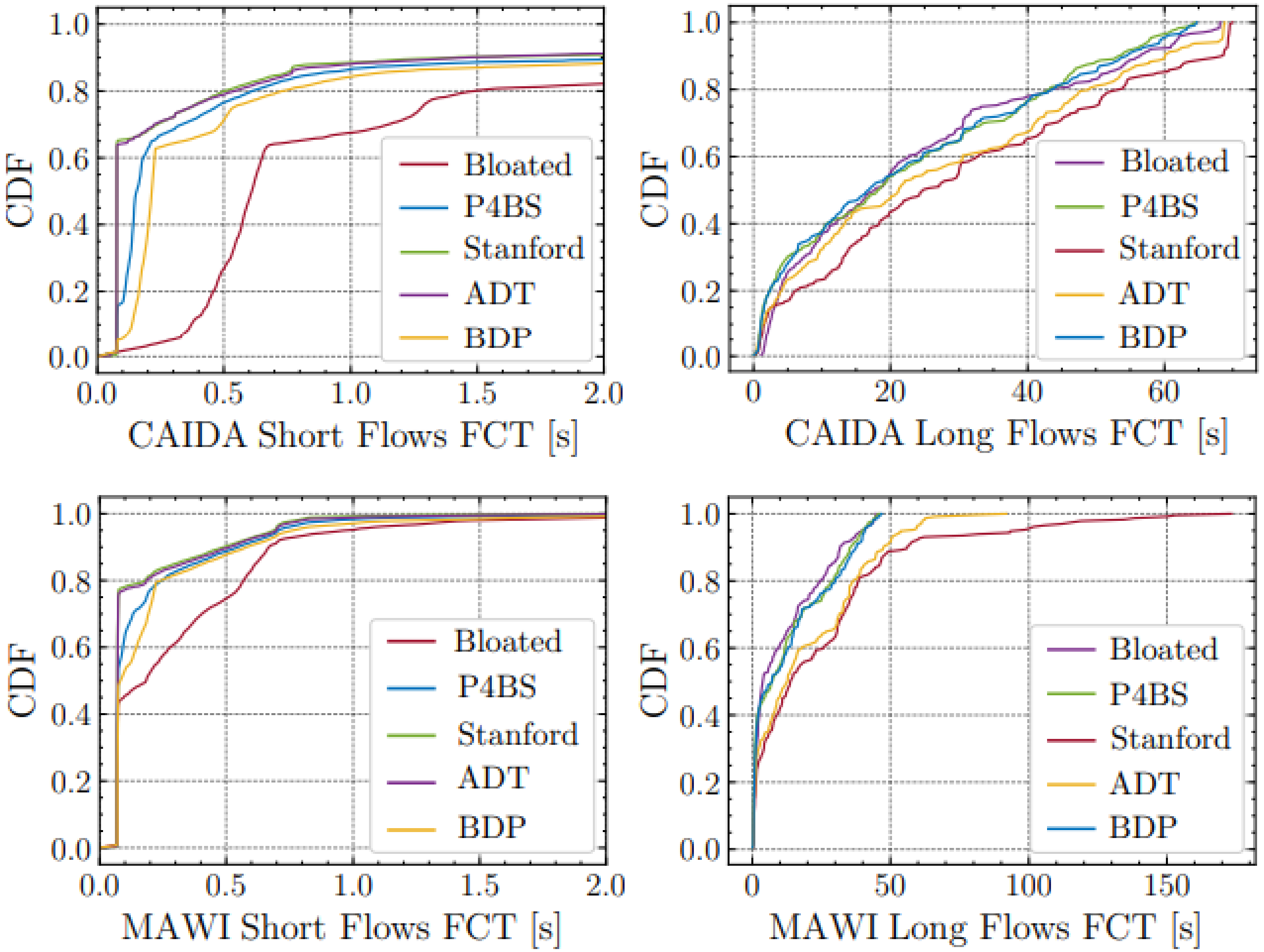






# Results

- These results use real traffic traces from CAIDA[1] and MAWI[2]
- They include long and short flows
- P4BS found a balance such that:
  - ➢ The FCT of short flows is close to that of the Stanford buffer
  - ➢ The FCT of long flows is close to that of the bloated buffer



