Software-Defined Networking

Paul Grubbs

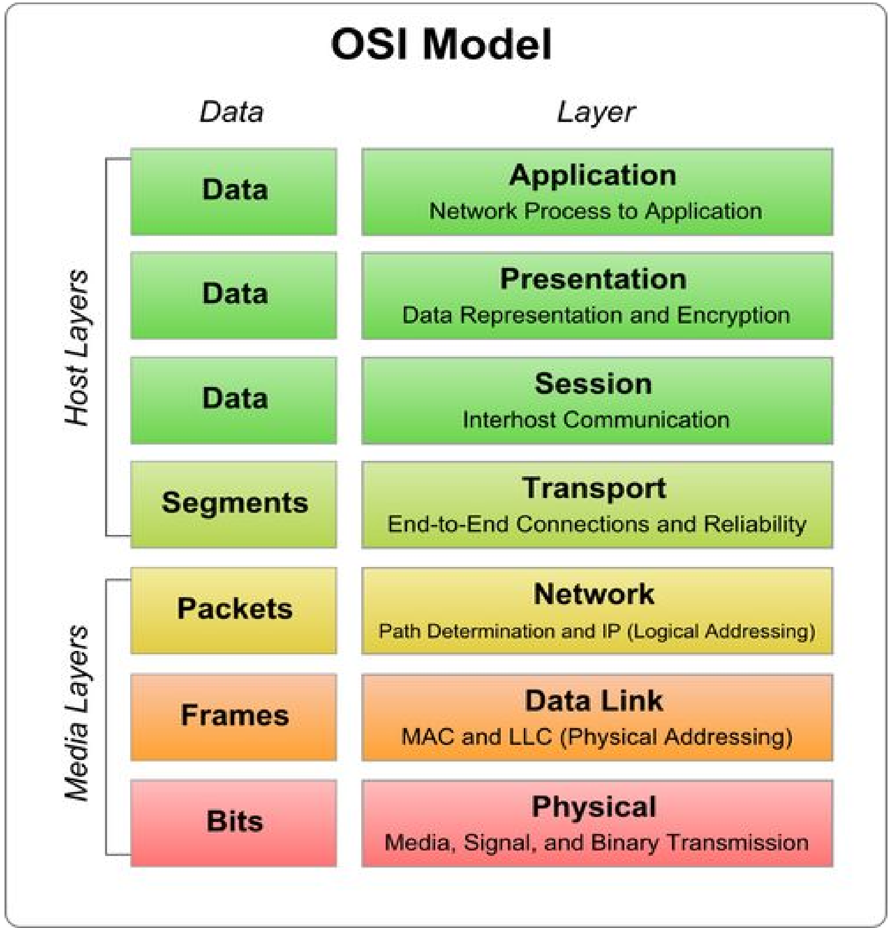
Portions of this talk taken from:
https://www.cs.rutgers.edu/~badri/552dir/papers/intro/nick09.pdf
http://dl.acm.org/citation.cfm?id=2602219
http://frenetic-lang.org/publications/frenetic-presto10-slides.pdf
http://frenetic-lang.org/publications/frenetic-icfp11-slides.pdf
Mohamed Ismail's talk from 6410 fall '13

What papers will we be discussing?

OpenFlow: Enabling Innovation in Campus Networks
Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner

Frenetic: A High-Level Language for OpenFlow Networks
Nate Foster, Rob Harrison, Matthew L. Meola, Michael J. Freedman, Jennifer Rexford, and David Walker.

Obligatory review of OSI model



OSI Model

	Data	*Layer*
Host Layers	**Data**	**Application** Network Process to Application
	Data	**Presentation** Data Representation and Encryption
	Data	**Session** Interhost Communication
	Segments	**Transport** End-to-End Connections and Reliability
Media Layers	**Packets**	**Network** Path Determination and IP (Logical Addressing)
	Frames	**Data Link** MAC and LLC (Physical Addressing)
	Bits	**Physical** Media, Signal, and Binary Transmission

Network devices

- Layer 2 (“data link”) forwarding
- Different machines on the same LAN communicate via a switch
- Uses MAC addresses

switch

- Layer 3 (“network”) routing
- Connects LANs together to form a WAN
- Uses IP addresses

router

The joke’s on us: “switch” and “router” are used almost interchangeably!

Control Plane

- Which packets go where?
- Routing (flow) tables

Data Plane

- Get packets to the right place
- Uses flow table rules defined by control plane to route packets

Conventional networking

- Code+administration+hardware fused together in networking
- Control plane + data plane on same device

Networking researchers:

- Build new protocol
- Test at small scales
- Wait a decade for IETF standardization
- Deploy

Industry networking:

- Cisco hardware
- Cisco operating system
- Works best with other Cisco hardware.
- To change something, need somebody certified with Cisco to use the Cisco UI.
- How to scale to increase in traffic? Buy more Cisco! Hire more CCNAs!

What is software-defined networking (SDN)?

