Dr. Nick Feamster
Professor

Software Defined Networking

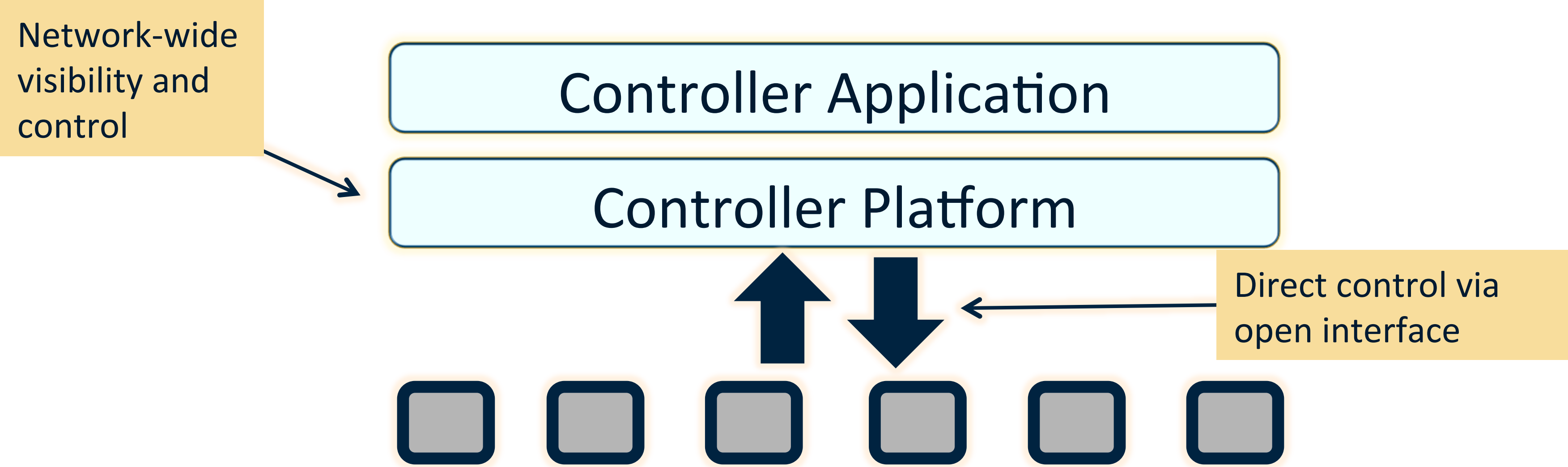
In this course, you will learn about software defined networking and how it is changing the way communications networks are managed, maintained, and secured.

This Module: Programming SDNs

- Lessons
  - Motivation for Programming SDNs
  - **Programming Languages for SDNs**
  - Composing SDN Control
    - Pyretic
  - Kinetic: Event-Driven SDN
- Programming Assignment
- Quiz