[1] Center for Applied Internet Data Analysis (CAIDA). https://www.caida.org/

[2] MAWI Working Group Traffic Archive. https://mawi.wide.ad.jp/mawi/

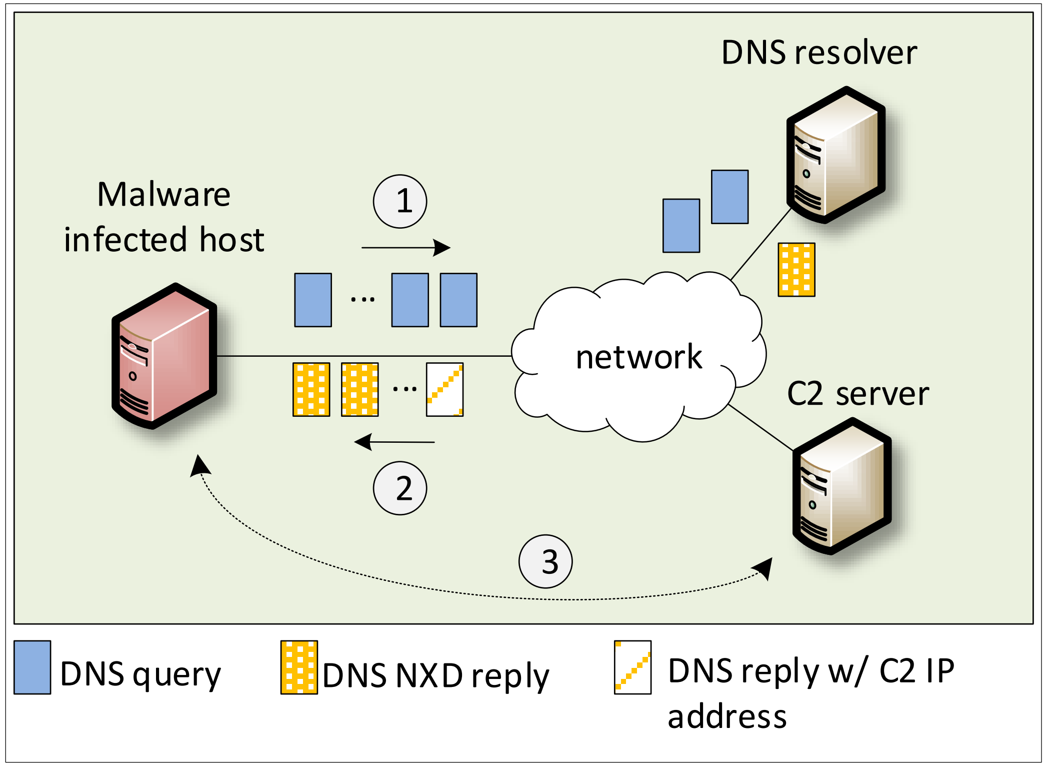
# DGA Family Classification using DNS Deep Packet Inspection on P4 Programmable Switches

# Introduction to DGAs

- Attackers often use a Command and Control (C2) server to establish communication between infected host/s and bot master
- Domain Generation Algorithms (DGAs) are the *de facto* dynamic C2 communication method used by malware, including botnets, ransomware, and many others

# Introduction to DGAs

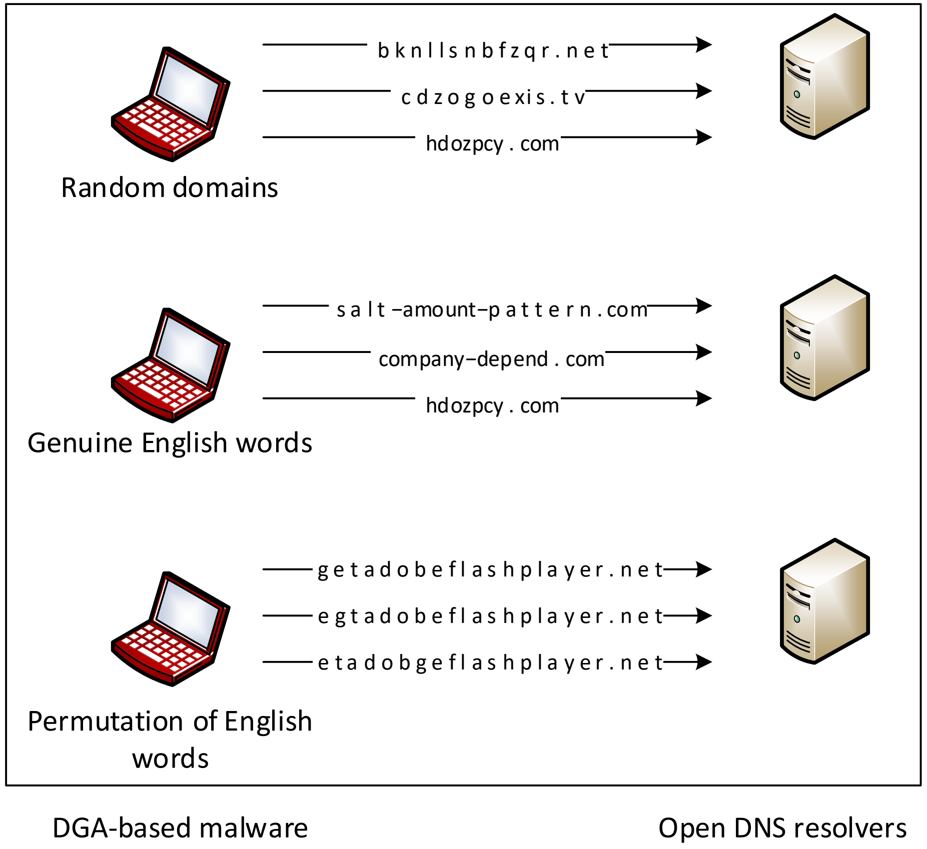
- DGAs evade firewall controls by frequently changing the domain name selected from a large pool of candidates
- The malware makes DNS queries to resolve the IP addresses of these generated domains
- Only a few of these queries will be successful; most of them will result in Non-Existent Domain (NXD) responses



(1) DNS queries. (2) (NXD) replies. (3) Eventually, a query for the actual domain is sent and malware-C2 communication starts.

