(SDN) Abstractions for Network Management

Nick Feamster

Georgia Tech

feamster@cc.gatech.edu

Students

Arpit Gupta

Hyojoon Kim

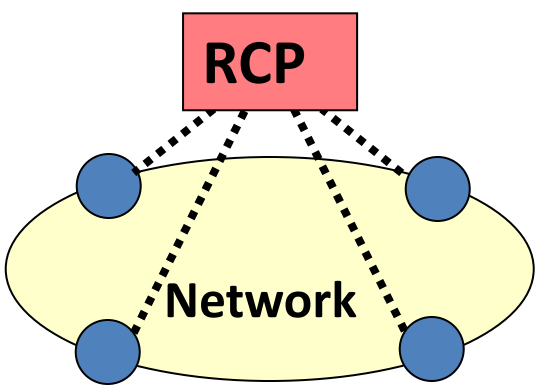
Muhammad Shahbaz

Networks are difficult to manage.

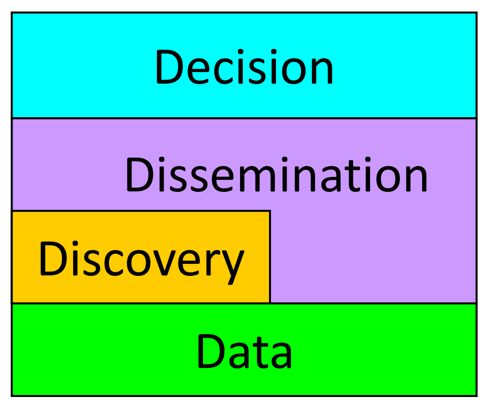
What Does Software Defined Networking Have to Do With It?

- **Distributed configuration is a bad idea**
- **Instead:** Control the network from a logically centralized system

2004



2005

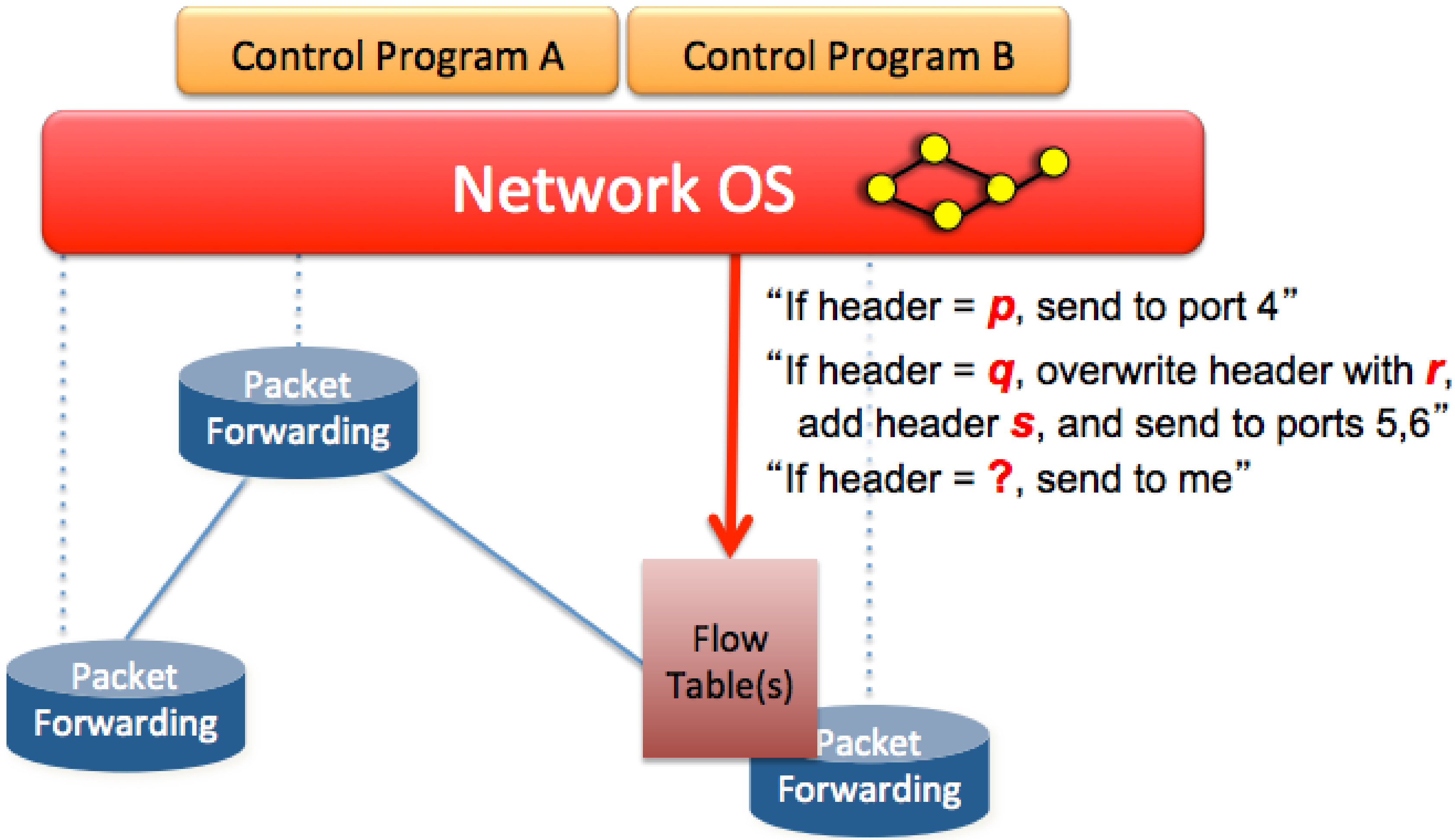


2008

OpenFlow

Feamster *et al*. The Case for Separating Routing from Routers. *Proc. SIGCOMM FDNA*, 2004
Caesar *et al.* Design and implementation of a Routing Control Platform. *Proc NSDI*, 2005

SDN Forwarding Abstraction



Can SDN help?

(Yes…but we must understand why configuration is hard in the first place.)

