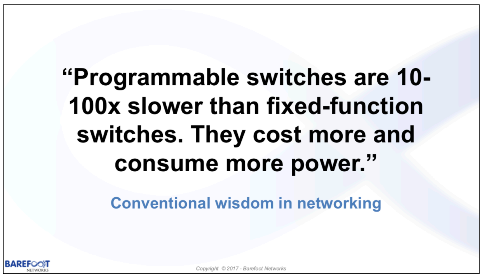
Advanced Topics in Communication Networks

Prof. Laurent Vanbever

Last week on

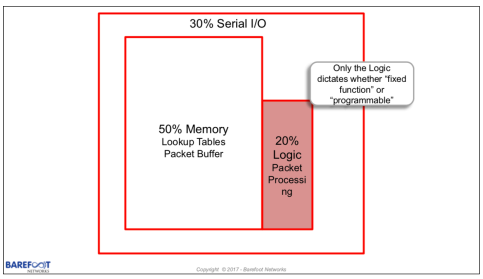
Advanced Topics in Communication Networks

We looked at the Tofino architecture together with two (key, value) store applications: Net/{Cache, Chain}



Source: Programmable Data Planes at Terabit Speeds, Vladimir Gurevich, 2017

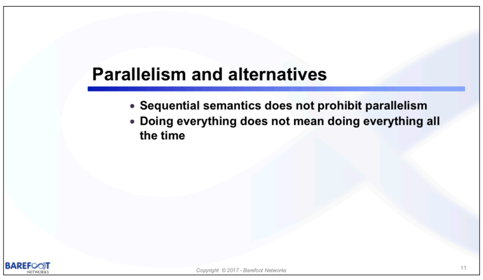
One of the main enabler for data-plane programmability is the shrinking size of the packet processing logic chip.



Source: Programmable Data Planes at Terabit Speeds, Vladimir Gurevich, 2017

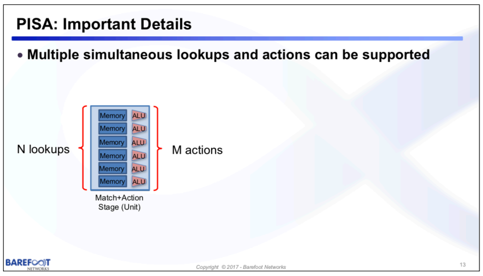
Barefoot Tofino processes packets in parallel,

even though the semantic of a P4 program is sequential



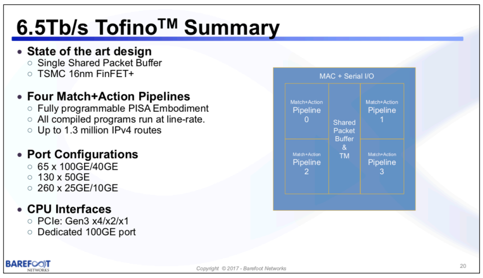
Source: Programmable Data Planes at Terabit Speeds, Vladimir Gurevich, 2017

Barefoot Tofino processes packets in parallel,
even though the semantic of a P4 program is sequential



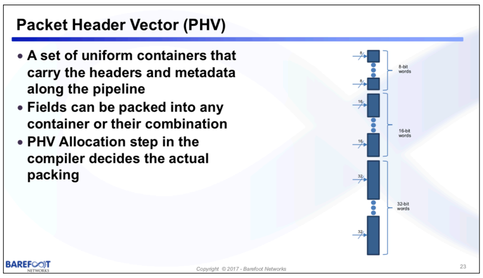
Source: Programmable Data Planes at Terabit Speeds, Vladimir Gurevich, 2017

Barefoot Tofino 6.5 Tbps backplane
several billion packets per second at line rate



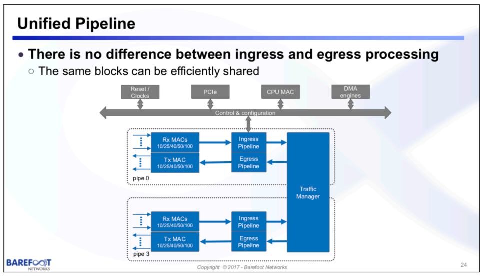
Source: Programmable Data Planes at Terabit Speeds, Vladimir Gurevich, 2017

Tofino relies on Packet Header Vector (PHV) to pass states between stages—this is one of the limiting factor



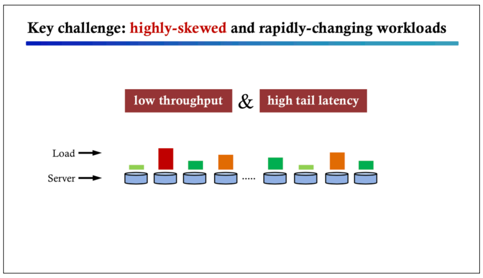
Source: Programmable Data Planes at Terabit Speeds, Vladimir Gurevich, 2017

Tofino uses a folded pipeline in which the *same* stages are used for both the ingress and the egress pipeline



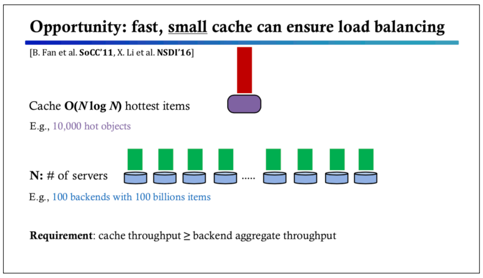
Source: Programmable Data Planes at Terabit Speeds, Vladimir Gurevich, 2017

NetCache solves the problem of load-balancing in key-values stores observing *dynamic*, *skewed* workload



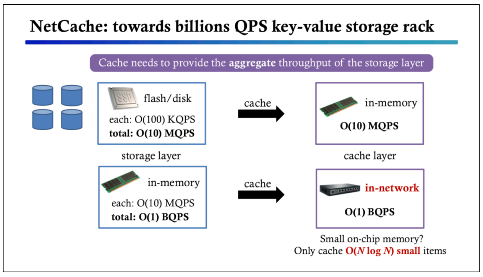
Source: NetCache: Balancing Key-Value Stores with Fast In-Network Caching, Xin Jin, 2017

It leverages that a small but very fast cache can provide perfect load-balancing… in theory