# Introduction to DGAs

- DGAs evade firewall controls by frequently changing the domain name selected from a large pool of candidates
- The malware makes DNS queries to resolve the IP addresses of these generated domains
- Only a few of these queries will be successful; most of them will result in Non-Existent Domain (NXD) responses

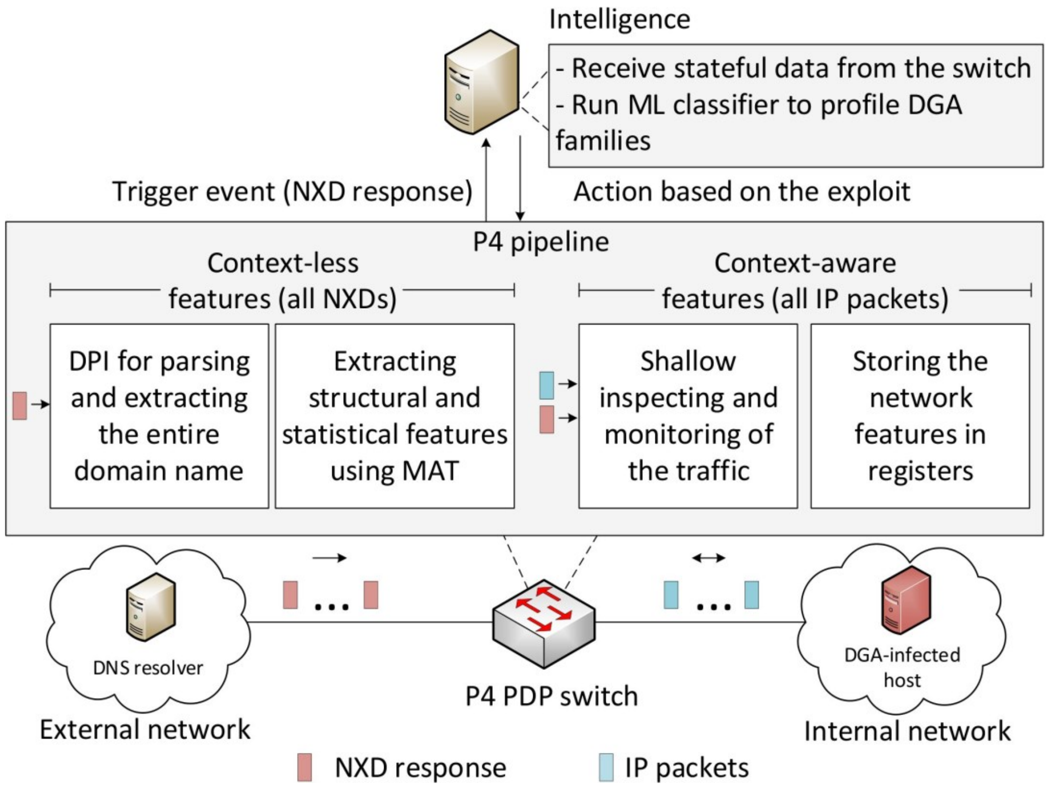


# Existing Mitigation Techniques

- **Context-aware** approaches analyze the **network traffic** behavior to fingerprint DGAs
  - Slow since they typically analyze batches of traffic offline
- **Context-less** approaches analyze domain names (**DNS-based**) via ML models
  - The use of a general-purpose CPU/GPU may create a bottleneck due to high traffic volume
- There is a need for a system that
  - uses both context-aware and context-less features
  - detects and classifies DGAs based on the family (Trojan, backdoor, etc.)

# Proposed System

- The P4 PDP switch collects and stores the **context-aware (traffic) features** of the hosts
  - Number of IP addresses contacted
  - Inter-arrival Time (IAT) between consecutive IP packets
  - Number of DNS requests made
  - Time it takes for the first NXD response to arrive
  - IAT between subsequent NXD responses[1]



Intelligence

- Receive stateful data from the switch
- Run ML classifier to profile DGA families

Trigger event (NXD response)

Action based on the exploit

P4 pipeline

Context-less features (all NXDs)

Context-aware features (all IP packets)