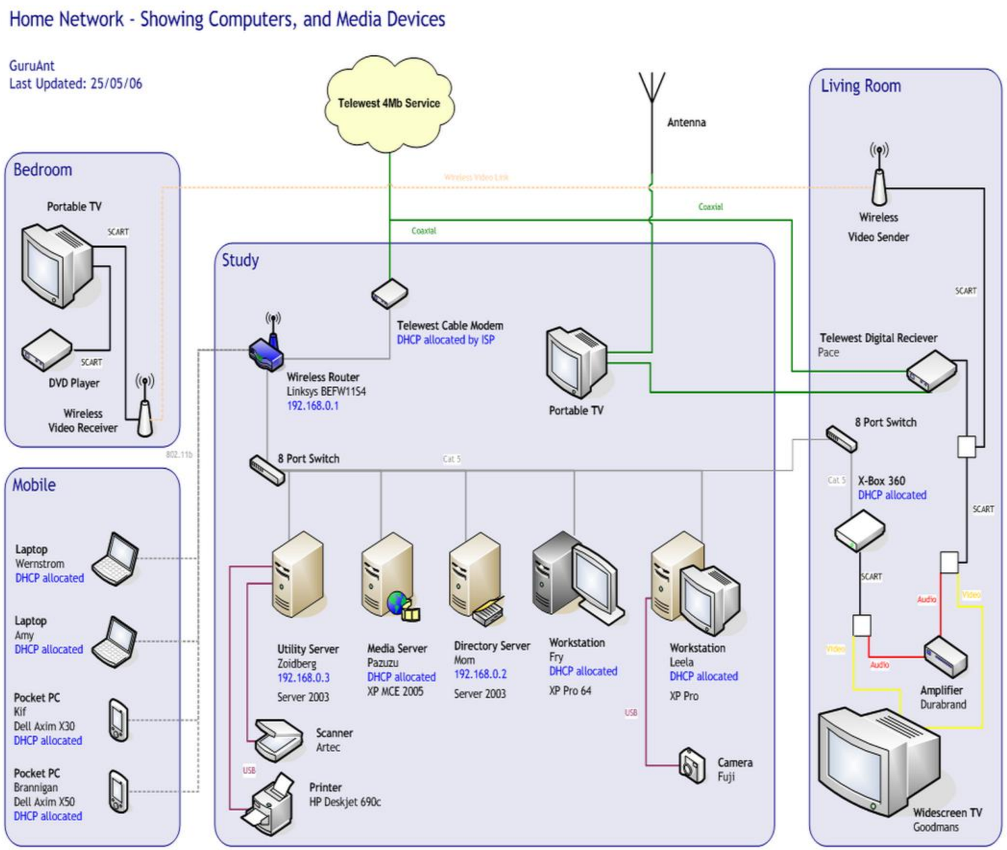
Configuration Changes are Frequent

- Changes to the network configuration occur daily
  - **Errors are frequent**

Georgia Tech	*add*	*del*	*mod*	Total
Routers (16)	31,178	27,064	262,216	326,458
Firewalls (365)	249,595	118,571	171,005	539,171
Switches (716)	216,958	20,185	116,277	353,420
Rtr avg. per device	2,324	1,692	16,389	20,404
FW avg. per device	684	325	469	1,477
Swt avg. per device	303	28	162	494

- Operators must determine
  - **What will happen** in response to a configuration change
  - Whether the configuration is **correct**

Configuration Exposes the (Complex) Physical Topology



http://www.ratemynetworkdiagram.com/?i=526

The Need for Abstractions

- Configuration changes are frequent
  - Policies are **dynamic,** depend on **temporal conditions** defined in terms of external events
  - **Abstraction:** State machine

- Configuration exposes the physical topology
  - Operators do not need to know about the physical topology when configuring policies. Instead, they need a logical view of the network
  - **Abstraction:** Virtual network

- Configuration languages are too low-level
  - Need to configure networks at a higher level
  - **Abstraction:** Functional programming primitives

Abstractions for Network Management

- **State machines for event processing**
  - State-based network policies
  - Composition operators
  - **Example applications:** Home, campus
- **Virtual networks for policy specification**
  - Network operators can express policies in terms of a virtual topology, without needing to know about details of underlying topology
  - **Example applications:** Internet exchange point (IXP)

The Need for Event Processing

As a student, staff, faculty,
that you may consume eac
are in place to ensure that
Exceeding your external ba
class network for up to one

However, there are no band
download/upload as much in
www.utexas.edu, UT Direct).

Guideline Statement

No individual service or system running on the wired/wireless network should use more than **10 gigabytes (10GBs) of bandwidth per day**, regardless of whether it is inbound or outbound over the commodity network link.
Initial Notification: Initially, a system will trigger an overuse notification if the 5 day average for either inbound or outbound usage exceeds 10 GBs. To calculate a 5 day average, we use the **greatest value** of inbound or outbound usage per day. These numbers are totaled then averaged.

Averaging high usage over a 5 day time period allows machines infrequent bursts of activity above the daily limit. As long as the usage totals less than 50 GB in a 5 day period, no notification would be issued.

The situation illustrated in the table below WOULD generate an initial overuse notification. However, no notification would be issued if, for example, the spike in inbound usage on Day 2 were only 40 GB (3.5 + 40 + 1.5 + 2 + 1.5 = 48.5 GB/5 = 9.7 GB)

Usage	Day 1	Day 2	Day 3	Day 4	Day 5	Total/Average
Inbound	**3.5 GB**					
Outbound	1 GB					

Notice that, in the followin
10 GBs. However, an ove
usage over this five day p

Bandwidth Allocations

Role	Allocation/Week
Students	500 Megabytes
Graduate Fellowships[1]	25 Gigabytes
Faculty	100 Gigabytes
Full-time Staff and Official Visitors	25 Gigabytes
Part-time Staff	5 Gigabyte

Procedure

It is possible to determine the bandwidth used by a single computer on ResNet over a set period of time. Based on this information, the following rules will be enforced:

1. On a daily basis, compute the per computer bandwidth usage for ResNet network traffic destined to sites off-campus.
2. For computers where the utilization exceeds a daily threshold of 5 GB (gigabytes), e-mail the owner of the computer, noting:
   - the bandwidth usage issues (e.g. shared resource, student's use as compared to the average, etc.)
   - pointers to information on curbing outbound traffic usage
   - consequences for continued over-use (include information about how many notices have been sent)
   - information on how to request additional bandwidth if needed for academic or research projects
3. After 5 such notices, the student's computer will have its outbound bandwidth restricted to 64 kb/s (kilobits per second) for the remainder of the semester.

The Need for Event Processing

- Rate limit all Bittorrent traffic between the hours of 9 a.m. and 5 p.m.
- Do not use more than 100 GB of my monthly allocation for Netflix traffic
- If a host becomes infected, re-direct it to a captive portal with software patches
- …