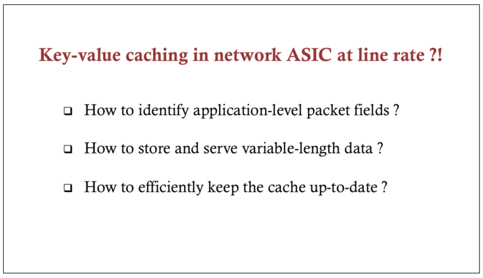
Source: NetCache: Balancing Key-Value Stores with Fast In-Network Caching, Xin Jin, 2017

NetCache relies on the O(billion) throughput of programmable network devices to achieve it in practice



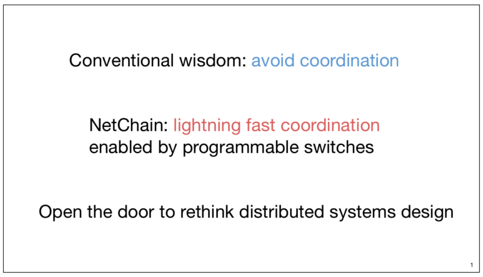
Source: NetCache: Balancing Key-Value Stores with Fast In-Network Caching, Xin Jin, 2017

It relies on a tailored UDP-based protocol, an de/encoding scheme for storing variable length values, and sketches



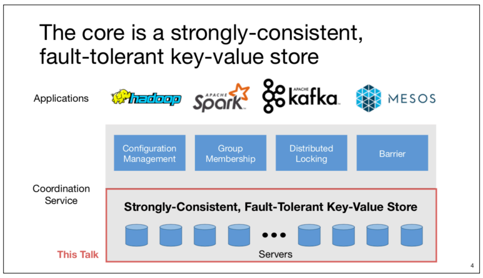
Source: NetCache: Balancing Key-Value Stores with Fast In-Network Caching, Xin Jin, 2017

NetChain builds upon NetCache to scale coordination services, a key building block of distributed systems



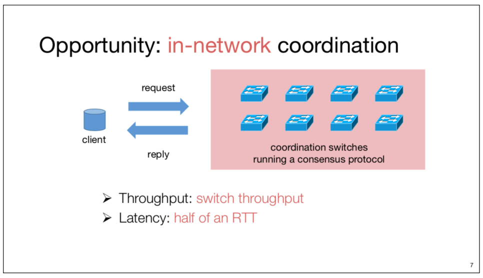
Source: NetChain: Scale-Free Sub-RTT Coordination, Xin Jin, 2018

Coordination services typically rely on a replicated key-value store for consistency and fault-tolerance



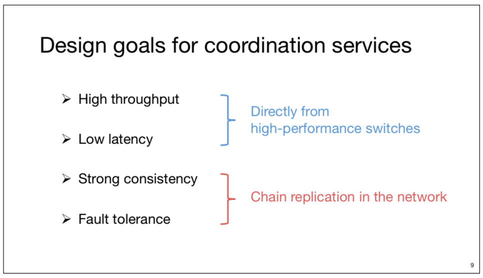
Source: NetChain: Scale-Free Sub-RTT Coordination, Xin Jin, 2018

State of the art server-based coordination services struggle to provide high-throughput and low-latency



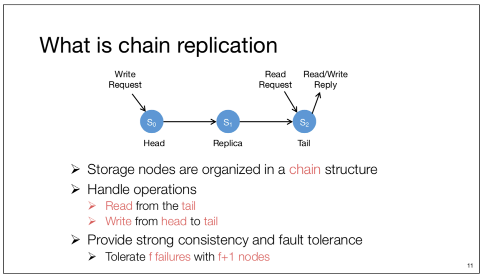
Source: NetChain: Scale-Free Sub-RTT Coordination, Xin Jin, 2018

Key challenge is to ensure consistency and fault-tolerance



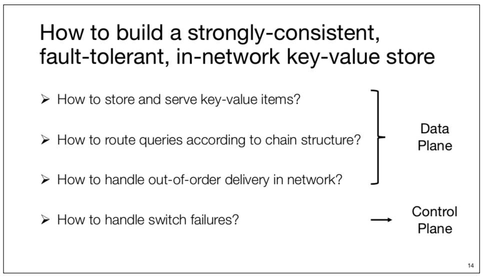
Source: NetChain: Scale-Free Sub-RTT Coordination, Xin Jin, 2018

NetChain does so using chain replication, building upon NetCache for storing values in each switch



Source: NetChain: Scale-Free Sub-RTT Coordination, Xin Jin, 2018

NetChain relies on a tailored UDP-based protocol, source-routing mechanisms and message serialization



Source: NetChain: Scale-Free Sub-RTT Coordination, Xin Jin, 2018

This week on

Advanced Topics in Communication Networks

A high-level, non-exhaustive overview of the research surrounding data plane programmability

A high-level, non-exhaustive overview of the research surrounding data plane programmability

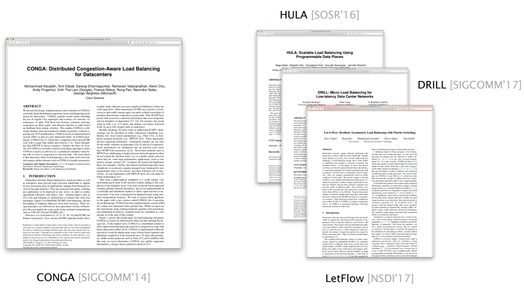
Data plane programmability for Performance, Monitoring, Applications offloading

Platforms, Correctness, Management for Data plane programmability

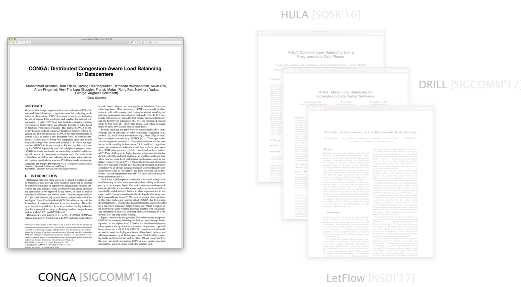
Data plane programmability for **Performance**, Monitoring, Applications offloading

Platforms, Correctness, Management for Data plane programmability

A large set of papers on programmable data planes aim at improving performance, esp. load balancing



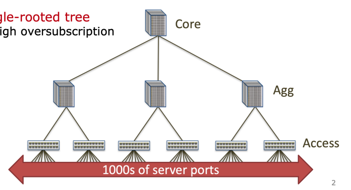
A large set of papers on programmable data planes aim at improving performance, esp. load balancing



Motivation

DC networks need large bisection bandwidth for distributed apps (big data, HPC, web services, etc)