DPI for parsing and extracting the entire domain name

Extracting structural and statistical features using MAT

Shallow inspecting and monitoring of the traffic

Storing the network features in registers

DNS resolver

External network

P4 PDP switch

DGA-infected host

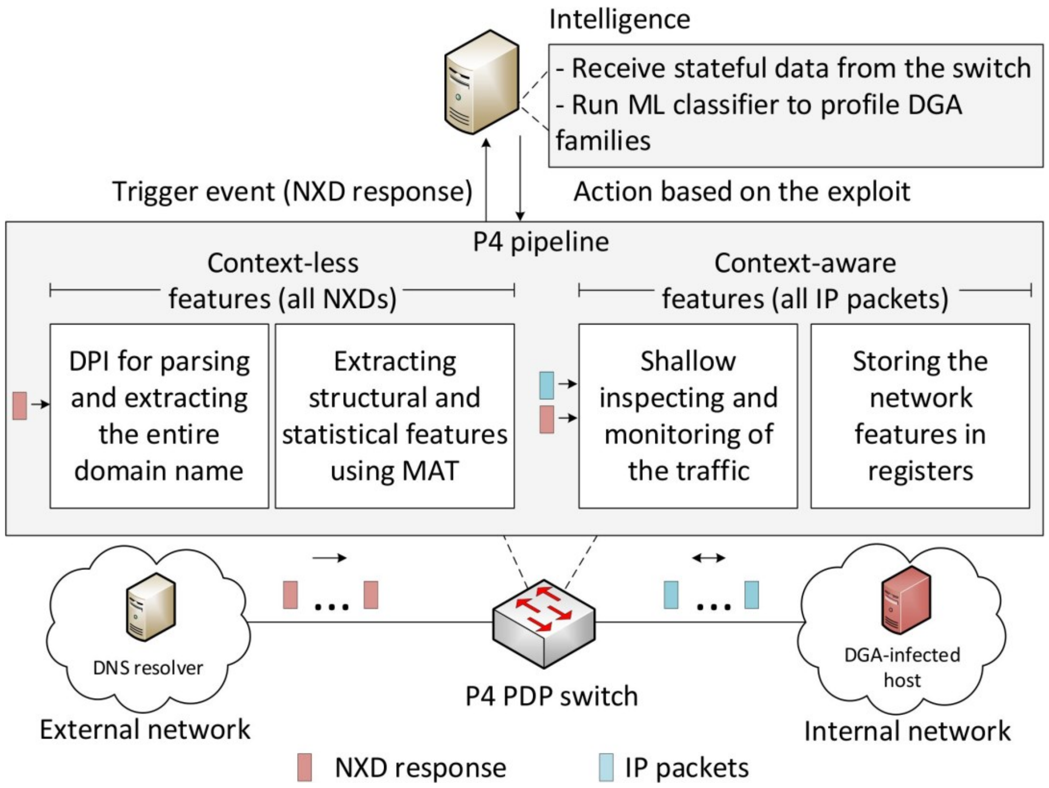
Internal network

🟥 NXD response 🟦 IP packets

[1] A. AlSabeh, K. Friday, J. Crichigno, E. Bou-Harb, "Effective DGA Family Classification using a Hybrid Shallow and Deep Packet Inspection Technique on P4 Programmable Switches", IEEE International Conference on Communications (ICC), Rome, Italy, June 2023.

# Proposed System

- When an NXD response is received, the switch performs DPI on the domain name to extract **context-less (domain) features**
  - The switch sends the collected features to the control plane
  - The control plane runs the intelligence to classify the DGA family and initiate the appropriate incidence response[1]


Intelligence
- Receive stateful data from the switch
- Run ML classifier to profile DGA families
Trigger event (NXD response)
Action based on the exploit
P4 pipeline
Context-less features (all NXDs)
Context-aware features (all IP packets)
DPI for parsing and extracting the entire domain name
Extracting structural and statistical features using MAT
Shallow inspecting and monitoring of the traffic
Storing the network features in registers
DNS resolver
External network
P4 PDP switch
DGA-infected host
Internal network
NXD response
IP packets

[1] A. AlSabeh, K. Friday, J. Crichigno, E. Bou-Harb, "Effective DGA Family Classification using a Hybrid Shallow and Deep Packet Inspection Technique on P4 Programmable Switches", IEEE International Conference on Communications (ICC), Rome, Italy, June 2023.

# Proposed System

- The scheme uses the bigram technique for **context-less (domain)** analysis:
  - It computes the bigram of the domain name; a bigram model may suffice to predict whether a domain name is a legitimate human readable domain

$$score\ (d) = \sum_{\forall\ subdomain\ s\ \in\ d} \left( \sum_{\forall\ bigram\ b\ \in\ s} f_s^b \right)$$

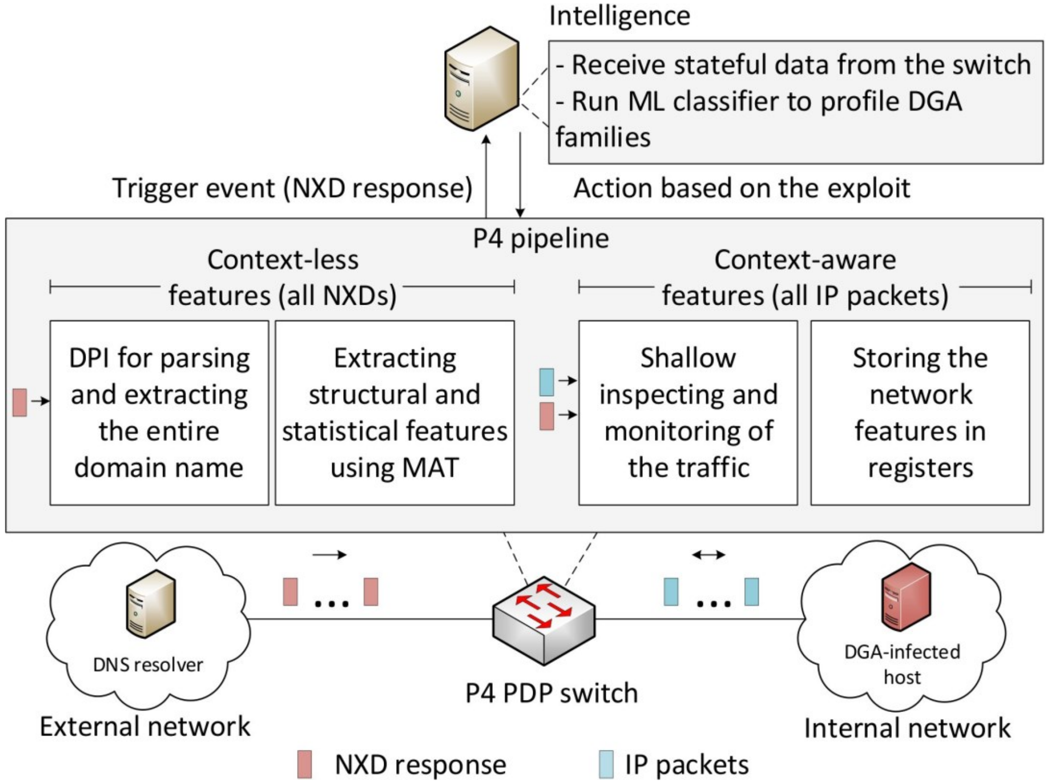
Where $f_s^b$ is the frequency of the bigram b in the subdomain *s*

  - The frequency value of a bigram b is pre-computed and stored in a Match-Action Table (MAT)
  - The lower the score, the more random the domain name
  - Example: the bigrams of “google” are: “$g”, “go”, “oo”, “og”, “gl”, “le”, “e$”

[1] A. AlSabeh, K. Friday, J. Crichigno, E. Bou-Harb, "Effective DGA Family Classification using a Hybrid Shallow and Deep Packet Inspection Technique on P4 Programmable Switches", IEEE International Conference on Communications (ICC), Rome, Italy, June 2023.