- Abstracts control from routing functionality
- Programmability of the control plane
  - Provides abstractions for device functions

History of SDN

- Active networking (mid 90s to early 00s)
    - Give programming interface that exposes network resources on individual devices
    - Ability to apply more fine-grained controls to specific packet streams
    - “[A]nathema to many in the internet community” who valued simplicity
- Control and data plane separation (early 00s to late 00s)
    - Standardized interfaces between the two
        - ForCES (Forwarding and Control Element Separation) IETF standard
    - Centralize management of control plane across different devices
        - Path Computation Element IETF standard
    - Challenge: distributed state management
- Around 2008, along comes….

OpenFlow

Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner

- SIGCOMM CCR 2008
- Open Networking foundation manages OpenFlow protocol
- OpenFlow protocol supported by most major router vendors, including Cisco, IBM, Juniper, Brocade, and many others

From Mohamed's slides

Authors

- Nick McKeown
  - '95 PhD UC Berkeley
  - Co-founded Nicira Networks, ONF
  - Faculty at Stanford

- Tom Anderson
  - '91 PhD Univ. of Wash.
  - UC Berkeley '91-'97
  - Faculty at Univ. of Wash.

- Hari Balakrishnan
  - '98 PhD UC Berkeley
  - Faculty at MIT

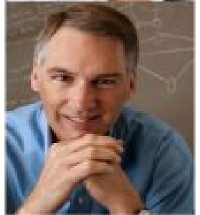
- Guru Parulkar
  - '87 PhD Univ. of Deliware
  - Many network-related startups
  - Executive director of Clean Slate Internet Design Program

- Larry Peterson
  - '85 PhD Purdue University
  - GENI project chair
  - Faculty at Princeton

- Jennifer Rexford
  - '96 PhD Univ. of Mich.
  - AT&T Labs '96-'05
  - Broader Gateway Protocol
  - Faculty at Princeton

- Scott Shenker
  - '83 PhD Univ. of Chig.
  - XEROX Parc
  - Co-founder of Nicira Networks, ONF
  - Faculty at Berkeley

- Jonathan Turner
  - Faculty at Washington University in St. Louis



Motivation

- Networking researchers need to do experiments
  - Small-scale experiments not accurate assessment of performance in real settings
- Explicitly changing routing tables in every router is very complex
  - Each vendor has their own language, hardware, etc.
- Why don’t we just ask the vendors to provide an open, standard platform for research?
  - Vendors jealously guard internal functions of router
  - No standard platform for experiments

Motivating questions

- “How will researchers control a portion of their local network in a way that does not disrupt others who depend on it?”
- “[W]hat functionality is needed in network switches to enable experiments?”

Flows

What is a flow?

- packets that have the same src and destination
  - (e.g. same src IP address and port, dest IP address and port, and protocol)
- “Paul’s traffic”
- “Traffic from Stanford”
- “HTTP traffic”

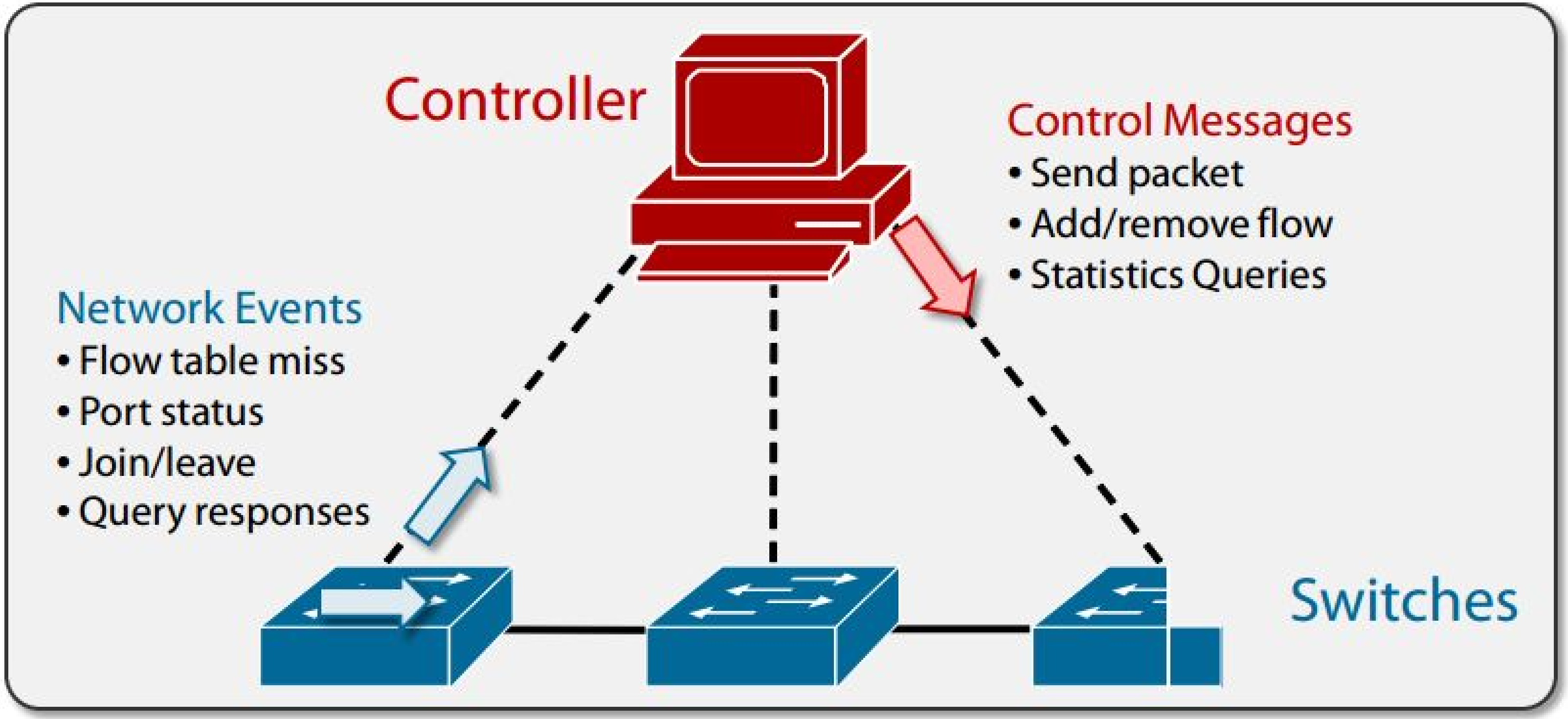
What do we want to do with a flow?

