Applying SDN to Network Management Problems

Nick Feamster
University of Maryland

Addressing the Challenges of Network Management

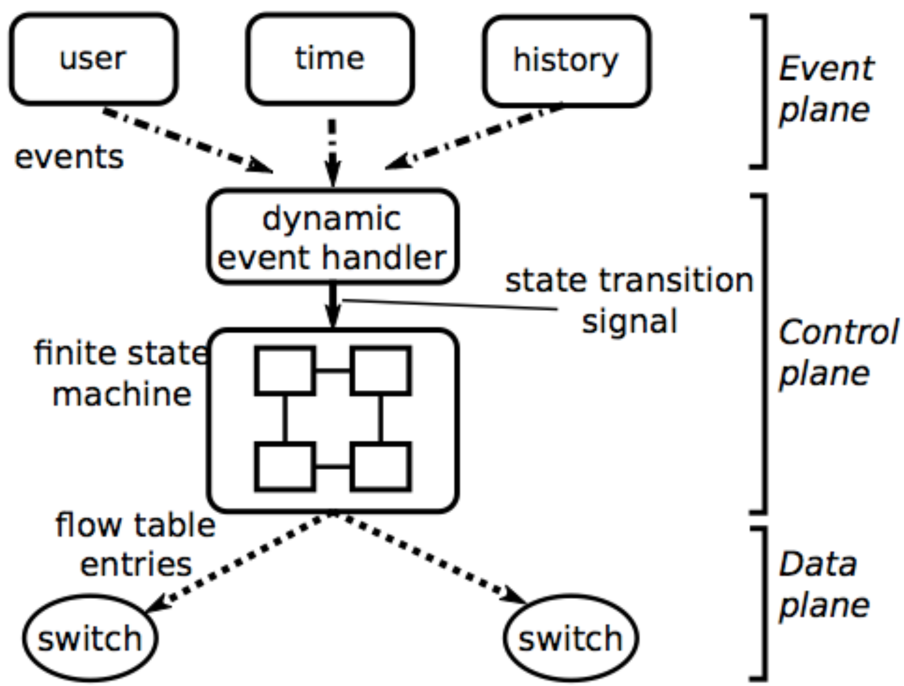
Challenge	Approach	System
Frequent Changes	Event-Based Network Control	Lithium
Minimal Visibility and Control	Programmable Measurement Platform	BISmark
Low-Level Configuration	High-Level Policy Language	Procera

Insight: Network Configuration is Really Just 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
- …

Lithium: Event-Based Network Control

Main Idea: Express network policies as event-based programs.