Some slide material courtesy of Jennifer Rexford

Programming SDNs



Network-wide visibility and control

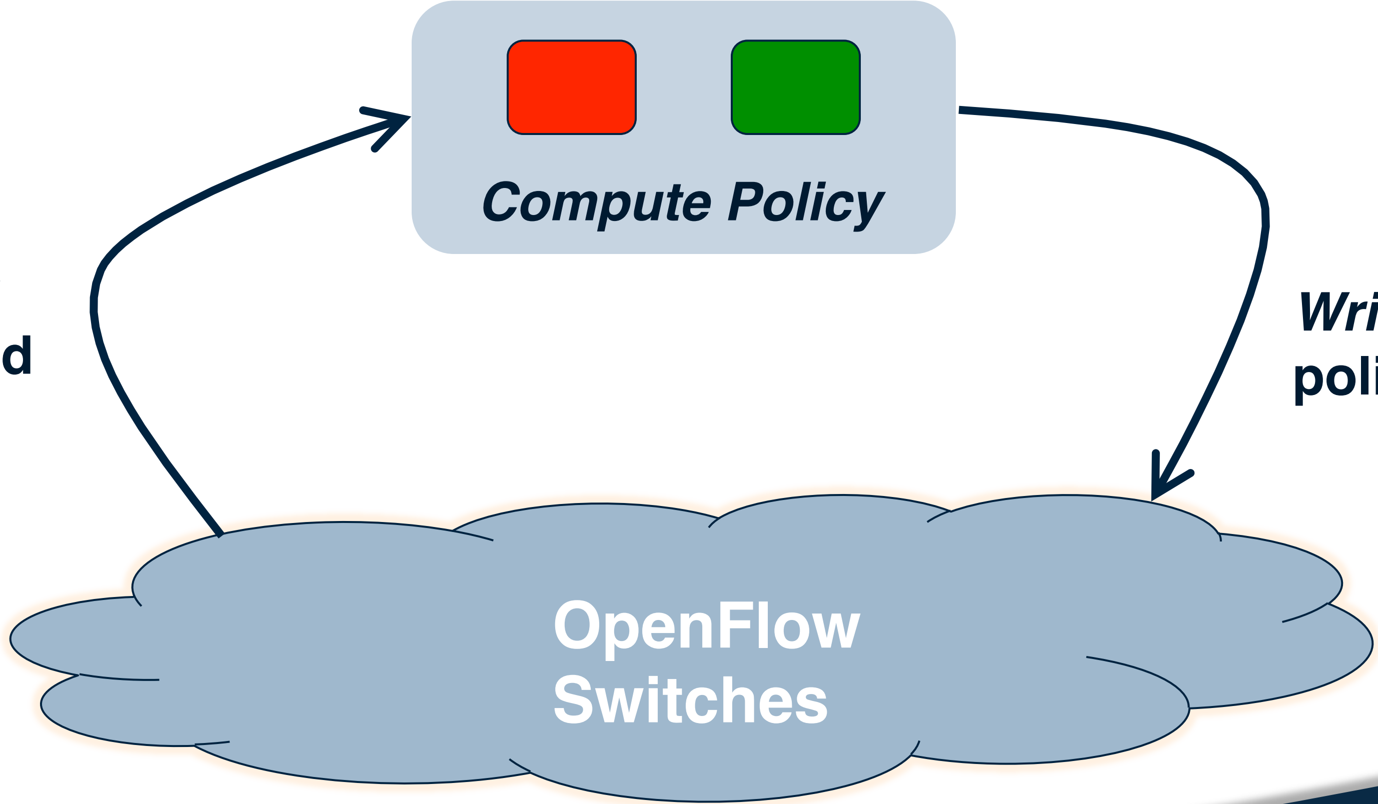
Controller Application

Controller Platform

Direct control via open interface

Southbound APIs such as OpenFlow are tied to the underlying hardware

SDN Programming: Three Steps



Compute Policy

Read/ Monitor state and events

Write policy

OpenFlow Switches

Reading State: Multiple Rules

- Traffic counters
  - Each rule counts bytes and packets
  - Controller can poll the counters
- Multiple rules
  - E.g., Web server traffic except for source 1.2.3.4

1. srcip = 1.2.3.4, srcport = 80
2. srcport = 80

- **Solution:** predicates
  - E.g., (srcip != 1.2.3.4) && (srcport == 80)
  - Run-time system
    translates into switch patterns

Reading State: Unfolding Rules

- Limited number of rules
  - Switches have limited space for rules
  - Cannot install all possible patterns
- Must add new rules as traffic arrives
  - E.g., histogram of traffic by IP address
  - … packet arrives from source 5.6.7.8