- Route flow
- Isolate flow
- Delete flow
- Compute statistics on flow

How do we implement a flow?

Implementing a flow?

- Use common functionality of switch/router *flow tables*
- OpenFlow is an open protocol to program the flow table
  - Crucially, does not require knowledge of inner workings of device
  - Vendor-friendly
- Three main parts:
  - Flow table
  - Secure channel to *controller*
  - OpenFlow protocol (standard connection between controller and device)



OpenFlow Switch Flow Table

The controller: it controls things

- Communicate with individual devices using OpenFlow
  - Statistics queries (e.g. “How many bytes from www.google.com?”)
- Devices ask controller for advice on previously-unseen packets
  - Controller can choose to install a new entry in the flow table in response to events

OpenFlow vs. IX/Arrakis?

- IX and Arrakis focus on making server networking fast and scalable for applications which need very low latency (e.g. object caches)
    - Modify existing kernels to move network stack to user level
    - Primarily general-purpose hardware
- OpenFlow focuses on layer below application
    - Vendor-specific hardware, little/no internal details
    - Don’t modify software or hardware
    - Instead expose standard way to program common behaviors in different systems
- In common: abstract “control plane” from “data plane” (kind of)
    - Both “virtualize” underlying network device

Two ways to use OpenFlow

Dedicated OpenFlow switches

or

OpenFlow-enabled switches

Dedicated OpenFlow switches

- "Dumb" datapath element that implements OpenFlow
- Three basic actions it must perform:
  - Forward packets in flow to port(s)
  - Encapsulate and forward packets to controller
  - Deny or drop packets in flow

Dedicated OpenFlow switches

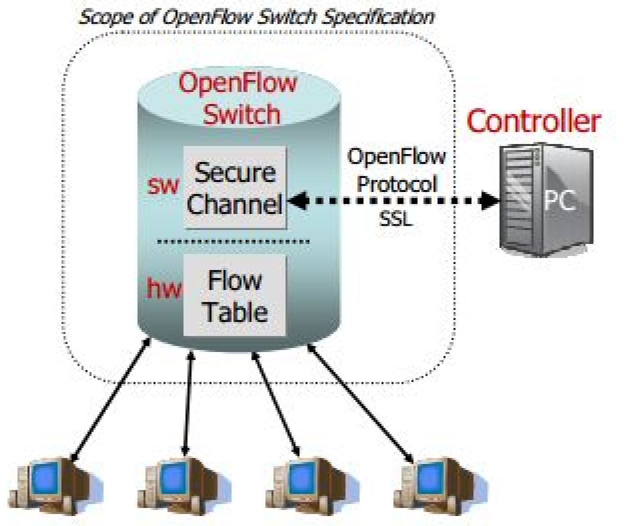


Figure 1: Idealized OpenFlow Switch. The Flow Table is controlled by a remote controller via the Secure Channel.

OpenFlow-enabled switches and routers

- Vendors implement OpenFlow API on existing devices
- Requirement: Isolate research traffic from normal flows
    - Either add a fourth action to tell device to send packet through normal flow, or
    - Define separate VLANs