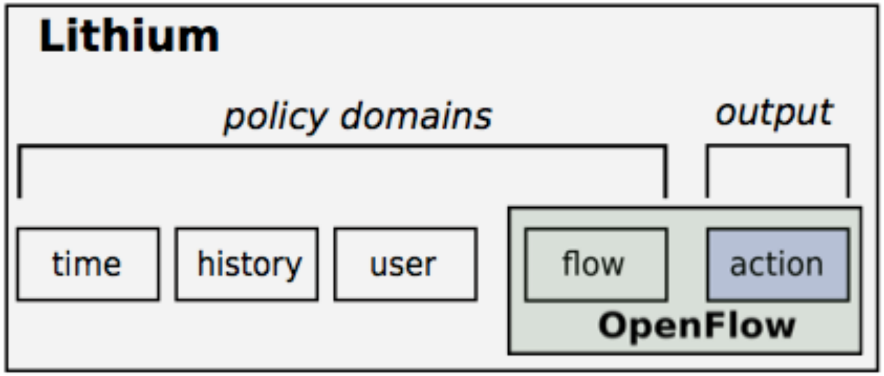


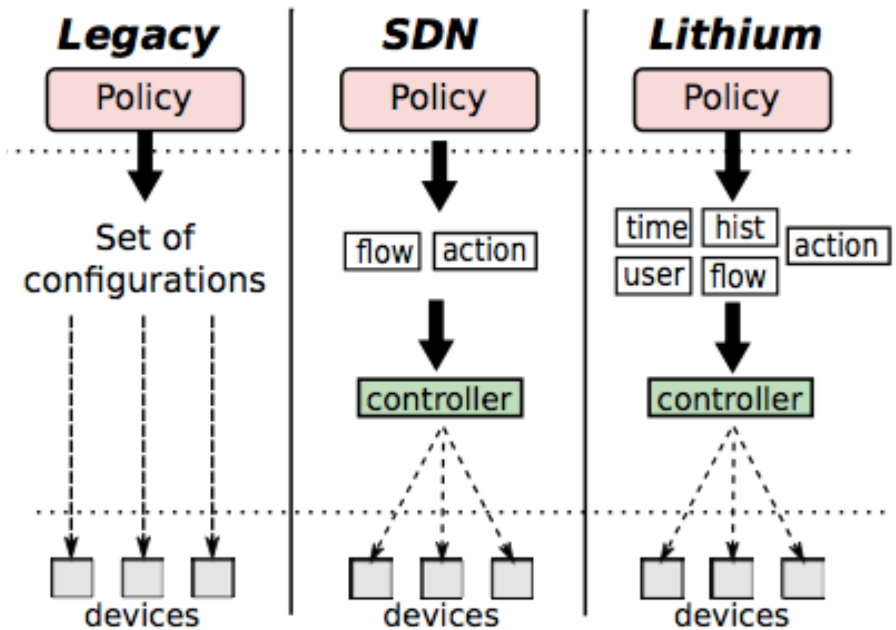
Resonance: Inference-Based Access Control for Enterprise Networks. Nayak, Reimers, Feamster, Clark. *ACM SIGCOMM Workshop on Enterprise Networks*. August 2009.

Event-Driven Control Domains

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

Extending the Control Model





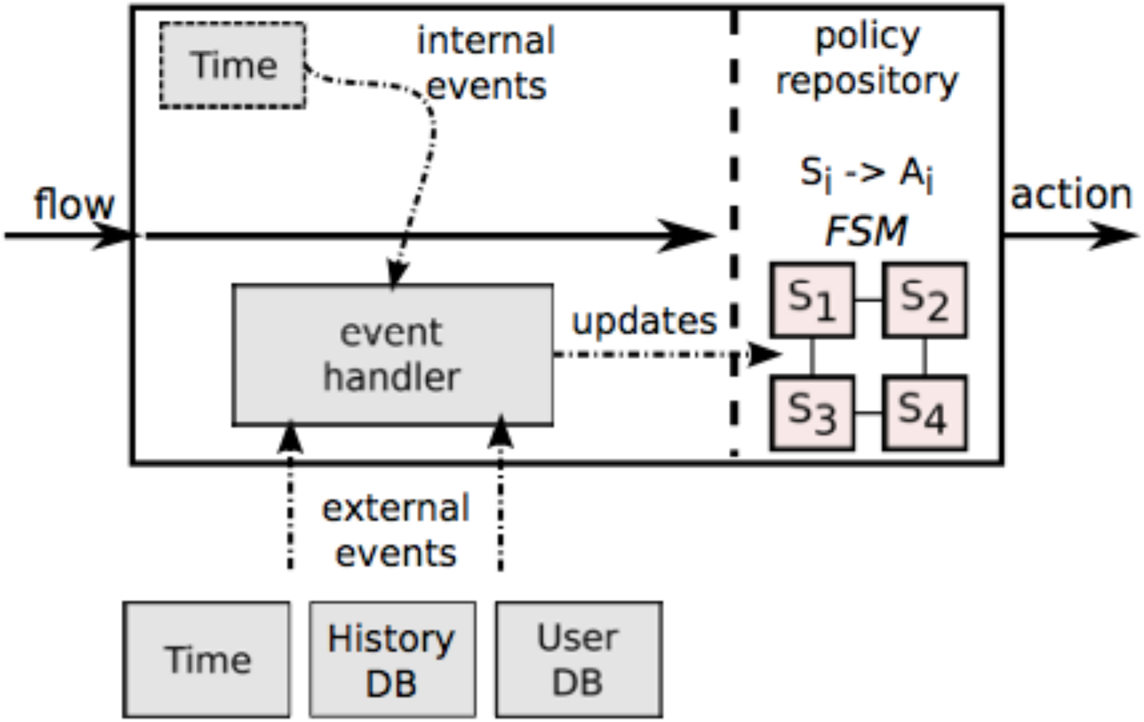
- Currently, OpenFlow only operates on flow properties
- Lithium extends the control model so that actions can be taken on **time**, **history**, and **user**

Lithium: Finite State Machine

- **State:** A set of domain values represents a state. Representation of network state.