Single-rooted tree
- High oversubscription



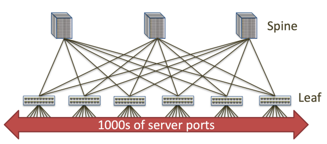
2

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

Motivation

DC networks need large bisection bandwidth for distributed apps (big data, HPC, web services, etc)

Multi-rooted tree [Fat-tree, Leaf-Spine, …]
- Full bisection bandwidth, achieved via multipathing



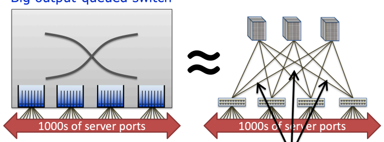
2

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

Multi-rooted != Ideal DC Network

Ideal DC network:
Big output-queued switch

Multi-rooted tree



- No internal bottlenecks → predictable
- Simplifies BW management
  [EyeQ, FairCloud, pFabric, Varys, …]

Possible bottlenecks

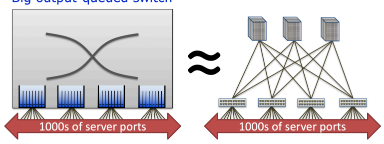
4

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

Multi-rooted != Ideal DC Network

Ideal DC network:
Big output-queued switch

Multi-rooted tree



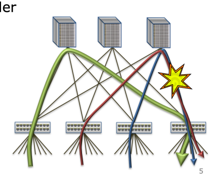
Need precise load balancing

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

Today: ECMP Load Balancing

Pick among equal-cost paths by a hash of 5-tuple
- Approximates Valiant load balancing
- Preserves packet order

Problems:
- Hash collisions
  (coarse granularity)
- Local & stateless
  (v. bad with asymmetry due to link failures)

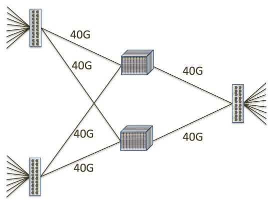


5

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

Dealing with Asymmetry

Handling asymmetry needs non-local knowledge

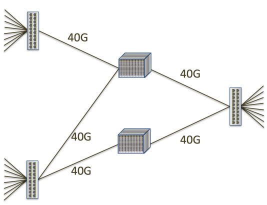


6

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

Dealing with Asymmetry

Handling asymmetry needs non-local knowledge

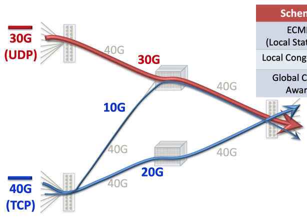


6

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

Dealing with Asymmetry: ECMP

Scheme	Thrput
ECMP (Local Stateless)	60G
Local Cong-Aware	
Global Cong-Aware	

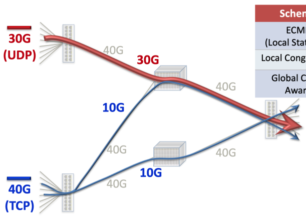


8

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

Dealing with Asymmetry: Local Congestion-Aware

Scheme	Thrput
ECMP (Local Stateless)	60G
Local Cong-Aware	50G
Global Cong-Aware	

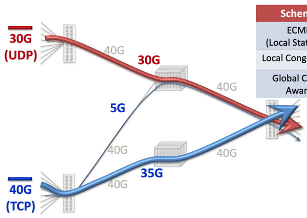


9

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

Dealing with Asymmetry: Global Congestion-Aware

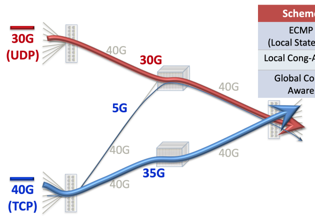
Scheme	Thrput
ECMP (Local Stateless)	60G
Local Cong-Aware	50G
Global Cong-Aware	70G



10

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

Dealing with Asymmetry: Global Congestion-Aware



Scheme	Thrput
ECMP (Local Stateless)	60G
Local Cong-Aware	50G
Global Cong-Aware	70G

10

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

Global Congestion-Awareness (in Datacenters)

		Datacenter
Opportunity →	Latency	microseconds
	Topology	simple, regular
Challenge →	Traffic	volatile, bursty

11

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

Global Congestion-Awareness (in Datacenters)

		Datacenter
Opportunity →	Latency	microseconds
	Topology	simple, regular
Challenge →	Traffic	volatile, bursty

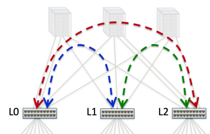
Key Insight:
Use *extremely* fast, low latency distributed control

11

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

CONGA in 1 Slide

1. Leaf switches (top-of-rack) track congestion to other leaves on different paths in near real-time
1. Use greedy decisions to minimize bottleneck util



Fast feedback loops between leaf switches, directly in dataplane

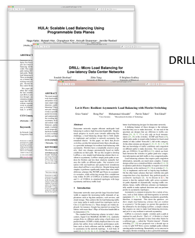
12

Source: CONGA: Distributed Congestion-Aware Load Balancing for Datacenters, Mohammad Alizadeh et al., 2014

A large set of papers on programmable data planes aim at improving performance, esp. load balancing

P4-based data-plane load-balancing with better scalability than CONGA — HULA [SOSR'16]

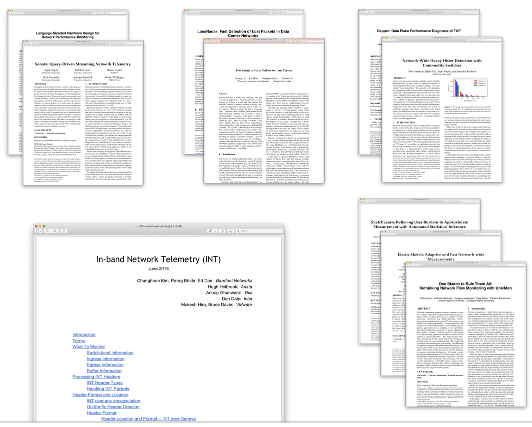
"micro" load balancing, packet-by-packet, can deal with micro-bursts — DRILL [SIGCOMM'17]