OpenFlow-enabled switches and routers

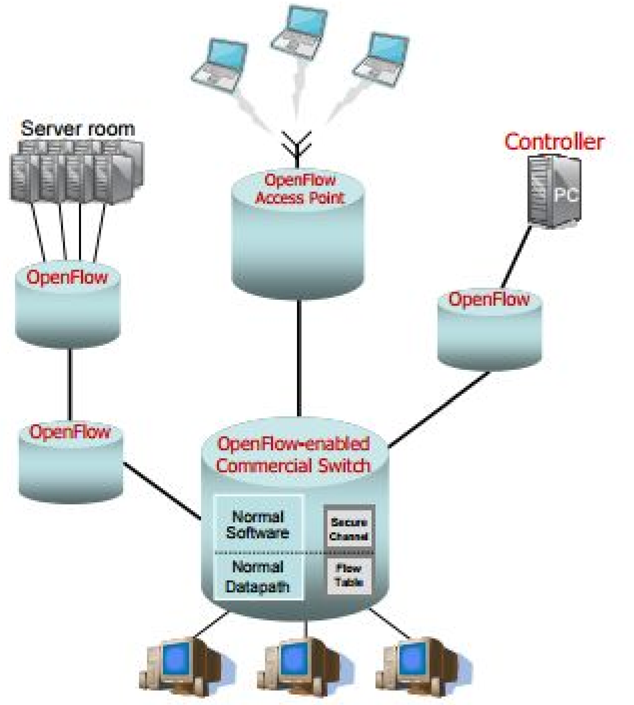


Figure 2: Example of a network of OpenFlow-enabled commercial switches and routers.

Programming OpenFlow: NOX

- NOX: Towards an operating system for networks.
  - Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, Scott Shenker
- OpenFlow is like a device driver, NOX is like an operating system. (More on that in a bit.)

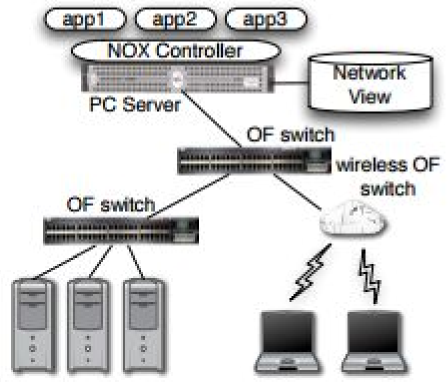


Figure 1: Components of a NOX-based network: OpenFlow (OF) switches, a server running a NOX controller process and a database containing the network view.

Thoughts/Questions?

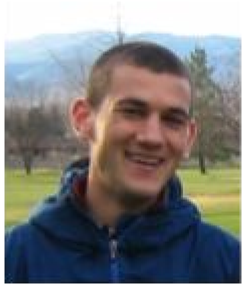
- They didn’t really evaluate OpenFlow at all. Do you think this hurt their “pitch”?
- Do you believe their claim that getting vendors to cooperate is too difficult?
- Is putting the controller in the routing path too slow? Are there other ways to do it?
- What did you like or dislike about this paper?

Frenetic: A High-level Language for OpenFlow Networks

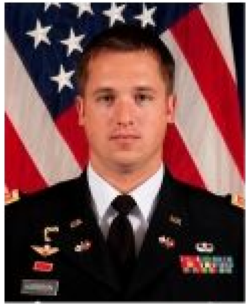
Nate Foster, Rob Harrison, Matthew L. Meola, Michael J. Freedman, Jennifer Rexford, David Walker