State Machine Abstraction

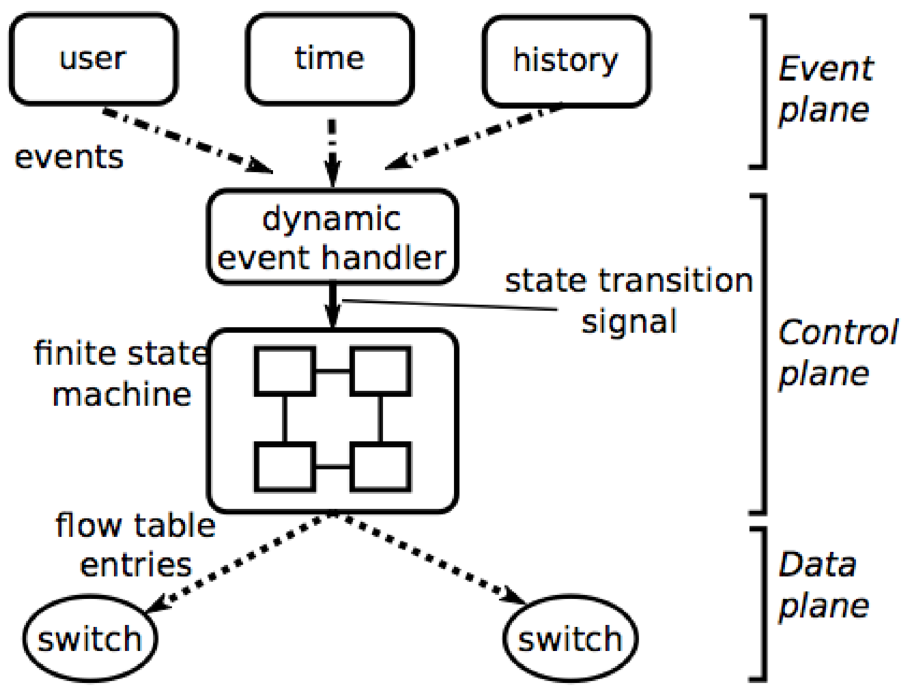
- **State:** A set of domain values.
- **Events:** Trigger state transitions in the controller's finite state machine.
  - Intrusions
  - Traffic fluctuations
  - Arrival/departure of hosts
- State machine transitions update the current policy/program that is "running" in the network.

Domains that Can Define States

domains	***Examples***
Time	peak traffic hours, academic semester start date
History	amount of data usage, traffic rate, traffic delay, loss rate
User	identity of the user, assignment to distinct policy group
Flow	ingress port, ether src, ether dst, ether type, vlan id, vlan priority, IP src, IP dst, IP dst, IP ToS bits, src port, dst port

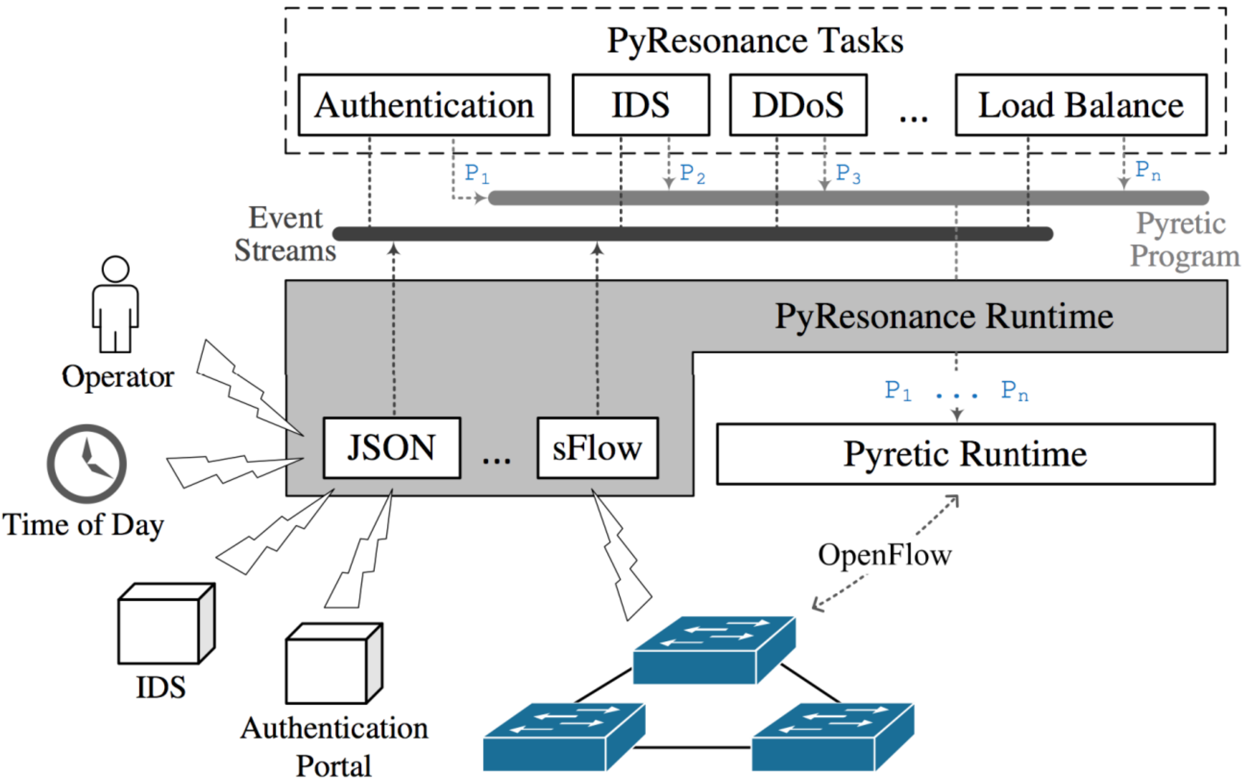
Resonance: Event-Based Network Control

Idea: Express network policies using state machines.



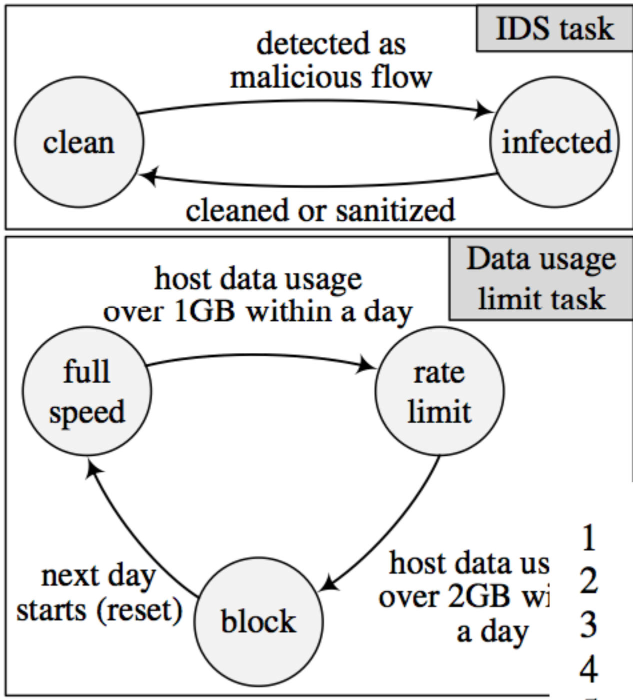
http://resonance.noise.gatech.edu/

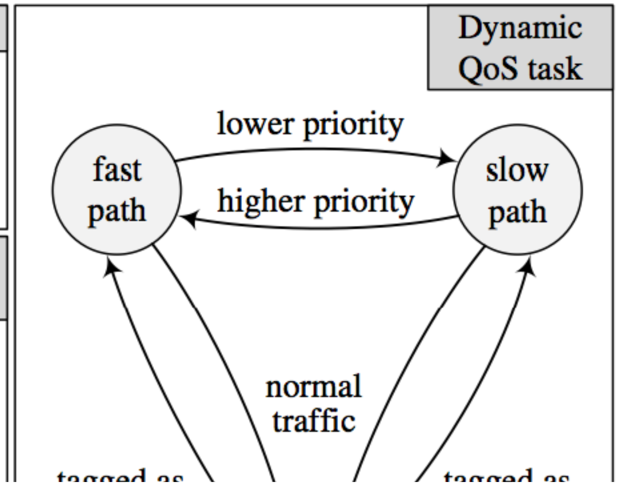
Resonance: Dynamic Event Handler



- Controller reacts to events
- Determines event source
- Updates state based on event type
- Can process both internal and external events