# Proposed System

- Note that the system incorporates
  - Customized packet parsing and processing
  - Fine-grained measurements
  - Per-packet traffic inspection
  - Stateful memory processing at line rate


Intelligence
- Receive stateful data from the switch
- Run ML classifier to profile DGA families
Trigger event (NXD response)
Action based on the exploit
P4 pipeline
Context-less features (all NXDs)
Context-aware features (all IP packets)
DPI for parsing and extracting the entire domain name
Extracting structural and statistical features using MAT
Shallow inspecting and monitoring of the traffic
Storing the network features in registers
DNS resolver
External network
P4 PDP switch
DGA-infected host
Internal network
NXD response
IP packets

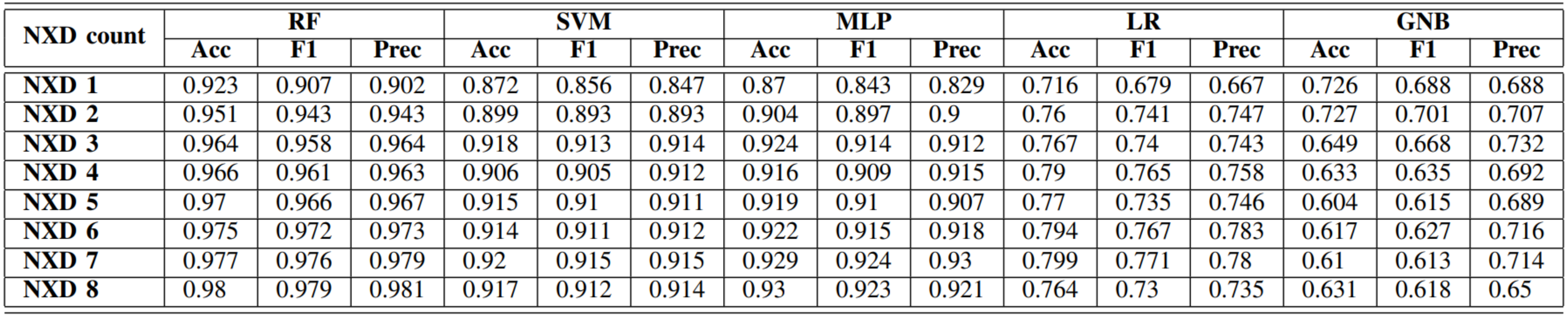
# Evaluation

- Experimental setup
  - Hundreds of GB of malware samples; 1,311 samples containing 50 DGA families[1]
  - We used samples that receive NXD responses containing domain names generated by DGAs[1]
  - The collected dataset was used to train ML models offline on a general-purpose CPU
  - 80% of data was used for training and 20% for testing

[1] D. Lohmann, “DGArchive.” [Online]. Available: https://tinyurl. com/yc6whwrc.

# Evaluation

- The evaluation reports the accuracy (Acc), F1 score, and Precision (Prec) of different ML classifiers during the first eight NXD responses
  - ➢ The Random Forest (RF) model performed best
  - ➢ The Accuracy (Acc) starts at 92% from the first NXD response received and reaches 98% by the 8th NXD response