From Mohamed's slides

- Nate Foster
  - '09 PhD Upenn
  - Faculty at Cornell



- Rob Harrison
  - '11 Masters Princeton
  - Westpoint



- Matthew L. Meola

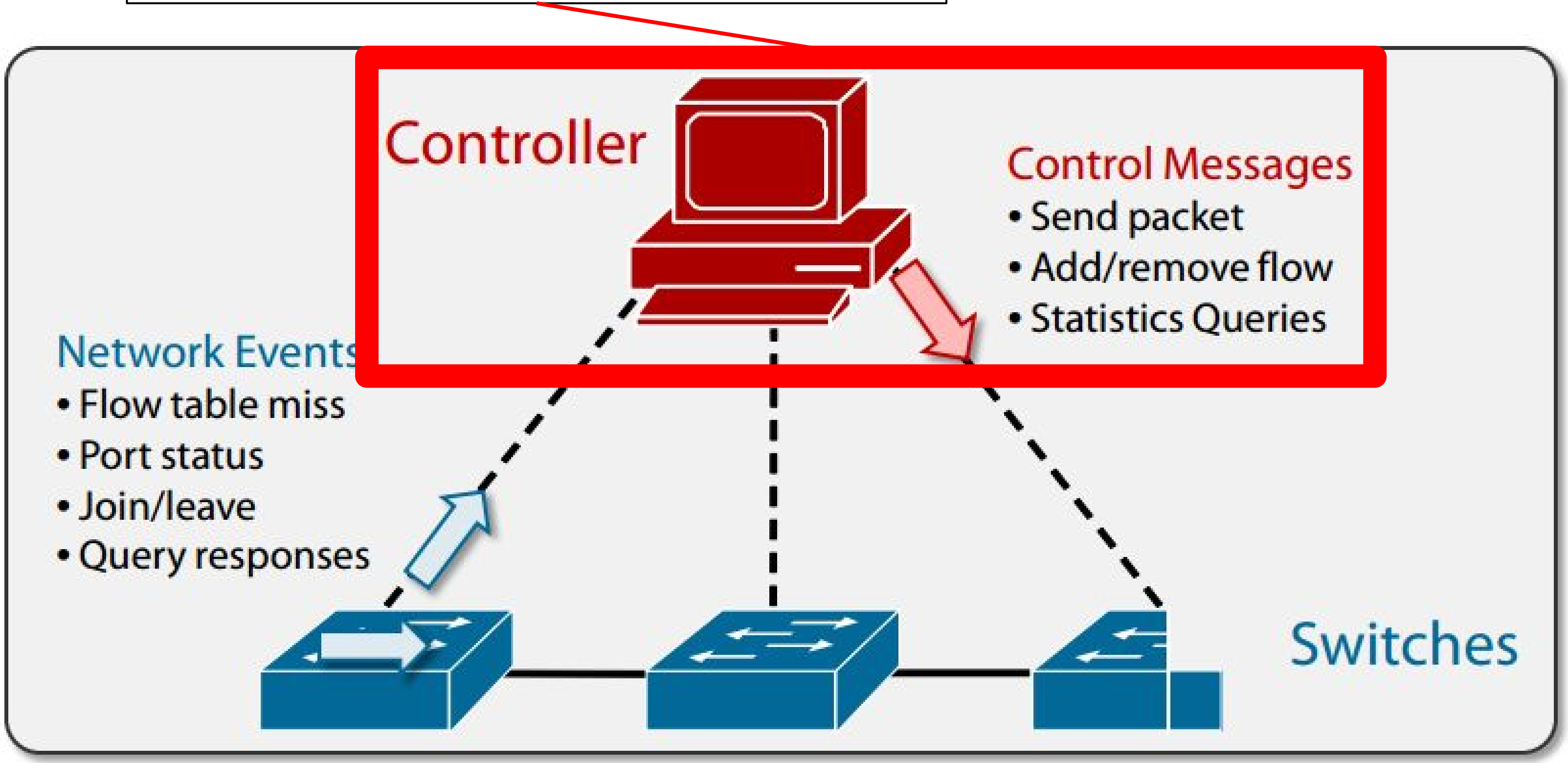
  MA, Princeton
  Stroz Friedberg LLC

- Michael J. Freedman
  - PhD NYU
  - CoralCDN
  - Faculty at Princeton

- Jennifer Rexford
  - '96 PhD Univ. of Mich.
  - AT&T Labs '96-'05
  - Broader Gateway Protocol
  - Faculty at Princeton

- David Walker
  - '01 PhD Cornell (Morriset
  - Faculty at Princeton

Frenetic deals with this part



OpenFlow Switch Flow Table

Programming OpenFlow/NOX is hard.

- Needs low-level understanding of routers and switches
- Changes to flow tables do not compose (!)
- Programmers need to reason about asynchronous behavior

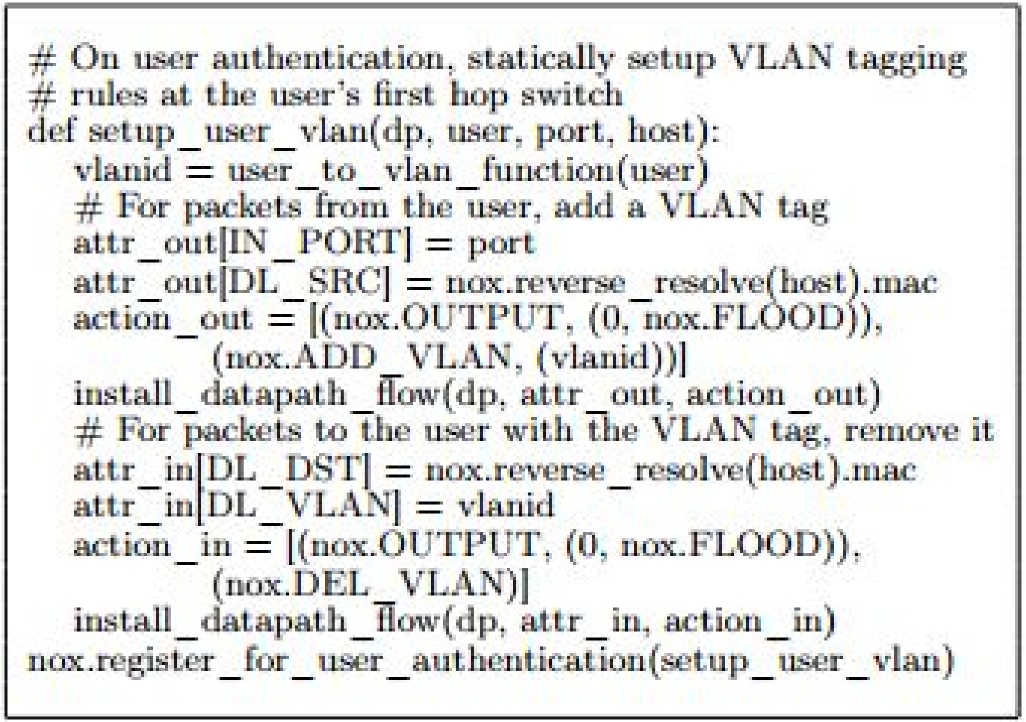
NOX: An OpenFlow platform

- Platform for programming OpenFlow
- Paper published to SIGCOMM CCR alongside OpenFlow
- C++ API on standard Linux