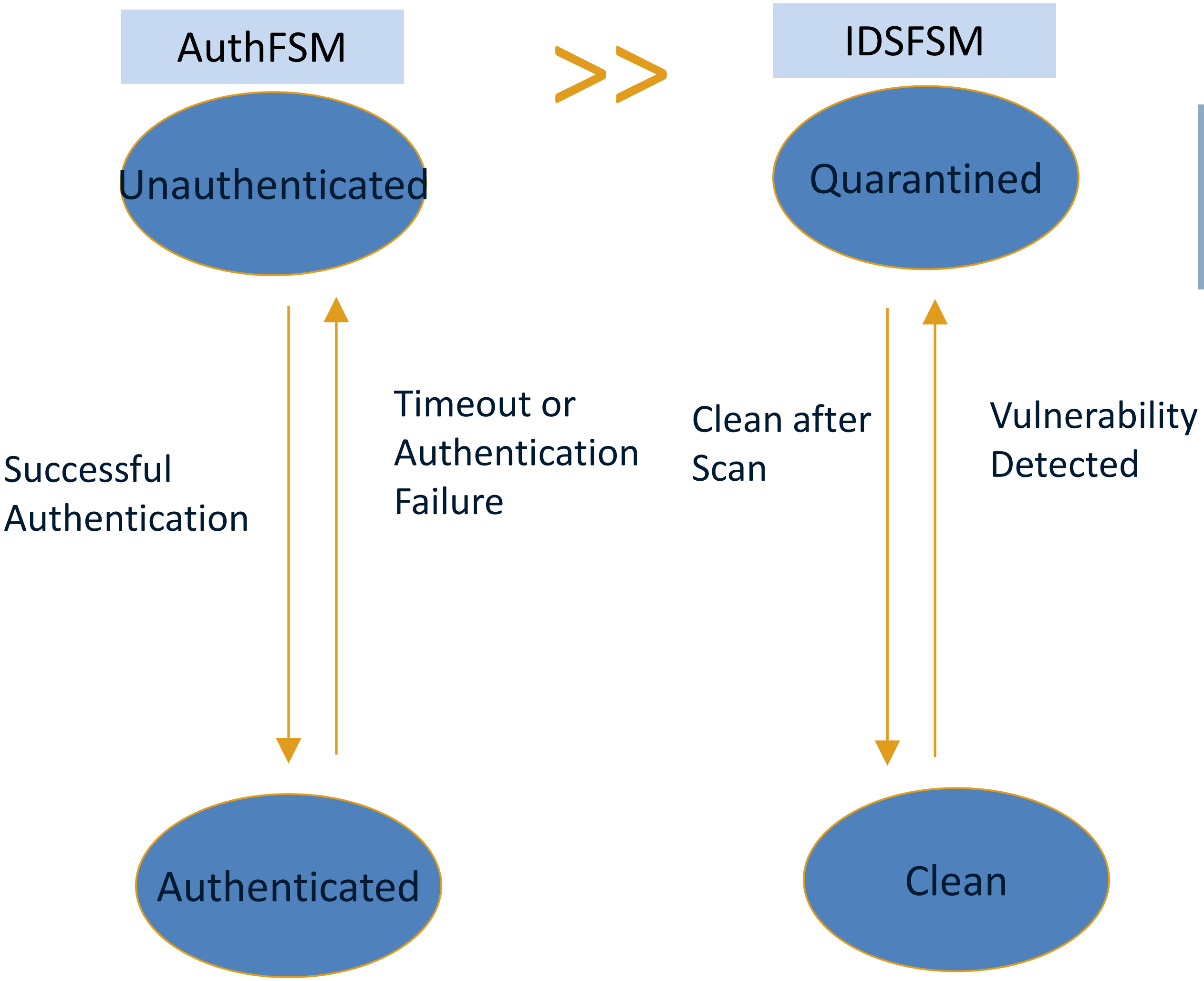
Example FSMs and Programs





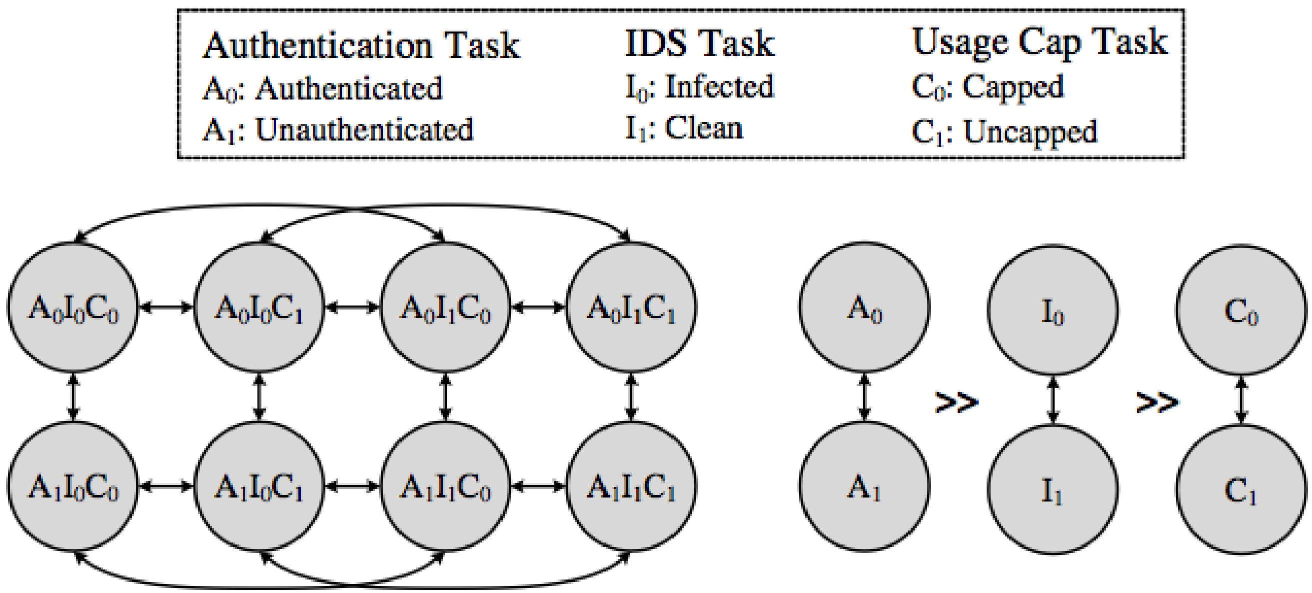

```
class IDSTask(BaseTask):
  ...
  def infected(self):
    return drop

  def clean(self):
    return passthrough

  def action(self):
    clean_flows=self.fsm.get_action('clean')
    p1 = if_(clean_flows,
              self.clean(),
              self.infected())
    return p1
```

Composition Mitigates State Explosion



Simpler: Use Pyretic to sequentially compose FSMs!