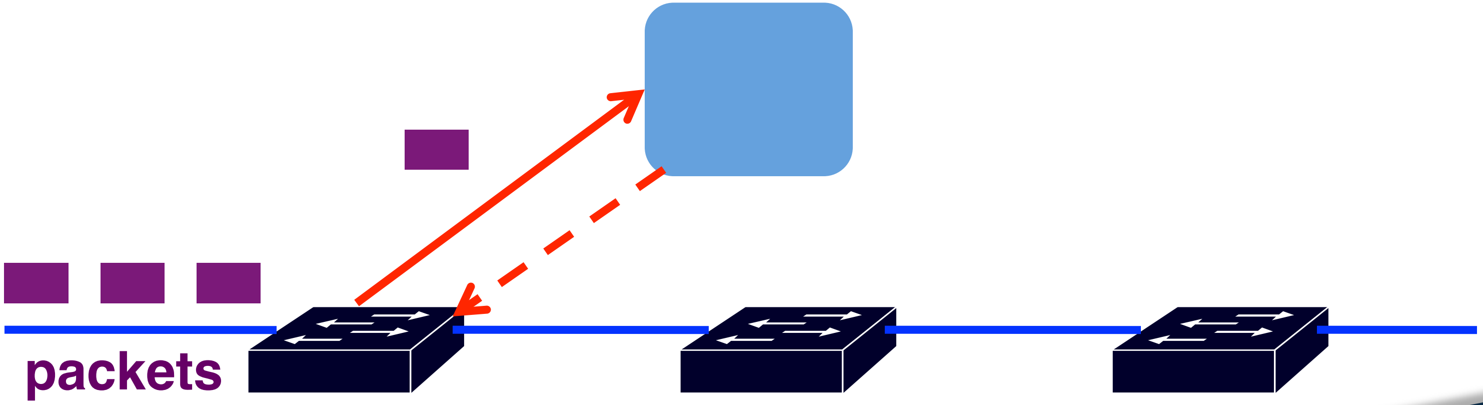
1. srcip = 1.2.3.4

1. srcip = 1.2.3.4
2. srcip = 5.6.7.8

- **Solution:** dynamic unfolding
  - Programmer specifies GroupBy(srcip)
  - Run-time system dynamically adds rules

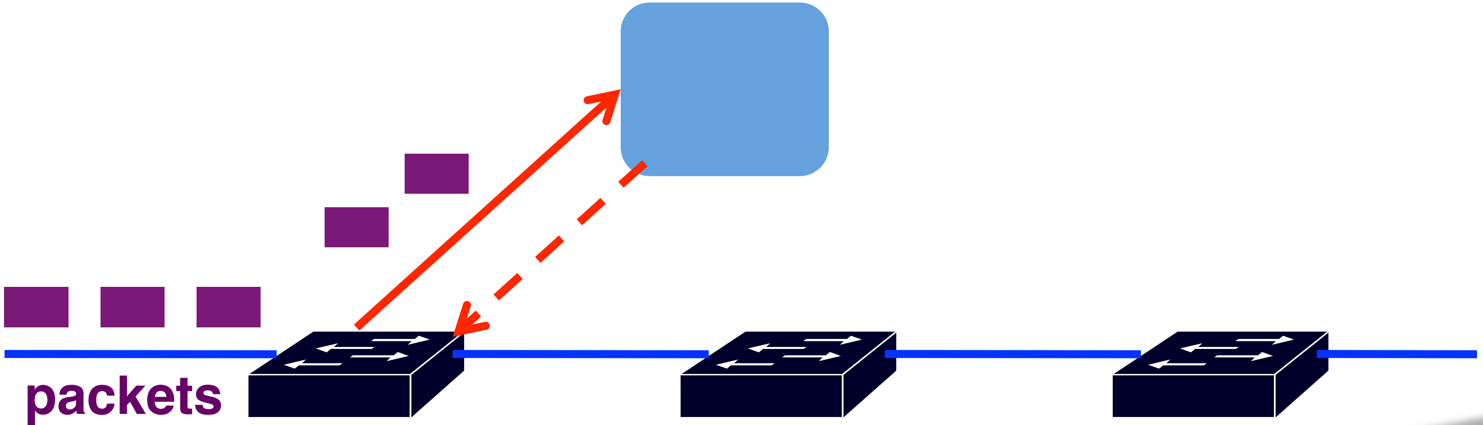
Reading State: Extra Unexpected Events

- Common programming idiom
  - First packet goes to the controller
  - Controller application installs rules