“NOX: Towards an Operating System for Networks”
Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, Scott Shenker

Example NOX program

?!?!?



```
# On user authentication, statically setup VLAN tagging
# rules at the user's first hop switch
def setup_user_vlan(dp, user, port, host):
    vlanid = user_to_vlan_function(user)
    # For packets from the user, add a VLAN tag
    attr_out[IN_PORT] = port
    attr_out[DL_SRC] = nox.reverse_resolve(host).mac
    action_out = [(nox.OUTPUT, (0, nox.FLOOD)),
               (nox.ADD_VLAN, (vlanid))]
    install_datapath_flow(dp, attr_out, action_out)
    # For packets to the user with the VLAN tag, remove it
    attr_in[DL_DST] = nox.reverse_resolve(host).mac
    attr_in[DL_VLAN] = vlanid
    action_in = [(nox.OUTPUT, (0, nox.FLOOD)),
               (nox.DEL_VLAN)]
    install_datapath_flow(dp, attr_in, action_in)
nox.register_for_user_authentication(setup_user_vlan)
```

?!?!?

Example w/o Frenetic

```
def repeater(switch):
    p1 = {IN_PORT:1}
    p2 = {IN_PORT:2}
    a1 = [output(2)]
    a2 = [output(1)]
    install(switch, p1, a1, DEFAULT)
    install(switch, p2, a2, DEFAULT)


def monitor(switch):
    p = {IN_PORT:2,TP_SRC:80}
    install(switch, p, [], DEFAULT)
    query_stats(switch, p)
```

```
def repeater_monitor(switch):
    p1 = {IN_PORT:1}
    p2 = {IN_PORT:2}
    p2web = {IN_PORT:2,TP_SRC:80}
    a1 = [output(2)]
    a2 = [output(1)]
    install(switch, p1, a1, DEFAULT)
    install(switch, p2, a2, DEFAULT)
    install(switch, p2web, a2, HIGH)
    query_stats(switch, p2web)
```

Monitor rule is *more specific* than repeater rule - *must* come first!!!!

FreNETic (get it?)

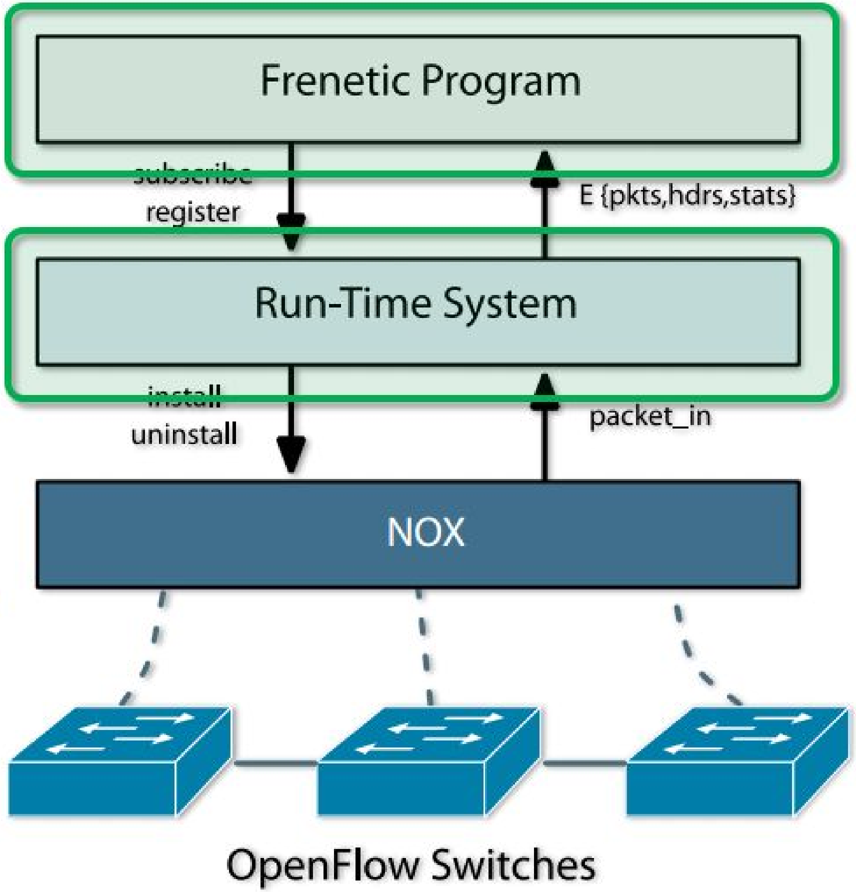
- Built on top of NOX/OpenFlow controller
- High-level language using *functional reactive programming* paradigm
- Implements common features needed for flows
- Compositionality is guaranteed by language and runtime
- Asynchronous behavior is abstracted from programmer, handled by runtime