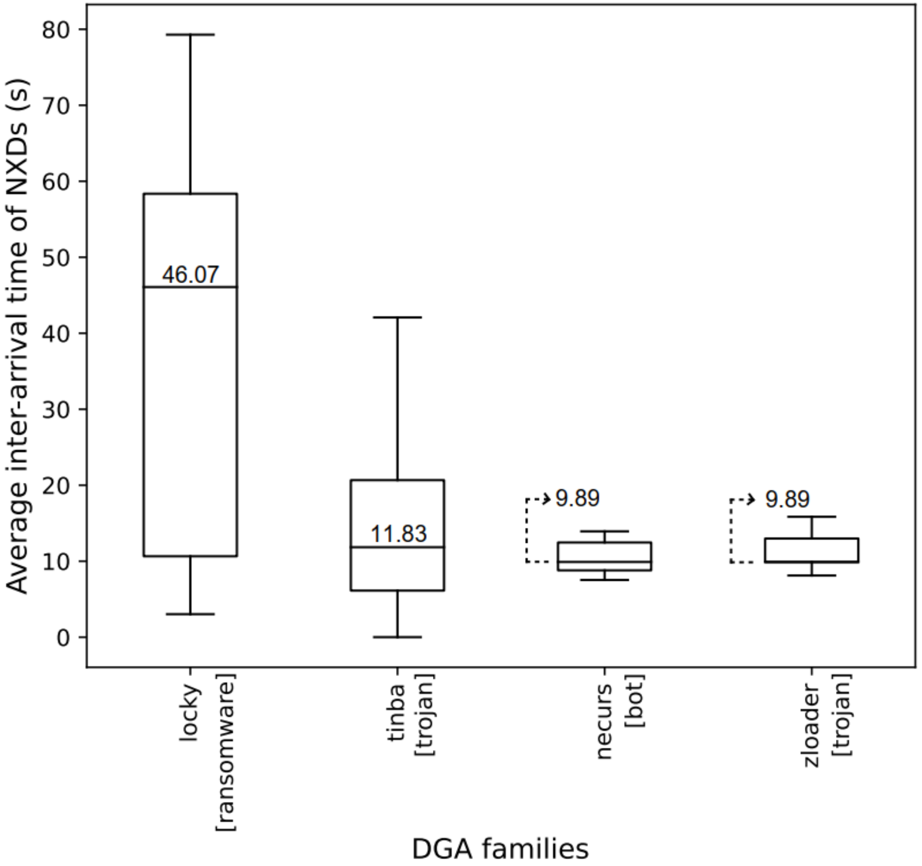


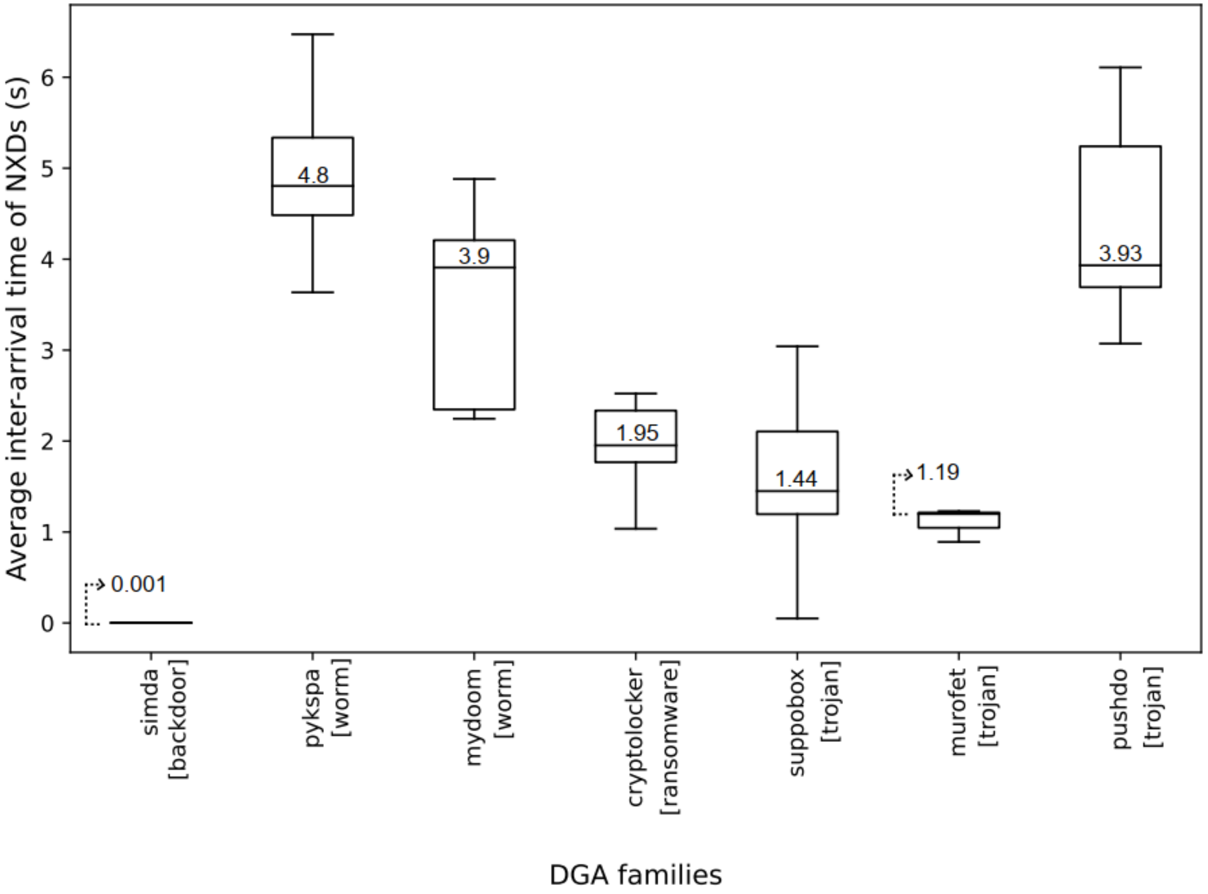
| NXD count | RF | | | SVM | | | MLP | | | LR | | | GNB | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Acc** | **F1** | **Prec** | **Acc** | **F1** | **Prec** | **Acc** | **F1** | **Prec** | **Acc** | **F1** | **Prec** | **Acc** | **F1** | **Prec** |
| **NXD 1** | 0.923 | 0.907 | 0.902 | 0.872 | 0.856 | 0.847 | 0.87 | 0.843 | 0.829 | 0.716 | 0.679 | 0.667 | 0.726 | 0.688 | 0.688 |
| **NXD 2** | 0.951 | 0.943 | 0.943 | 0.899 | 0.893 | 0.893 | 0.904 | 0.897 | 0.9 | 0.76 | 0.741 | 0.747 | 0.727 | 0.701 | 0.707 |
| **NXD 3** | 0.964 | 0.958 | 0.964 | 0.918 | 0.913 | 0.914 | 0.924 | 0.914 | 0.912 | 0.767 | 0.74 | 0.743 | 0.649 | 0.668 | 0.732 |
| **NXD 4** | 0.966 | 0.961 | 0.963 | 0.906 | 0.905 | 0.912 | 0.916 | 0.909 | 0.915 | 0.79 | 0.765 | 0.758 | 0.633 | 0.635 | 0.692 |
| **NXD 5** | 0.97 | 0.966 | 0.967 | 0.915 | 0.91 | 0.911 | 0.919 | 0.91 | 0.907 | 0.77 | 0.735 | 0.746 | 0.604 | 0.615 | 0.689 |
| **NXD 6** | 0.975 | 0.972 | 0.973 | 0.914 | 0.911 | 0.912 | 0.922 | 0.915 | 0.918 | 0.794 | 0.767 | 0.783 | 0.617 | 0.627 | 0.716 |
| **NXD 7** | 0.977 | 0.976 | 0.979 | 0.92 | 0.915 | 0.915 | 0.929 | 0.924 | 0.93 | 0.799 | 0.771 | 0.78 | 0.61 | 0.613 | 0.714 |
| **NXD 8** | 0.98 | 0.979 | 0.981 | 0.917 | 0.912 | 0.914 | 0.93 | 0.923 | 0.921 | 0.764 | 0.73 | 0.735 | 0.631 | 0.618 | 0.65 |