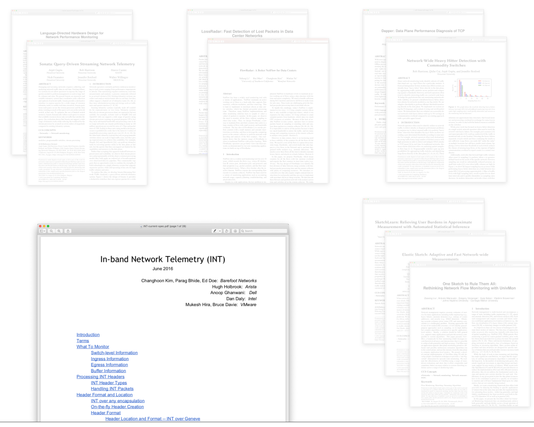


stateless, yet congestion-aware load-balancing decision — LetFlow [NSDI'17]

Data plane programmability for Performance / Monitoring / Applications offloading

Platforms / Correctness / Management for Data plane programmability





Current monitoring methods are inadequate

- Not fast enough
  - Involve CPU and control planes
  - Network state changes rapidly
- Do not provide end-to-end state
  - Difficult to correlate per-element state with the actual path of a flow

Source: In-band Network Telemetry, Mukesh Hira and Naga Katta, 2015

INT : In-band Network Telemetry

- Mechanism for collecting network state in the dataplane
  - As close to realtime as possible
  - At current and future line rates
  - With a framework that can adapt over time
- Examples of network state
  - Switch ID, Ingress Port ID, Egress Port ID
  - Egress Link Utilization
  - Hop Latency
  - Egress Queue Occupancy
  - Egress Queue Congestion Status
  - ....

Source: In-band Network Telemetry, Mukesh Hira and Naga Katta, 2015

INT Header Format

4 Bytes

Ver	Flags	Instruction Count	Max Hop Count	Total Hop Count	Metadata Header
Instruction Bitmap			Reserved		
0	Most Recent INT Metadata				Metadata
0	INT Metadata				
	. .				
1	First INT Metadata				

Source: In-band Network Telemetry, Mukesh Hira and Naga Katta, 2015

INT using P4

- P4 enables flexible packet parsing and modification for INT
- P4 allows INT to adapt to
  - Any Encapsulation format
  - Any State required to be collected
  - Any feature, protocol – current and future

Source: In-band Network Telemetry, Mukesh Hira and Naga Katta, 2015

INT : P4 Code Snippet

Exact-match Table Definition

```
table int_inst {
  reads {
    int_header.instruction_mask : exact;
  }
  actions {
    int_set_header_i0;
    int_set_header_i1;
    int_set_header_i2;
    int_set_header_i3;
    .....
  }
```

Action Definitions

```
action int_set_header_i0() {
}
action int_set_header_i1() {
  int_set_header_3();
}
action int_set_header_i2() {
  int_set_header_2();
}
action int_set_header_i3() {
  int_set_header_3();
  int_set_header_2();
}
.....
```

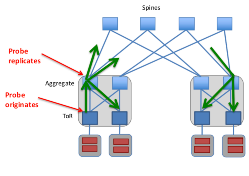
Source: In-band Network Telemetry, Mukesh Hira and Naga Katta, 2015

HULA: INT + Flowlet routing

1. Periodic INT probes
   - disseminate path utilization to switches
2. Flowlet detection and path selection
   - happens at all switches
   - hop-by-hop adaptive routing

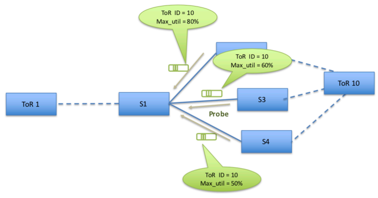
Source: In-band Network Telemetry, Mukesh Hira and Naga Katta, 2015

INT probes traverse multiple paths



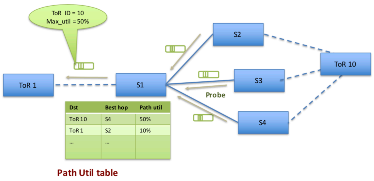
Source: In-band Network Telemetry, Mukesh Hira and Naga Katta, 2015

Probes carry path utilization



Source: In-band Network Telemetry, Mukesh Hira and Naga Katta, 2015

Probes update switch state

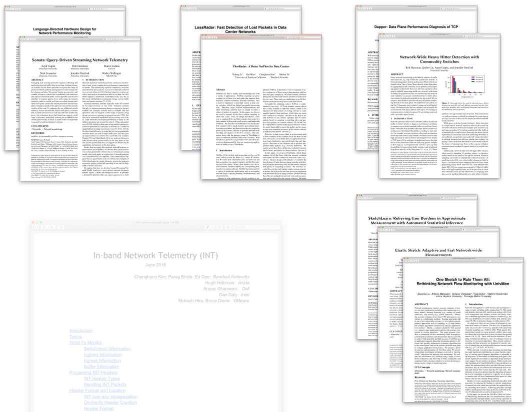


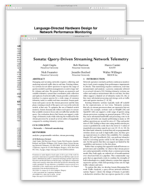
Source: In-band Network Telemetry, Mukesh Hira and Naga Katta, 2015

Summary

- INT provides real-time network state directly in the dataplane
  - Scales to arbitrarily large networks
  - Scales to current and future link speeds
  - Can adapt to any network, any encap, any application
- Knowledge of real-time network state opens up new possibilities
  - Enhanced monitoring and troubleshooting
  - Network-state aware routing
  - …

Source: In-band Network Telemetry, Mukesh Hira and Naga Katta, 2015





MARPLE [SIGCOMM'17]

SONATA [SIGCOMM'18]

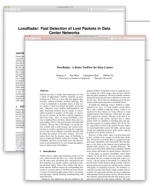
Both papers enable operators to express monitoring queries

```
result = filter(pktstream, qid == Q and switch == S
         and t_out - t_in > 1ms)
```

returns a stream of packets experiencing high queuing latencies

A compiler then compiles these queries to: switch programs + control code

The two papers differ among others in the types of queries they support



LossRadar [CoNEXT'16]

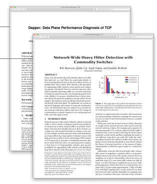
FlowRadar [NSDI'16]

Develop techniques and tools to monitor *all flows* by

- relying on in-switch data structures (Bloom Filters) and
- decoding them at the controller-level

DAPPER [SOSR'17]

Network-Wide HH [SOSR'18]



Develop P4-based detection mechanisms to

- diagnose TCP performance issue (e.g. small receiver buffers)
- heavy-hitter (e.g. port scanners, superspreader, DDoS)

Introduce techniques to make sketch-based monitoring more practical (by making sketches adaptive or "universal")

SketchLearn [SIGCOMM'18]

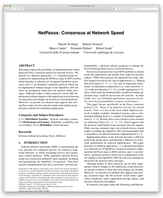
Elastic Sketch [SIGCOMM'18]