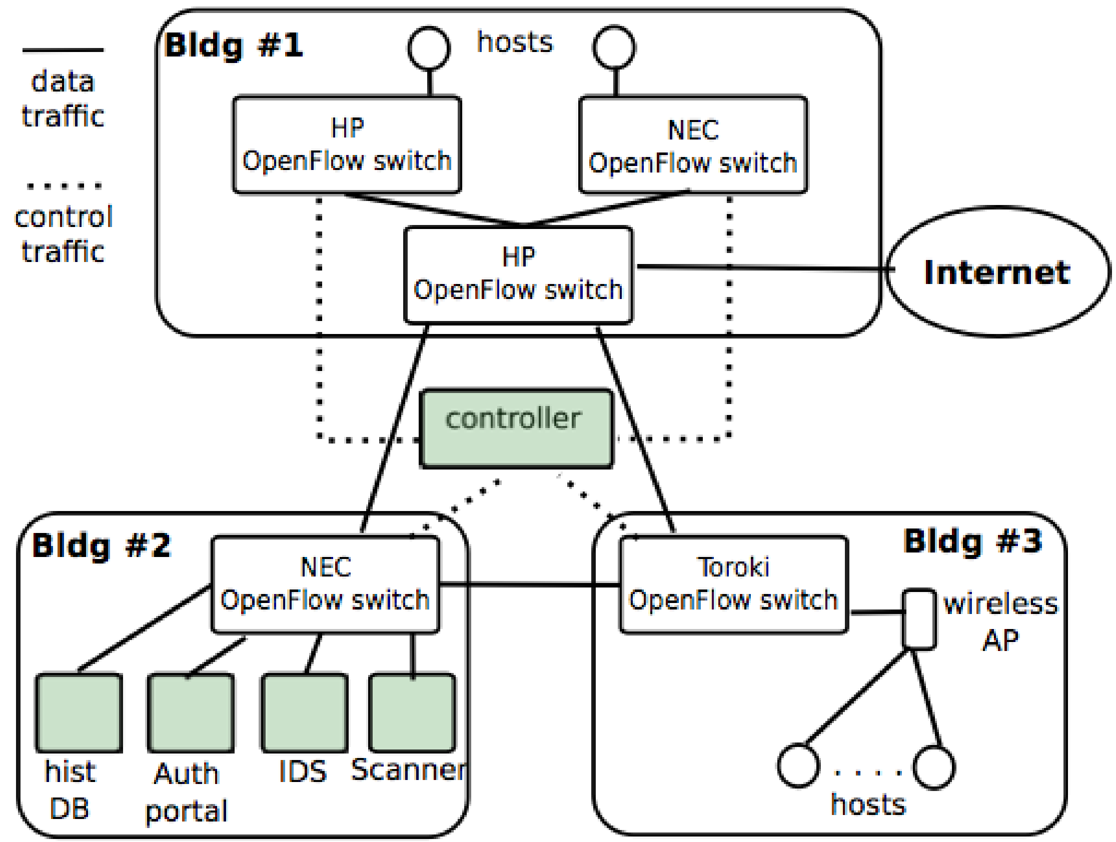
Mitigating State Explosion



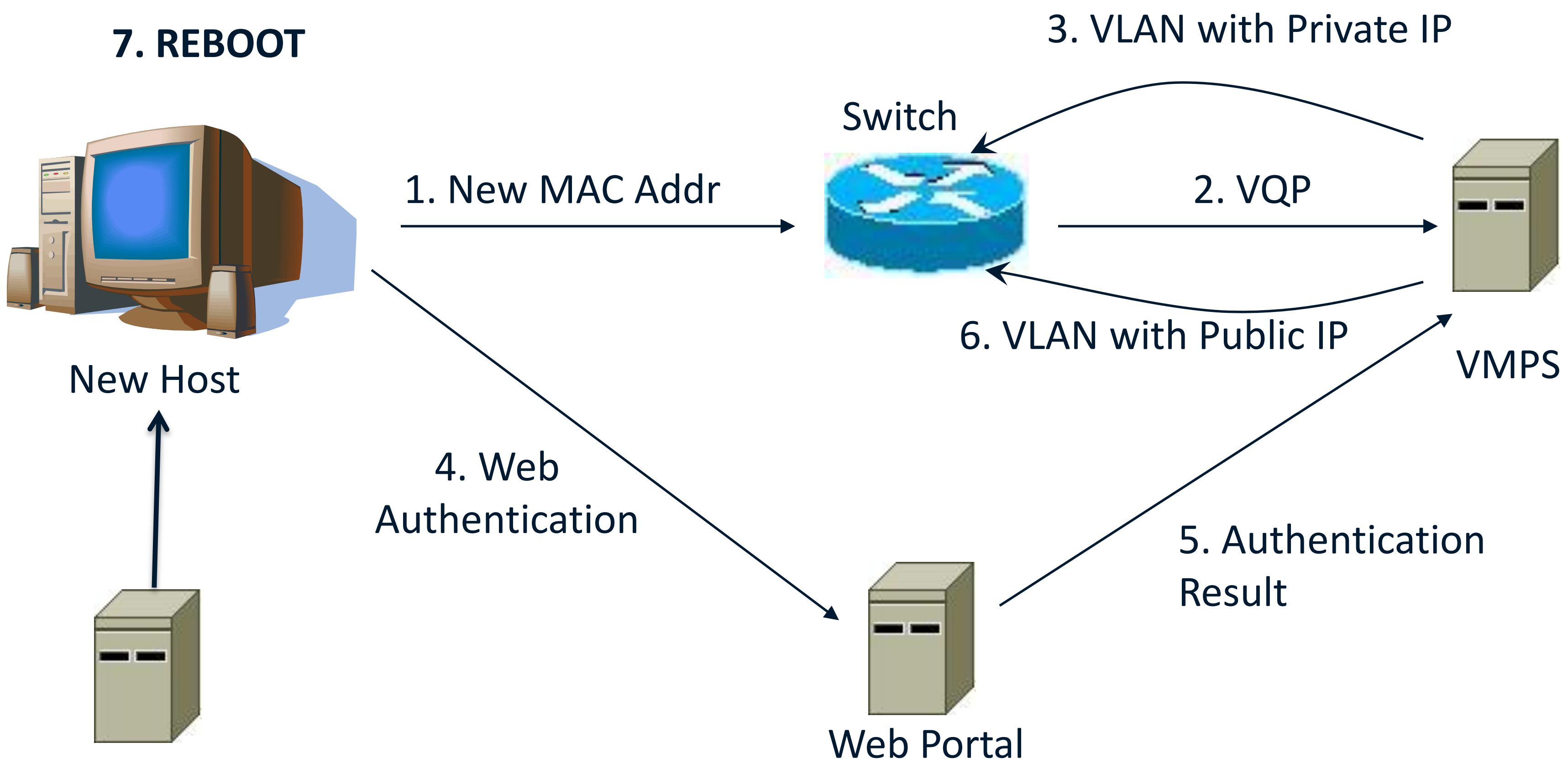
(a) *Without composition* **(b)** *With composition*