A High-level Language

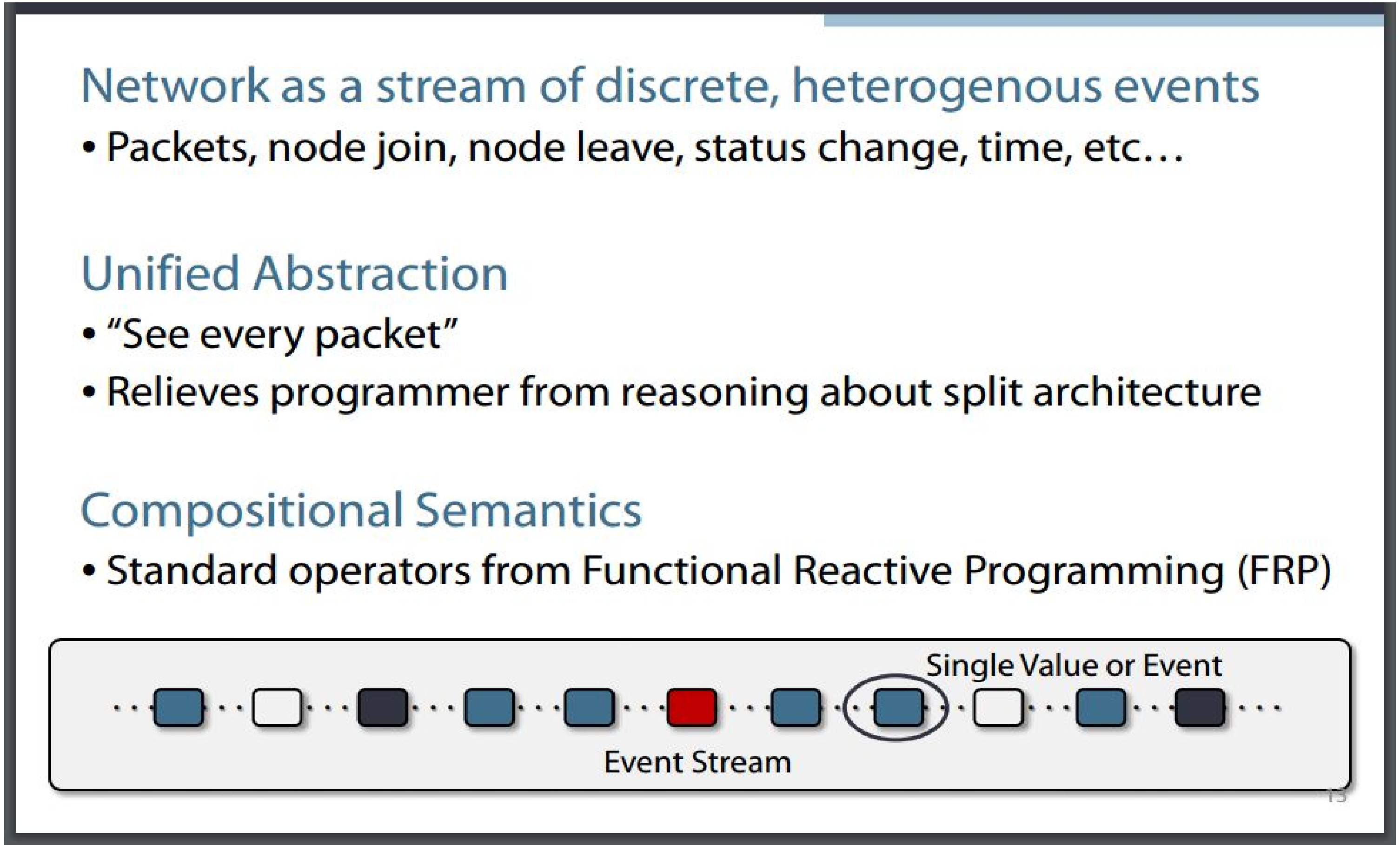
- High-level patterns to describe flows
- Unified abstraction
- Composition

A Run-time System

- Handles module interactions
- Deals with asynchronous behavior



Core abstraction: streams



Network as a stream of discrete, heterogenous events

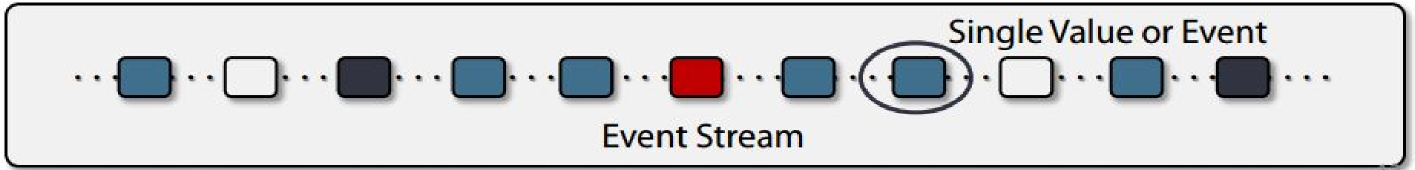
- Packets, node join, node leave, status change, time, etc…