RF: Random Forest; SVM: Support Vector Machine; MLP: Multilayer perceptron; LR: Logistic Regression; GNB: Gaussian Naive Bayes

# Evaluation

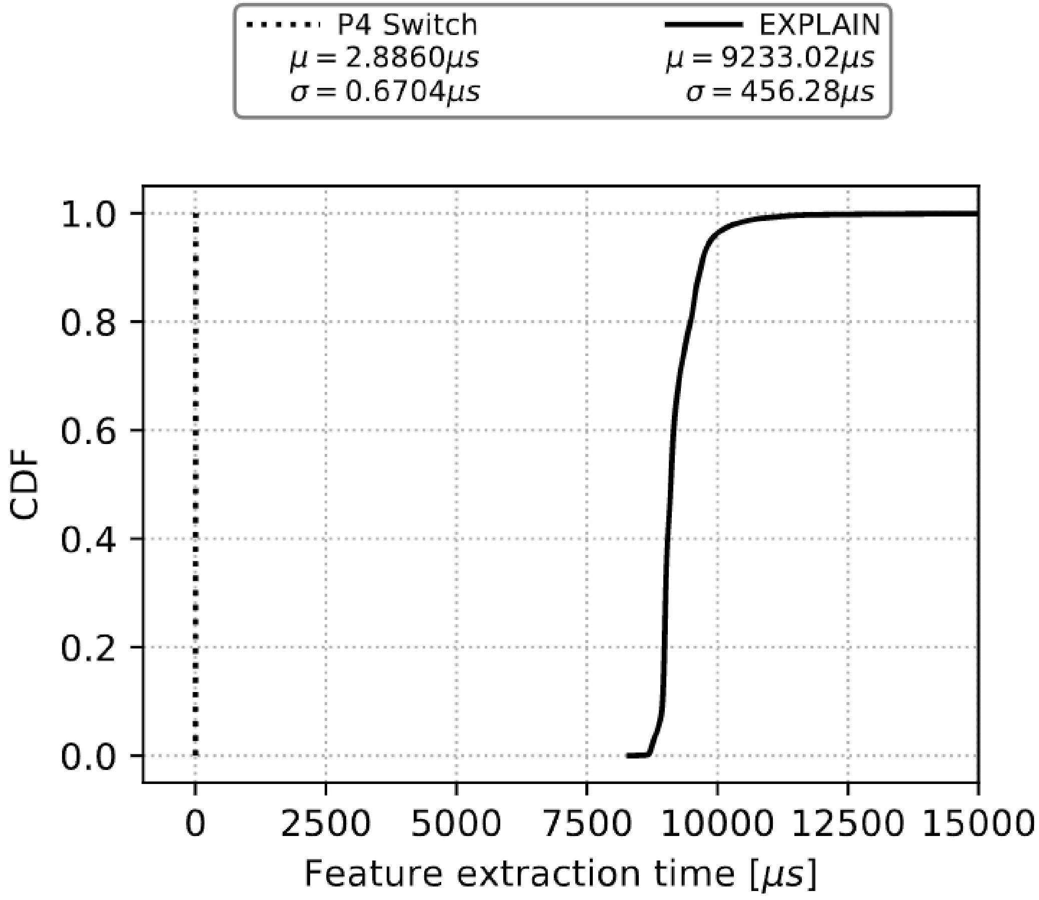
- The scheme can accurately characterize traffic flows (context-aware features)
- Interarrival times between NXDs of DGA families with the largest number of samples





# Evaluation

- Comparison of the feature extraction time of the proposed approach vs EXPLAIN[1]
  - The proposed approach runs on the switch data plane
  - EXPLAIN runs on a general-purposed CPU with 64 GB RAM, 2.9 GHz processor with eight cores