Campus Network Deployment

- Software-defined network in use across three buildings across the university
- Redesign of network access control
- Also deployed at other universities



Application: Campus Access Control



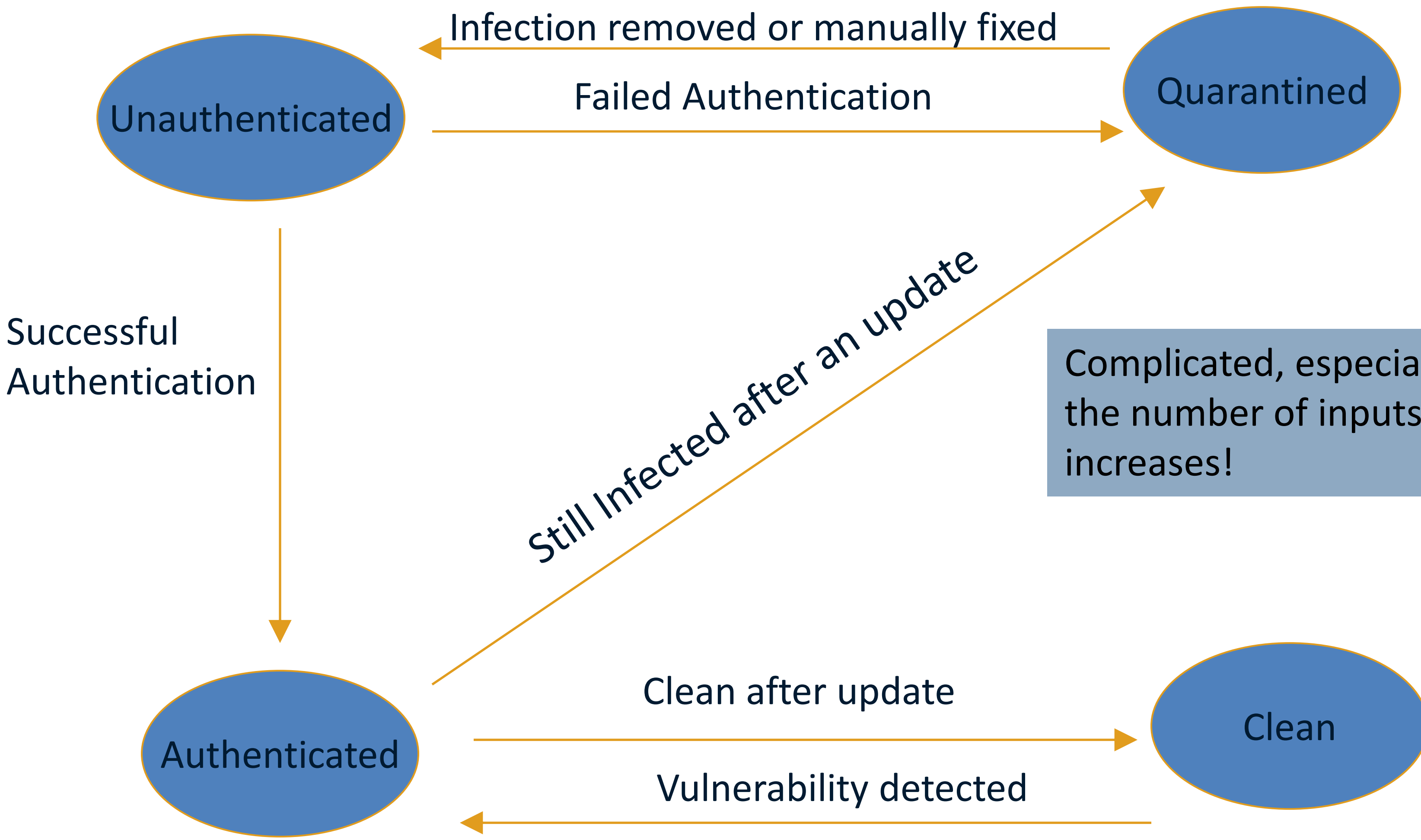
Problems with Conventional Approach

- Access control is **too coarse-grained**
  - Static, inflexible and prone to misconfigurations
  - Need to rely on VLANs to isolate infected machines

- **Cannot dynamically remap** hosts to different portions of the network
  - Needs a DHCP request which for a windows user would mean a reboot

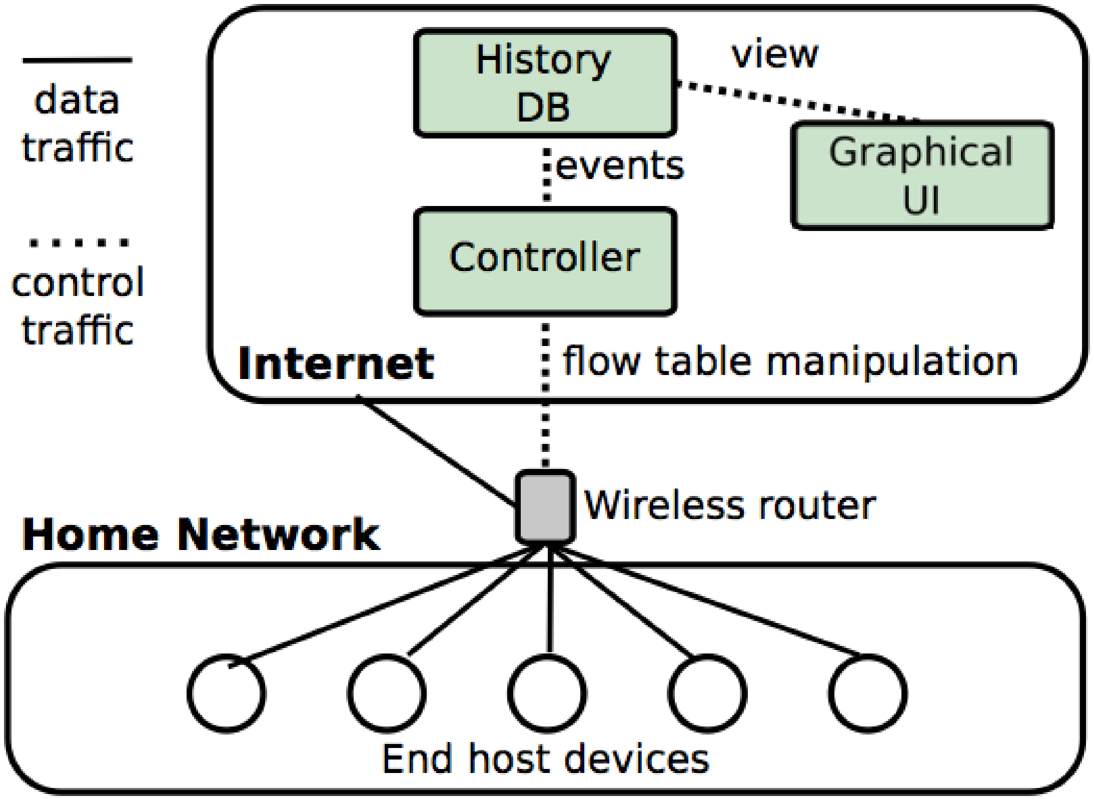
- **Monitoring is not continuous**

Policy: State Machine, OpenFlow Rules



Complicated, especially as the number of inputs increases!