UnivMon [SIGCOMM'16]

Data plane programmability for Performance, Monitoring, Applications offloading

Platforms, Correctness, Management for Data plane programmability

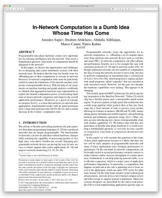
[SOSR'15]



NetPaxos: Consensus at Network Speed

Consensus at network speed

[HotNets'17]



In-Network Computation is a Dumb Idea Whose Time Has Come

In-Network Aggregation
(e.g., for MapReduce, graph analytics, ML)

[SIGCOMM'17]

SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs

Stateful layer-4 load balancers

+ NetCache [SOSP'17], NetChain [NSDI'18]

Data plane programmability	for	Performance Monitoring Applications offloading
Platforms Correctness Management	for	Data plane programmability

---

"Data-plane" programmability goes beyond switch programmability (or P4 for that matter)

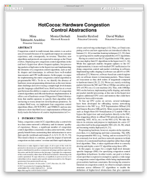
Offloading… … to FPGA-based SmartNICS

host networking

Azure Accelerated Networking: SmartNICs in the Public Cloud

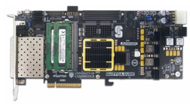
[NSDI'18]

congestion control



HotCocoa: Hardware Congestion Control Abstractions

[HotNets'17]

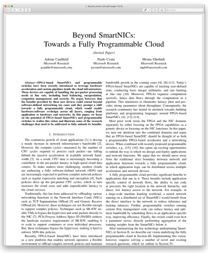


NetFPGA SUME board

Host-based programmability + SmartNICs + programmable switches = fully programmable platforms

Big question is

How to combine them best?



Beyond SmartNICs: Towards a Fully Programmable Cloud

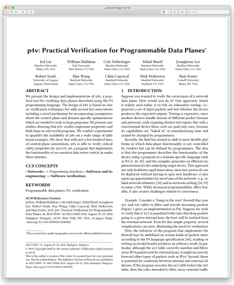
IEEE International Conference on High Performance Switching and Routing, 2018

Data plane programmability	for	Performance Monitoring Applications offloading
Platforms Correctness Management	for	Data plane programmability

---

So you've a programmable networks…

How do you make sure that it works as it should?!



p4v: Practical Verification for Programmable Data Planes

[SIGCOMM'18]

Debugging P4 programs with Vera

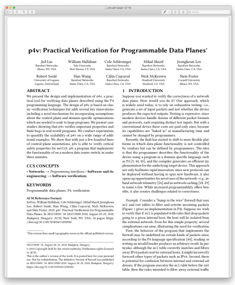
Verification of P4 Programs in Feasible Time using Assertions

[SIGCOMM'18]

[CoNEXT'18]

So you've a programmable networks…

How do you make sure that it works as it should?!



[SIGCOMM'18]

[SIGCOMM'18]

[CoNEXT'18]

Programmable routers...

(specifically, programmable data planes)

...how do they work?

Let's verify!

Give programmers language-based verification tools

P4 also used as HDL for fixed-function devices

Source: p4v, Practical Verification for Programmable Data Planes, Liu et al., 2018

P4 by example

- P4 is a low-level language → many gotchas
- Let's explore by example!
  - IPv6 router w/ access control list (ACL)

```
control ingress { apply(acl); }

table acl {
  reads { ipv6.dstAddr: lpm; }
  actions { allow; deny; }
}

action allow() {
  modify_field(std_meta.egress_spec, 1);
}

action deny() { drop(); }
```

What could *possibly* go wrong?

Source: p4v, Practical Verification for Programmable Data Planes, Liu et al., 2018

What if we didn't receive an IPv6 packet?
`ipv6` header will be **invalid**

What goes wrong
Table reads arbitrary values
→ Intended ACL policy violated

Can read values from a previous packet
→ Side channel vulnerability!

Real programs are complicated:
hard to keep validity in your head

```
control ingress { apply(acl); }

table acl {
  reads { ipv6.dstAddr: lpm; }
  actions { allow; deny; }
}

action allow() {
  modify_field(std_meta.egress_spec, 1);
}

action deny() { drop(); }
```

Property #1: header validity

Source: p4v, Practical Verification for Programmable Data Planes, Liu et al., 2018

What if `acl` table misses (no rule matches)?
Forwarding decision is unspecified

What goes wrong
Forwarding behaviour depends on hardware
- May not do what you expect!
- Code not portable

```
control ingress { apply(acl); }

table acl {
  reads { ipv6.dstAddr: lpm; }
  actions { allow; deny; }
}

action allow() {
  modify_field(std_meta.egress_spec, 1);
}

action deny() { drop(); }
```

Property #2: unambiguous forwarding

Source: p4v, Practical Verification for Programmable Data Planes, Liu et al., 2018

Types of properties

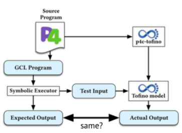
General safety
- **Header validity**
- Arithmetic-overflow checking
- Index bounds checking (header stacks, registers, meters, ...)

Architectural
- **Unambiguous forwarding**
- **Reparseability**
- **Mutual exclusion of headers**
- Correct metadata usage (e.g., read-only metadata)

Program-specific
- Custom assertions in P4 program — e.g., IPv4 `ttl` correctly decremented

Source: p4v, Practical Verification for Programmable Data Planes, Liu et al., 2018

Challenge #1: imprecise semantics



- P4 language spec doesn't give precise semantics
- Defined semantics by translation to GCL (a simple imperative language)
- Tested semantics
  - Symbolically executed GCL to generate input–output tests for several programs
  - Ran w/ Barefoot P4 compiler & Tofino simulator

Source: p4v, Practical Verification for Programmable Data Planes, Liu et al., 2018

Challenge #2: modelling the control plane

- A P4 program is just half the program
  - Table rules are not statically known
  - Populated by the control plane at run time
- Control planes are carefully programmed
  - Tables rarely take arbitrary actions
- To rule out false positives, need to model behaviour of control plane

```
table acl {
  reads {
    ipv6.dstAddr: lpm;
  }
  actions { allow; deny; }
}
```

→

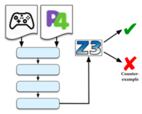
```
   ( @[ Action ] acl <hit> (allow);
     std_meta.egress_spec := 1)

[] ( @[ Action ] acl <hit> (deny);
     std_meta.egress_spec := 511)

[]   @[ Action ] acl <miss>
```