Unified Abstraction

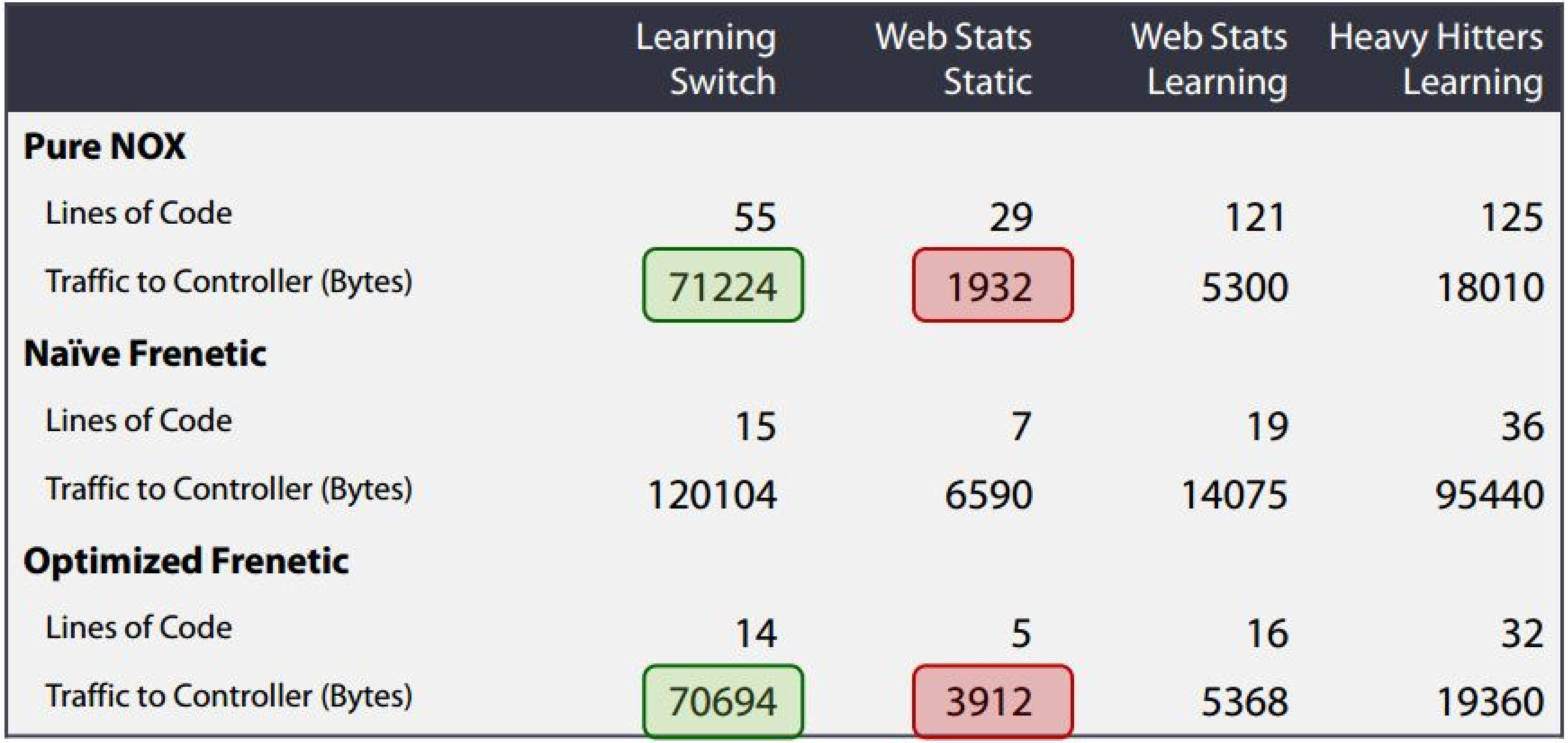
- "See every packet"
- Relieves programmer from reasoning about split architecture

Compositional Semantics

- Standard operators from Functional Reactive Programming (FRP)



Performance compared to NOX



	Learning Switch	Web Stats Static	Web Stats Learning	Heavy Hitters Learning
Pure NOX				
Lines of Code	55	29	121	125
Traffic to Controller (Bytes)	71224	1932	5300	18010
Naïve Frenetic				
Lines of Code	15	7	19	36
Traffic to Controller (Bytes)	120104	6590	14075	95440
Optimized Frenetic				
Lines of Code	14	5	16	32
Traffic to Controller (Bytes)	70694	3912	5368	19360

Thoughts/Questions?

- Is a custom language really easier than NOX’s approach?
  - Does it lead to fewer bugs and better programs overall?
- With Frenetic and NetKAT, the evolution of programmable networks looks pretty familiar
  - Evolving pretty much how regular computers and languages did (hardware->OSs->applications)
  - Can this give us any insight into the next few years of research in this space?
    - What are the major pitfalls to avoid?
  - What about the future of commercial programmable networks?
- What did you like or dislike about this paper?

Happy Thanksgiving!!!!