Home Network Deployment



- User monitors behavior and sets policies with UI
- Resonance controller manages policies and router behavior
- Clean UI built on top of abstractions

Abstractions for Network Management

- **State machines for event processing**
  - State-based network policies
  - Composition operators
  - **Example applications:** Home, campus
- **Virtual networks for policy specification**
  - Network operators can express policies in terms of a virtual topology, without needing to know about details of underlying topology
  - **Example applications:** Internet exchange point (IXP)

Improving Interdomain Routing

- Routing only on destination IP prefix
  - No customization of routes by application, sender
- Influence only over neighbors
  - No ability to affect end-to-end paths
- Indirect expression of policy
  - Indirect mechanisms to influence path selection (e.g., local preference, AS path prepending)

What BGP Cannot Support

- **Application-specific peering:** Peering for specific applications like video
- **Redirection to middleboxes:** Redirection of specific traffic subsets to middleboxes
- **Traffic offloading:** Avoiding sending traffic through intermediate peers at exchanges
- **Preventing free-riding:** Dropping inbound traffic that is not associated with any peering relationship
- **Wide-area load balancing:** Rewriting destination IP address for load balancing (vs. DNS)

Evolve BGP at Internet Exchanges

- New technology at a single IXP can yield benefits for tens to hundreds of ISPs.

- IXPs are currently experiencing a rebirth (*e.g.*, Open IX) and wanting to differentiate.

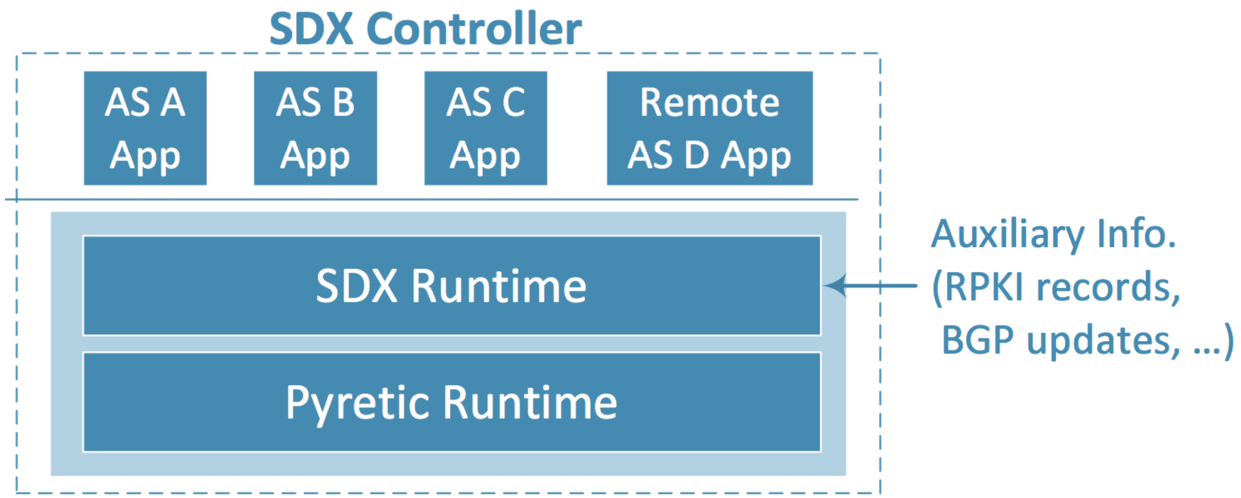
- New applications create need
for richer peering.

SDN: Challenges and Opportunities

- **Opportunities:** Freedom from constraints
  - Matching of different packet header fields
  - Control messages from remote networks
  - Direct control over data plane

- **Challenges:** No existing SDN control framework for interdomain routing
  - **Scaling:** Hundreds to thousands of ISPs at an IXP

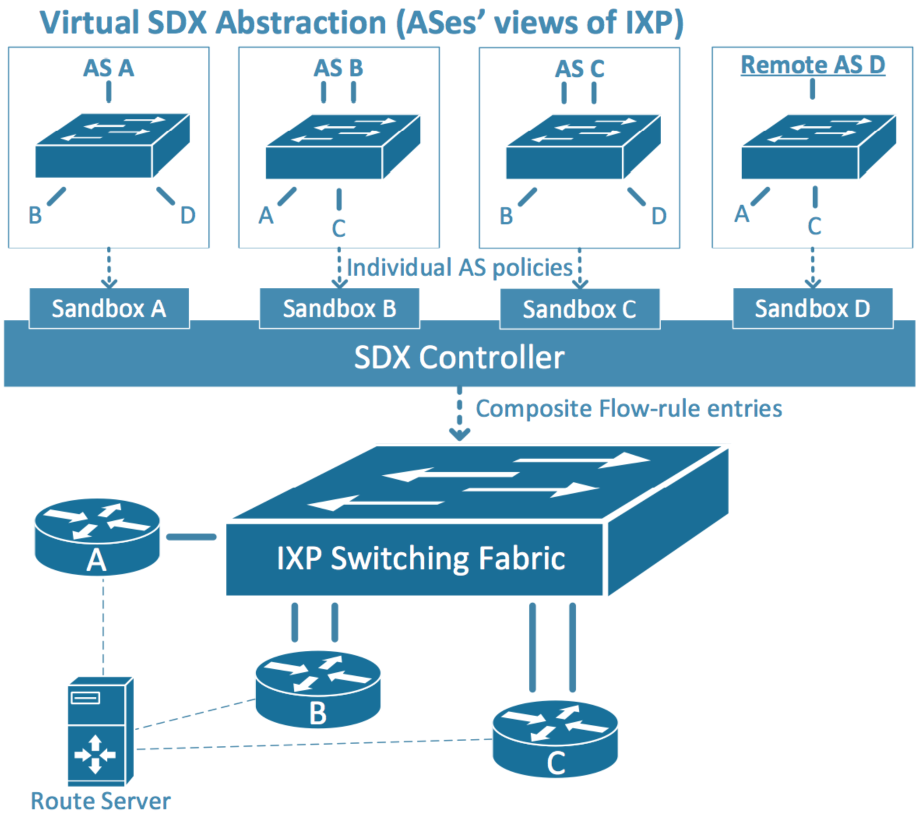
SDX Design: Multiple "Applications"