Tables translated into *unconstrained* nondeterministic choice

Source: p4v, Practical Verification for Programmable Data Planes, Liu et al., 2018

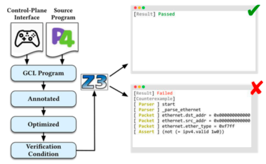
p4v overview

- **Automated** tool for verifying P4 programs
- Considers **all paths**
  - But also practical for **large programs**
- Includes basic safety properties for any program
- **Extensible** framework
  - Verify custom, program-specific properties
  - Assert-style debugging



Source: p4v, Practical Verification for Programmable Data Planes, Liu et al., 2018

p4v architecture



1. Start w/ CPI & P4 program
2. Translate to GCL
3. Auto-annotate w/ assertions
4. Standard optimizations
5. Generate formula
6. Send to Z3
7. Success or counterexample
   - Input packet
   - Program trace
   - Violated assertion

Source: p4v, Practical Verification for Programmable Data Planes, Liu et al., 2018

Data plane programmability	for	Performance Monitoring Applications offloading	
Platforms Correctness Management	for	Data plane programmability	

So you've a *verified* programmable networks…

How do you manage it?!

How do you perform planned maintenance?
now that you've state in your switches…

How do you run multiple applications in your switches?
monitoring, forwarding, load-balancing, etc.

How do you share resources amongst applications?
especially memory and # packet operations

We need an Operating System for the data plane

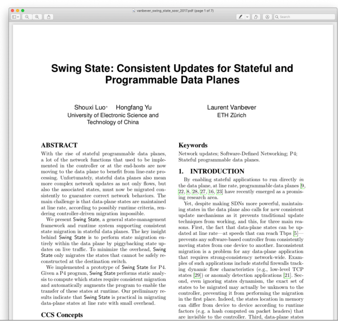
Definition
Wikipedia

An **operating system** is a system software that **manages** computer **hardware and software resources** and **provides common services** for computer programs.

Do we have that? Nope. Not yet at least.

We're working on it…

[SOSR'17]



Swing State: Consistent Updates for Stateful and Programmable Data Planes

Shouxi Luo, Hongfang Yu
University of Electronic Science and Technology of China

Laurent Vanbever
ETH Zürich

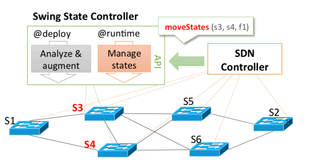
ABSTRACT

Keywords

1. INTRODUCTION

CCS Concepts

Swing State is a state management framework with 1 primitive: `moveStates`



Source: Swing State: Consistent Updates for Stateful and Programmable Data Planes
Luo et al., SOSR 2017

Advanced Topics in Communication Networks

Lectures/Exercices	Group project
~7 weeks how to program in P4	>= 7 weeks in teams of 2—3

Advanced Topics in Communication Networks

COMPLETED

Lectures/Exercices	Group project
~7 weeks	>= 7 weeks
how to program in P4	in teams of 2—3

The group project starts this week
It accounts for 50% of your final grade

The evaluation of your project will depend on your implementation, report, and presentation

---

The evaluation of your project will depend on your implementation, report, and presentation

implementation 70%	achieves the basic goals is properly documented runs…

The evaluation of your project will depend on your implementation, report, and presentation

implementation 70%	achieves the basic goals is properly documented runs…
report 15%, 10 pages max	describes the main building blocks evaluates the solution describes what each group member did

---

The evaluation of your project will depend on your implementation, report, and presentation

implementation 70%	achieves the basic goals is properly documented runs…
report 15%, 10 pages max	describes the main building blocks evaluates the solution describes what each group member did
presentation 15%, 12 min. +questions	summarizes the problem and the solution contains a *live* demo involves all group members

---

The final deadline for the project is
Wed Dec 19 at 23.59pm

This week	Select a proposal from the list (see Doodle) or send us your own proposal by email
Every week	Meet with the responsible assistant schedule a recurring slot in [10.15am; noon]
Wed Dec 19 11.59pm	Send us an archive with report, code, slides
Thu Dec 20 8.15am—	Groups presentation + course/exam debrief attendance is mandatory

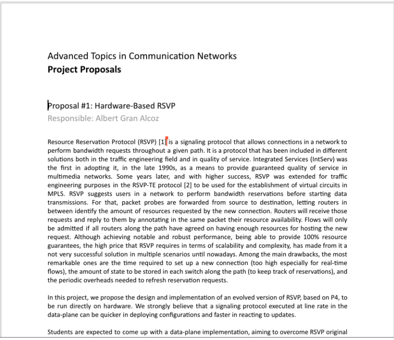
---

The project has to be done in groups of 3 students
"Matching" process for incomplete groups via Slack

Project grade is shared by each group member provided that each collaborated (roughly equally)

- Let us know in advance if that's *not* the case
- Briefly describe in the report the contribution of each group member
- Each group member should be involved in the presentation and be able to answer questions

Details about each proposal is available on our website


Advanced Topics in Communication Networks
Project Proposals

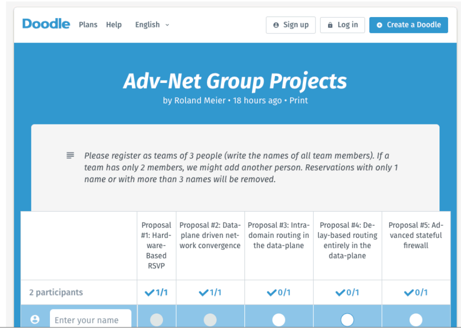
Proposal #1: Hardware-Based RSVP
Responsible: Albert Gran Alcoz

Resource Reservation Protocol (RSVP) [1] is a signaling protocol that allows connections in a network to perform bandwidth requests throughout a given path. It is a protocol that has been included in different solutions both in the traffic engineering field and in quality of service. Integrated Services (IntServ) was the first in adopting it, in the late 1990s, as a means to provide guaranteed quality of service in multimedia networks. Some years later, and with higher success, RSVP was extended for traffic engineering purposes in the RSVP-TE protocol [2] to be used for the establishment of virtual circuits in MPLS. RSVP suggests users in a network to perform bandwidth reservations before starting data transmissions. For that, packet probes are forwarded from source to destination, letting routers in between identify the amount of resources requested by the new connection. Routers will receive those requests and reply to them by annotating in the same packet their resource availability. Flows will only be admitted if all routers along the path have agreed on having enough resources for hosting the new request. Although achieving notable and robust performance, being able to provide 100% resource guarantees, the high price that RSVP requires in terms of scalability and complexity, has made from it a not very successful solution in multiple scenarios until nowadays. Among the main drawbacks, the most remarkable ones are the time required to set up a new connection (too high especially for real-time flows), the amount of state to be stored in each switch along the path (to keep track of reservations), and the periodic overheads needed to refresh reservation requests.