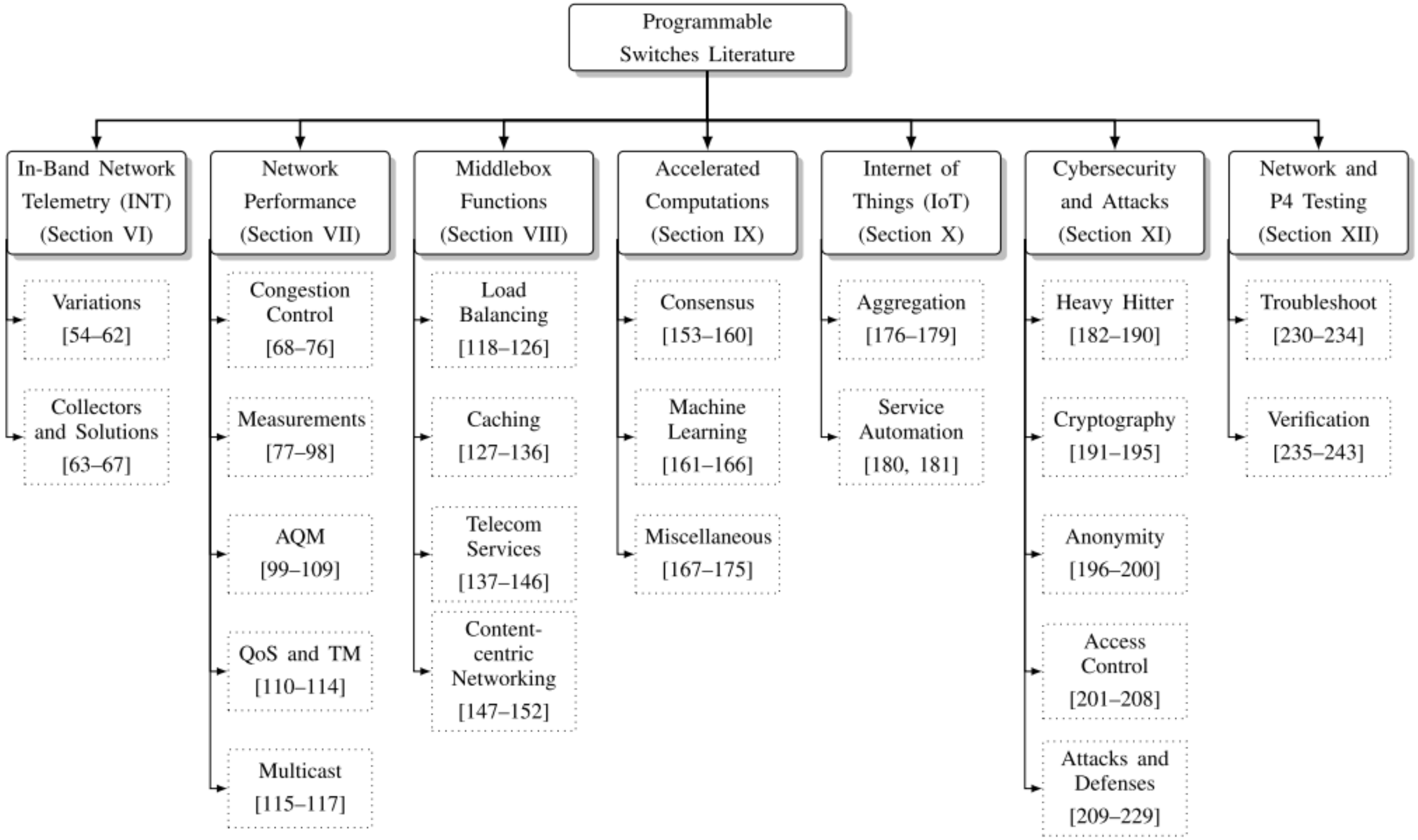
[1] A. Drichel, N. Faerber, U. Meyer, “First step towards explainable DGA multiclass classification,” in the 16th International Conference on Availability, Reliability and Security, pp. 1–13, 2021.

# Conclusion

- This presentation briefly described the evolution of networking devices, from legacy (monolithic) units to SDN to P4 PDP switches
- It discussed the capabilities offered by PDP switches to enable end programmers to produce fine-grained measurements, customized parsers and functions, and line-rate computation
- Such capabilities were applied to solve two different problems
  - ➢ Buffer sizing problem, where programmability was enabled in non-programmable devices, to solve the buffer sizing problem via a passive deployment of P4 switches
  - ➢ DGA problem, where the P4 application was able to detect and classify DGAs using a combination of DNS deep packet inspection and traffic characterization

# Conclusion

- The previous two are only a couple of examples of the impressive work produced by the P4 community, which suggests that deep programmability (switches, smart NICs, etc.) will continue in the near future[1]



Programmable Switches Literature

In-Band Network Telemetry (INT) (Section VI)
- Variations [54–62]
- Collectors and Solutions [63–67]

Network Performance (Section VII)
- Congestion Control [68–76]
- Measurements [77–98]
- AQM [99–109]
- QoS and TM [110–114]
- Multicast [115–117]

Middlebox Functions (Section VIII)
- Load Balancing [118–126]
- Caching [127–136]
- Telecom Services [137–146]
- Content-centric Networking [147–152]

Accelerated Computations (Section IX)
- Consensus [153–160]
- Machine Learning [161–166]
- Miscellaneous [167–175]

Internet of Things (IoT) (Section X)
- Aggregation [176–179]
- Service Automation [180, 181]

Cybersecurity and Attacks (Section XI)
- Heavy Hitter [182–190]
- Cryptography [191–195]
- Anonymity [196–200]
- Access Control [201–208]
- Attacks and Defenses [209–229]

Network and P4 Testing (Section XII)
- Troubleshoot [230–234]
- Verification [235–243]

[1] E. Kfoury, J. Crichigno, E. Bou-Harb, "An Exhaustive Survey on P4 Programmable Data Plane Switches: Taxonomy, Applications, Challenges, and Future Trends", IEEE Access, June 2021.

# Contact Information

Jorge Crichigno
College of Engineering and Computing, University of South Carolina
jcrichigno@cec.sc.edu
http://ce.sc.edu/cyberinfra

UNIVERSITY OF
SOUTH CAROLINA