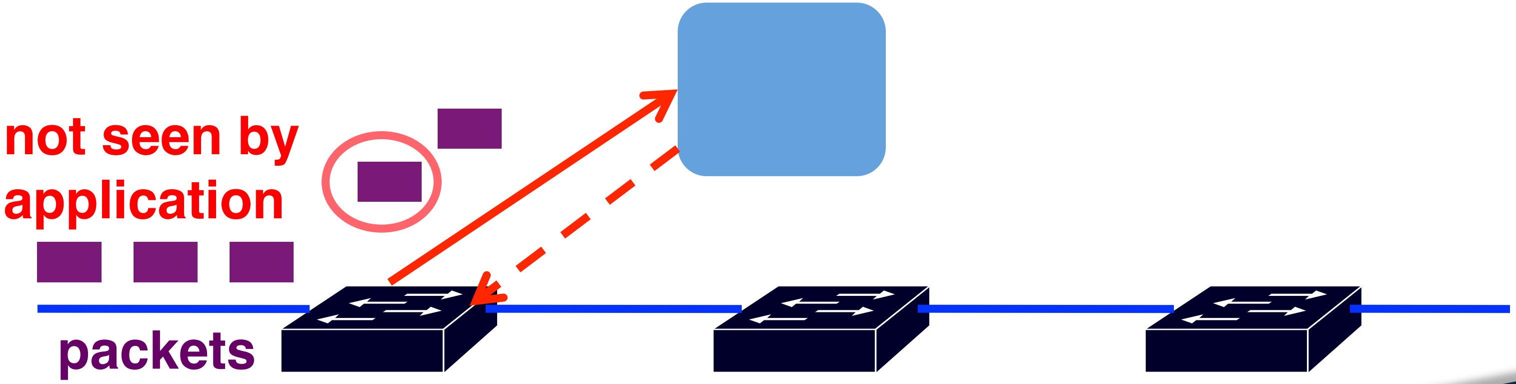
Reading State: Extra Unexpected Events

- More packets arrive before rules installed?
  - Multiple packets reach the controller



Reading State: Extra Unexpected Events

- **Solution:** suppress extra events
  - Programmer specifies “Limit(1)”
  - Run-time system hides the extra events



Frenetic: SQL-Like Query Language

- **Get what you ask for**
  - Nothing more, nothing less
- SQL-like query language
  - Familiar abstraction
  - Returns a stream
  - Intuitive cost model
- Minimize controller overhead
  - Filter using high-level patterns
  - Limit the # of values returned
  - Aggregate by #/size of packets

Traffic Monitoring

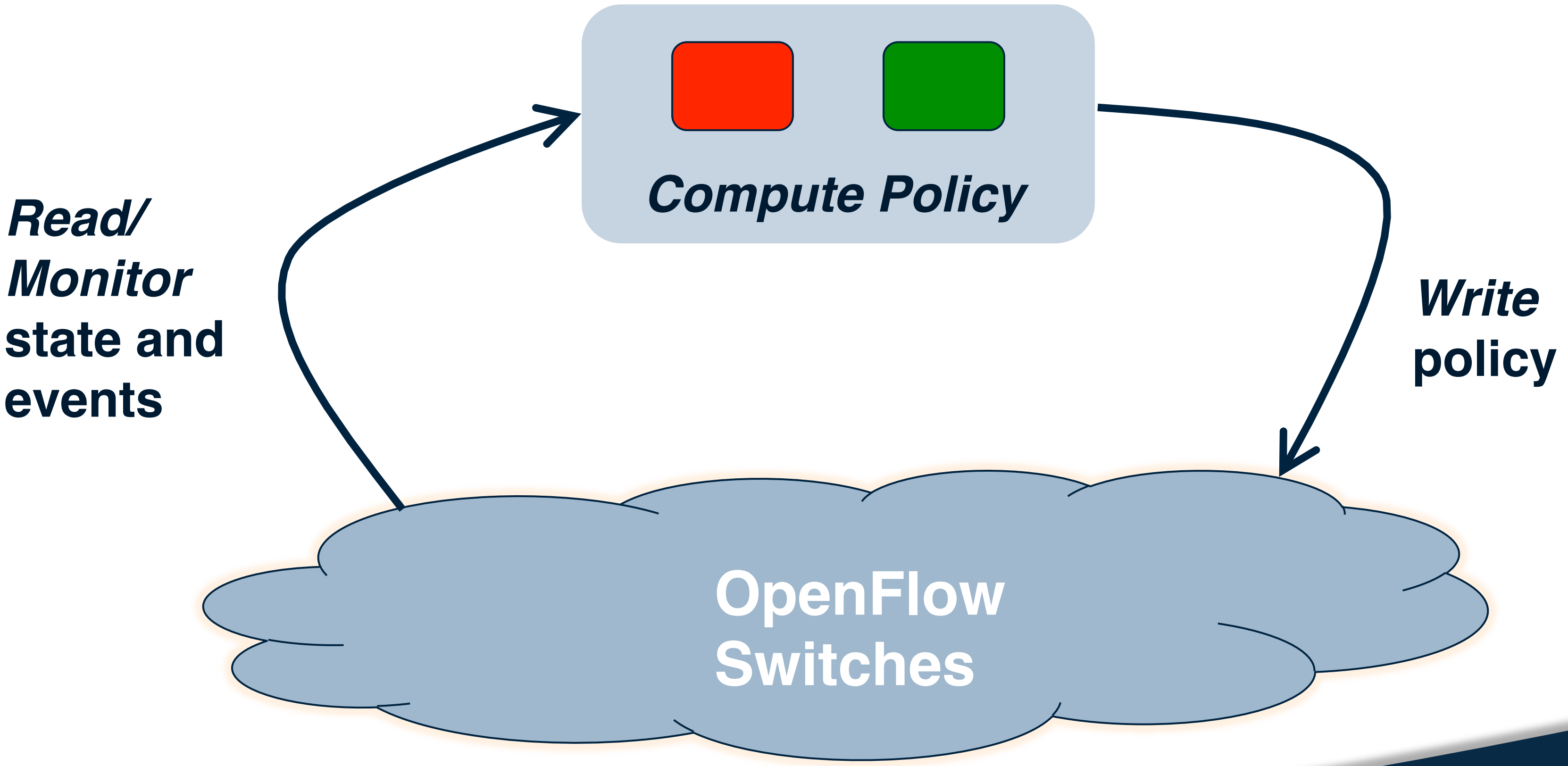
```
Select(bytes) *
Where(in:2 & srcport:80) *
GroupBy([dstmac]) *
Every(60)
```