In this project, we propose the design and implementation of an evolved version of RSVP, based on P4, to be run directly on hardware. We strongly believe that a signaling protocol executed at line rate in the data-plane can be quicker in deploying configurations and faster in reacting to updates.

Students are expected to come up with a data-plane implementation, aiming to overcome RSVP original

Register your proposal (one per group)
from Friday 3pm until Sunday 11.59pm



If you want to propose your own project,
send me an email describing it by Friday (Nov 2) 3pm

lvanbever@ethz.ch

Quick overview of the proposals

Albert Thomas Roland Alexander Edgar

Quick overview of the proposals

Albert Thomas Roland Alexander Edgar

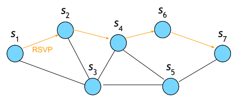
Proposal #1
Hardware-Based RSVP

Bandwidth reservations throughout a given path:

- Quality of Service guarantees (IntServ)
- Establishment of virtual circuits (MPLS)

Exclusive data plane implementation:

- Personalized headers
- Header stacks
- Registers
- Bloom filters

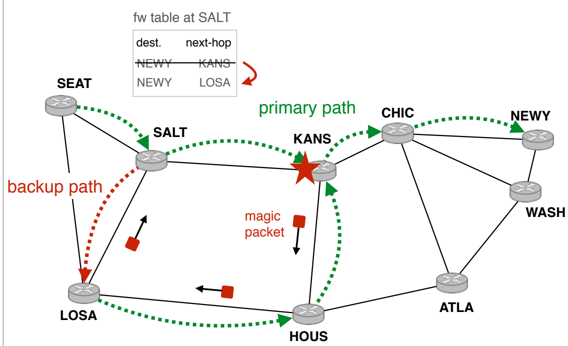


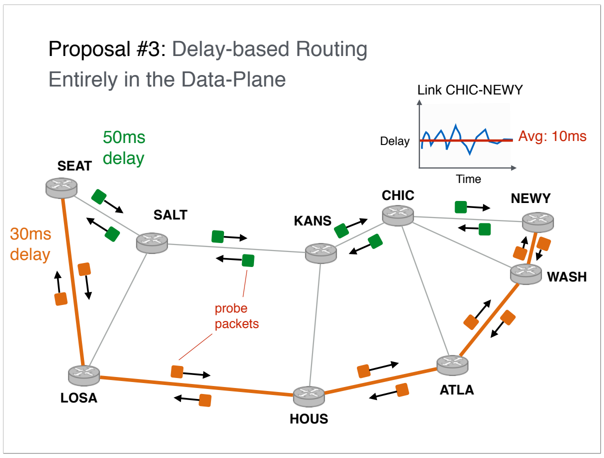
Faster and more scalable than traditionally

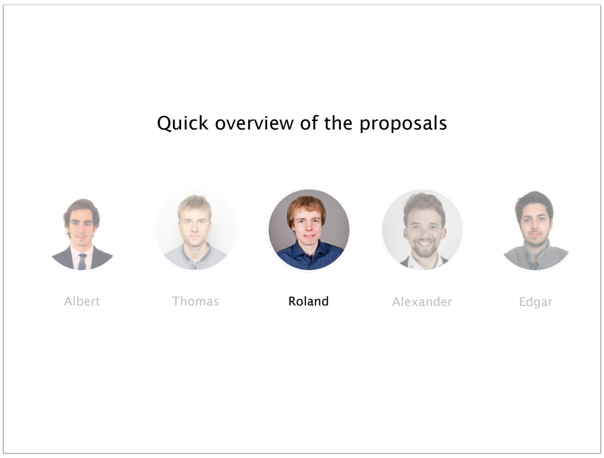
Quick overview of the proposals

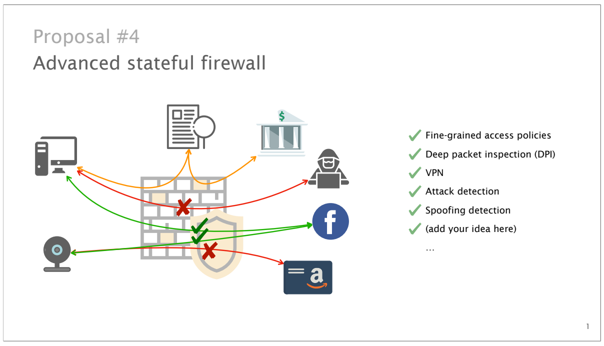
Albert **Thomas** Roland Alexander Edgar

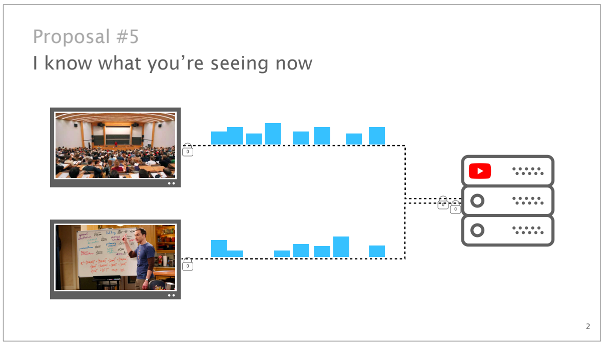
Proposal #2: Data-plane Driven Network Convergence