- **Problem:** Each participant needs to see its own version of the topology.
- **Soluton: Each AS sees only its own virtual IXP topology**
- Applications run on top of SDX runtime
  - Makes decisions, resolves conflicts based on both participants' applciations and policies and auxiliary information (e.g., route server information)

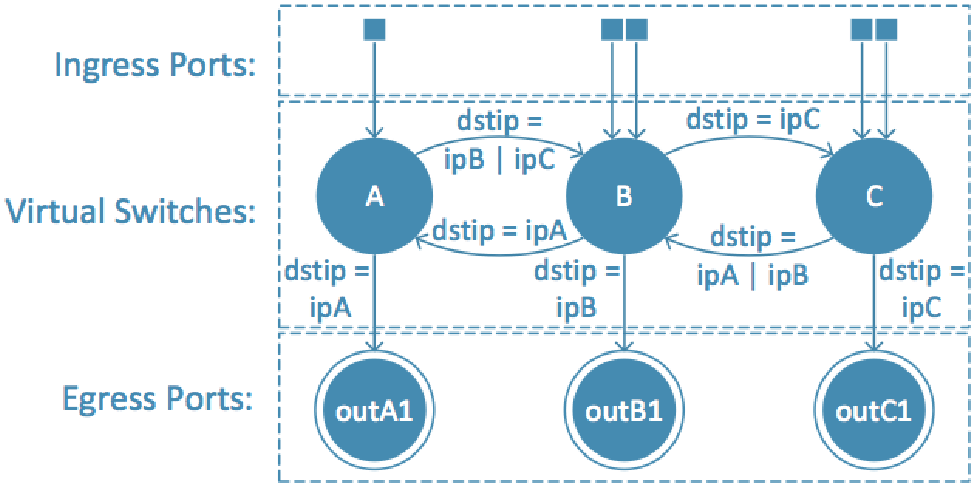
Virtual Network Abstraction

- ISPs that do not have business relationships with one another cannot see each other.
  - (*e.g.*, AS A and C have no direct connection)
- Enforced using symbolic execution at SDX



Implementing the Virtual Network Abstraction

- **Symbolic execution:** Tag packets on input, use state machine to determine output port.



- **Sequential composition of ISP policies:** SDX runtime composes policies in order based on result from symbolic execution.

Abstractions for Network Management

- **State machines for event processing**
  - State-based network policies
  - Composition operators
  - **Example applications:** Home, campus
- **Virtual networks for policy specification**
  - Network operators can express policies in terms of a virtual topology, without needing to know about details of underlying topology
  - **Example applications:** Internet exchange point (IXP)

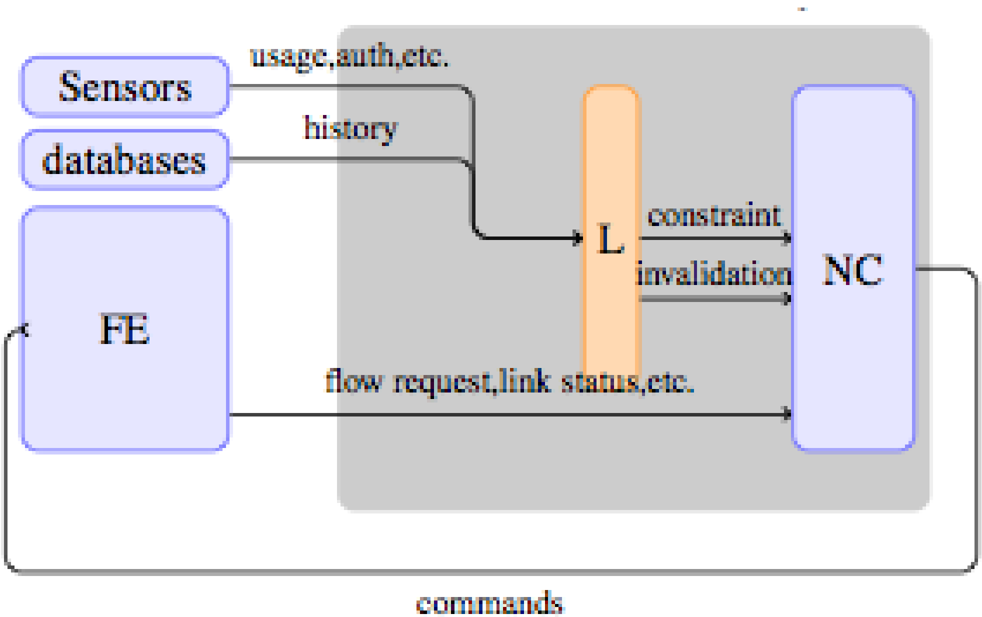
The Need for Abstractions

- Configuration changes are frequent
  - Policies are **dynamic,** depend on **temporal conditions** defined in terms of external events
  - **Abstraction:** State machine

- Configuration exposes the physical topology
  - Operators do not need to know about the physical topology when configuring policies. Instead, they need a logical view of the network
  - **Abstraction:** Virtual network

- Configuration languages are too low-level
  - Need to configure networks at a higher level
  - **Abstraction:** Functional programming primitives

Procera: Functional Programming Abstractions



Define a signal function for a device going over (or under) the usage cap:

```
overUnderEvent =
  proc env -> do
    capMap <- capTracker -< env
    usagedb <- usageTracker -< env
    usageChanges <- usageChangesTracker -< env
    let now = calendarTime env
    let over src =
      monthlyUsage usagedb now > capMap ! src
    condSplit over -< usageChanges
```

Define the set of devices over the cap:

```
overSetStream =
  proc env -> do
    (over, under) <- overUnderEvent -< env
    toSetStream -< (over, under)
```

- **Input signals** from environment
- **Windowing and aggregation functions** that process and combine
- Periodically updates a **flow constraint function** that controls the forwarding elements

Procera Language Properties

- **Declarative Reactivity:** Describing when events happen, what changes they trigger, and how permissions change over time.
- **Expressive and Compositional Operators:** Building reactive permissions out of smaller re-active components.
- **Well-defined Semantics:** Simple semantics, simplifying policy specification.
- **Error Checking & Conflict Resolution:** Leveraging well-defined, mathematical semantics.

Summary

- Configuration changes are frequent
  - Policies are **dynamic,** depend on **temporal conditions** defined in terms of external events
  - **Abstraction:** State machine

- Configuration exposes the physical topology
  - Operators do not need to know about the physical topology when configuring policies. Instead, they need a logical view of the network
  - **Abstraction:** Virtual network

- Configuration languages are too low-level
  - Need to configure networks at a higher level
  - **Abstraction:** Functional programming primitives (Procera)

Other Collaborators

- Resonance
  - Josh Reich (Princeton)
- SDX
  - Jennifer Rexford (Princeton)
  - Scott Shenker (Berkeley)
  - Laurent Vanbever (Princeton)
- Procera
  - Andi Voellmy (Yale)