Learning Host Location

```
Select(packets) *
GroupBy([srcmac]) *
SplitWhen([inport]) *
Limit(1)
```

Foster, Nate, et al. "Frenetic: A network programming language." *ACM SIGPLAN Notices* 46.9 (2011): 279-291.

SDN Programming: Three Steps



Compute Policy

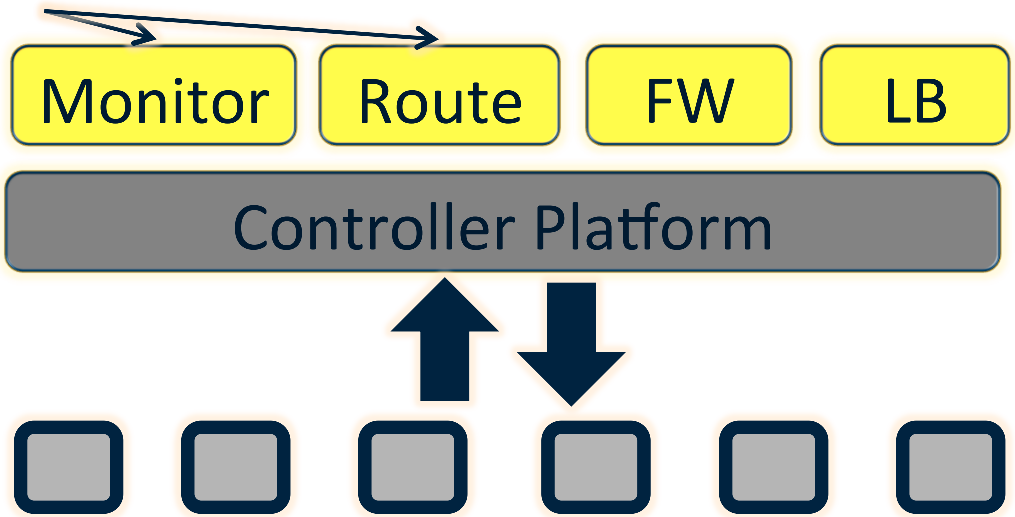
Read/ Monitor state and events

Write policy

OpenFlow Switches

But, Modules Affect the Same Traffic

Each module *partially* specifies the handling of the traffic



Next Lesson:
How to combine modules
into a complete application?

Summary

- SDN control programs: common abstractions
  - Reading and monitoring state and events
  - Computing policy
  - Writing state
- Frenetic: SQL-Like query language to control the traffic seen at the controller
- Other challenges: *Composing* policy, responding to events, compilation