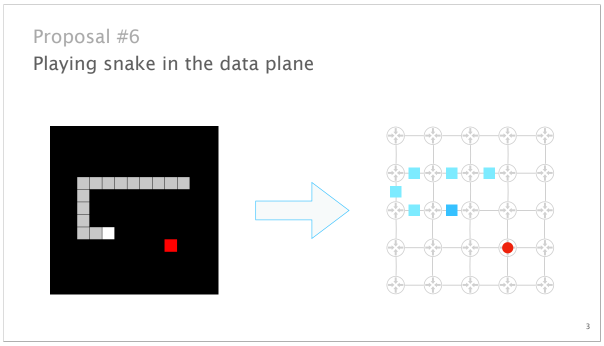


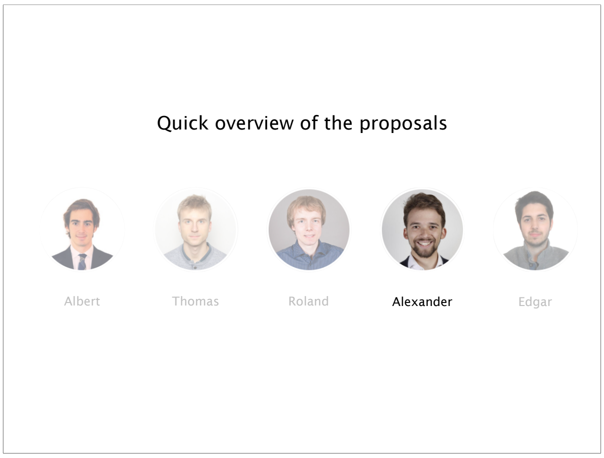

Proposal #3: Delay-based Routing
Entirely in the Data-Plane
Link CHIC-NEWY
Delay
Time
Avg: 10ms
50ms delay
30ms delay
SEAT
SALT
KANS
CHIC
NEWY
WASH
LOSA
HOUS
ATLA
probe packets

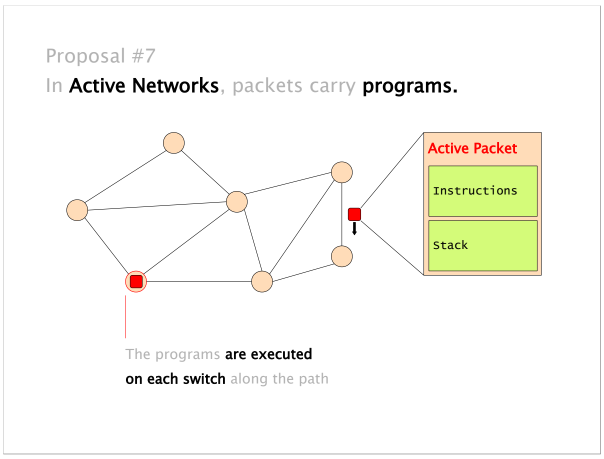

Quick overview of the proposals
Albert
Thomas
Roland
Alexander
Edgar

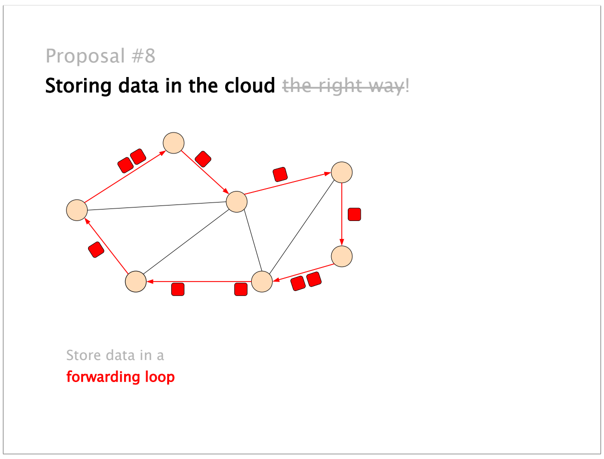

Proposal #4
Advanced stateful firewall
Fine-grained access policies
Deep packet inspection (DPI)
VPN
Attack detection
Spoofing detection
(add your idea here)
...
1


Proposal #5
I know what you're seeing now
2

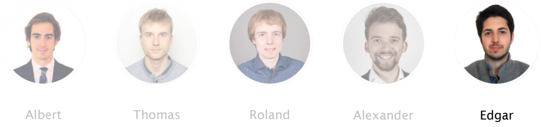

Proposal #6
Playing snake in the data plane
3


Quick overview of the proposals
Albert
Thomas
Roland
Alexander
Edgar


Proposal #7
In Active Networks, packets carry programs.
Active Packet
Instructions
Stack
The programs are executed
on each switch along the path

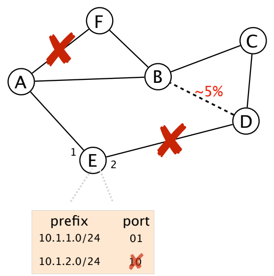

Proposal #8
Storing data in the cloud the right way!
Store data in a
forwarding loop

Quick overview of the proposals



Proposal #9

Data Plane Failure Detection



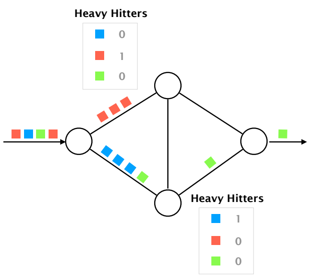
Detect local and remote link failures (A–C)

Detect random packet drops (B–C)

Detect corrupted table entries (E)

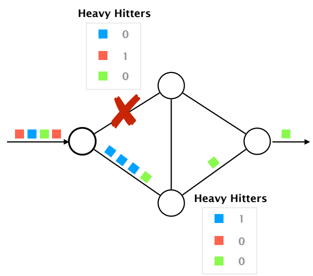
Proposal #10

Stateful Application Migration



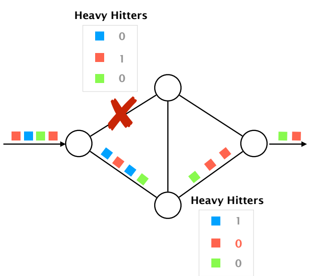
Proposal #10

Stateful Application Migration



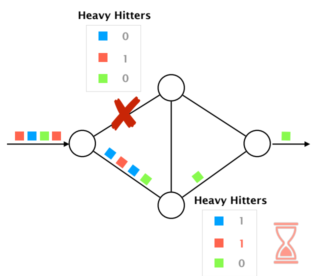
Proposal #10

Stateful Application Migration



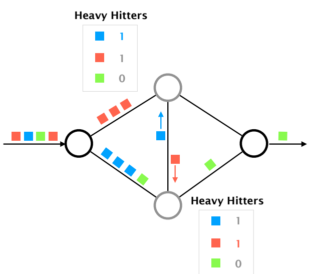
Proposal #10

Stateful Application Migration



Proposal #10

Stateful Application Migration



Proposal #11

P4 Switch

Management and Configuration API

Control Plane

Basic Features	Advanced Features
l2 forwarding, learning, multicast	Spanning Tree Protocol
ipv4, ipv6, l3 multicast	netflow, sFlow or similar
ECMP, Weighted ECMP	VXLAN, MPLS, Gre
ICMP	DHCP Server
ARP	DNS Cache
ECN	Simple Firewall
Simple QoS	NAT

Data Plane