- **Events:** Event-driven control domains invoke events, which trigger state transitions in the controller's finite state machine.
  - Intrusions
  - Traffic fluctuations
  - Arrival/departure of hosts

Lithium: Dynamic Event Handler

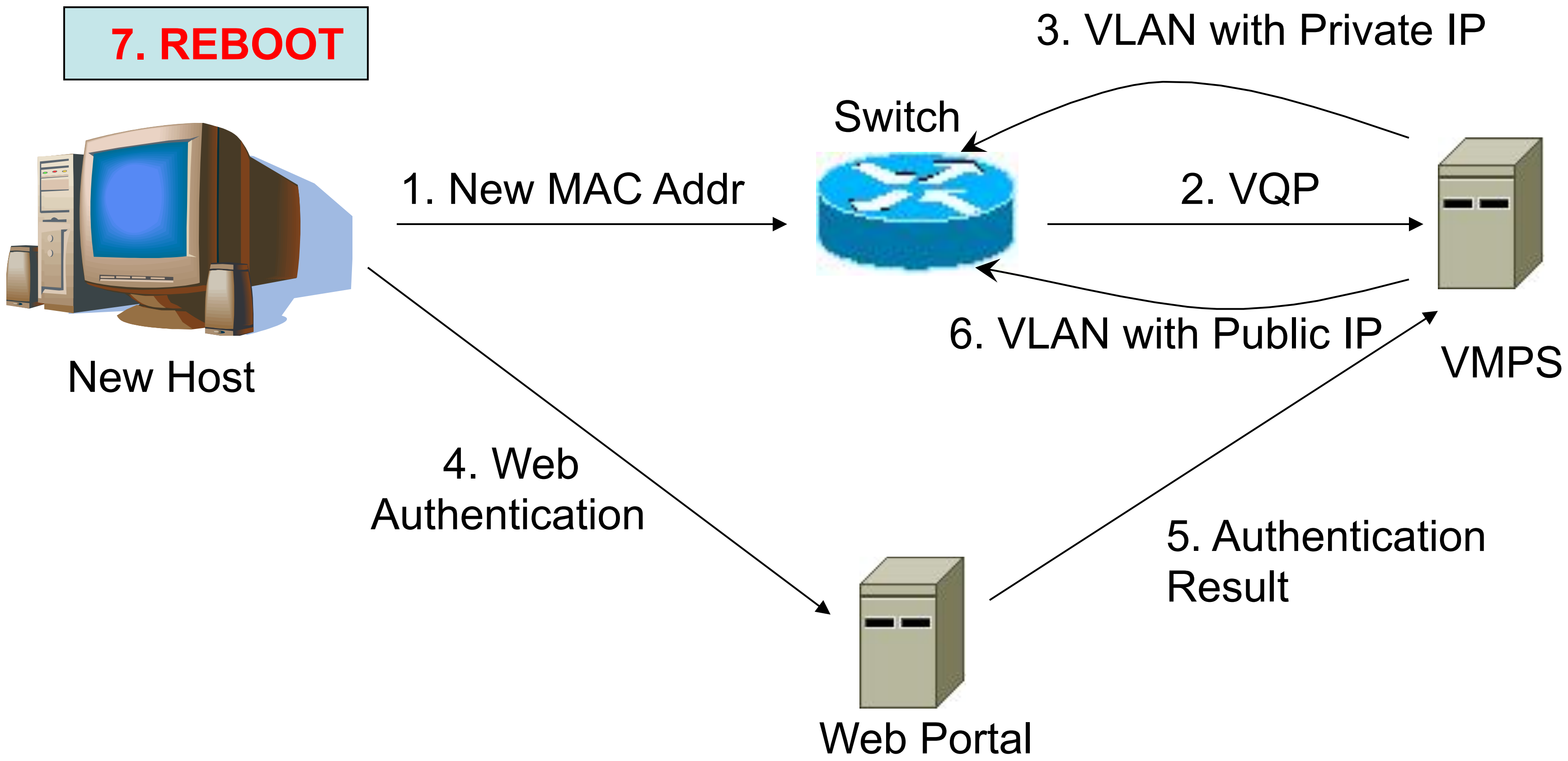


- Reacts to domain events
- Determines event source
- Updates state based on event type
- Can process both internal and external events

Two Real-World Deployments

- **Access control in enterprise networks**
  - Re-implementation of access control on the Georgia Tech campus network
  - **Today:** Complicated, low-level
  - **With SDN:** Simpler, more flexible
- **Usage control in home networks**
  - Implementation of user controls (e.g., usage cap management, parental controls) in home networks
  - **Today:** Not possible
  - **With SDN:** Intuitive, simple

Example from Campus Network: Enterprise Access Control

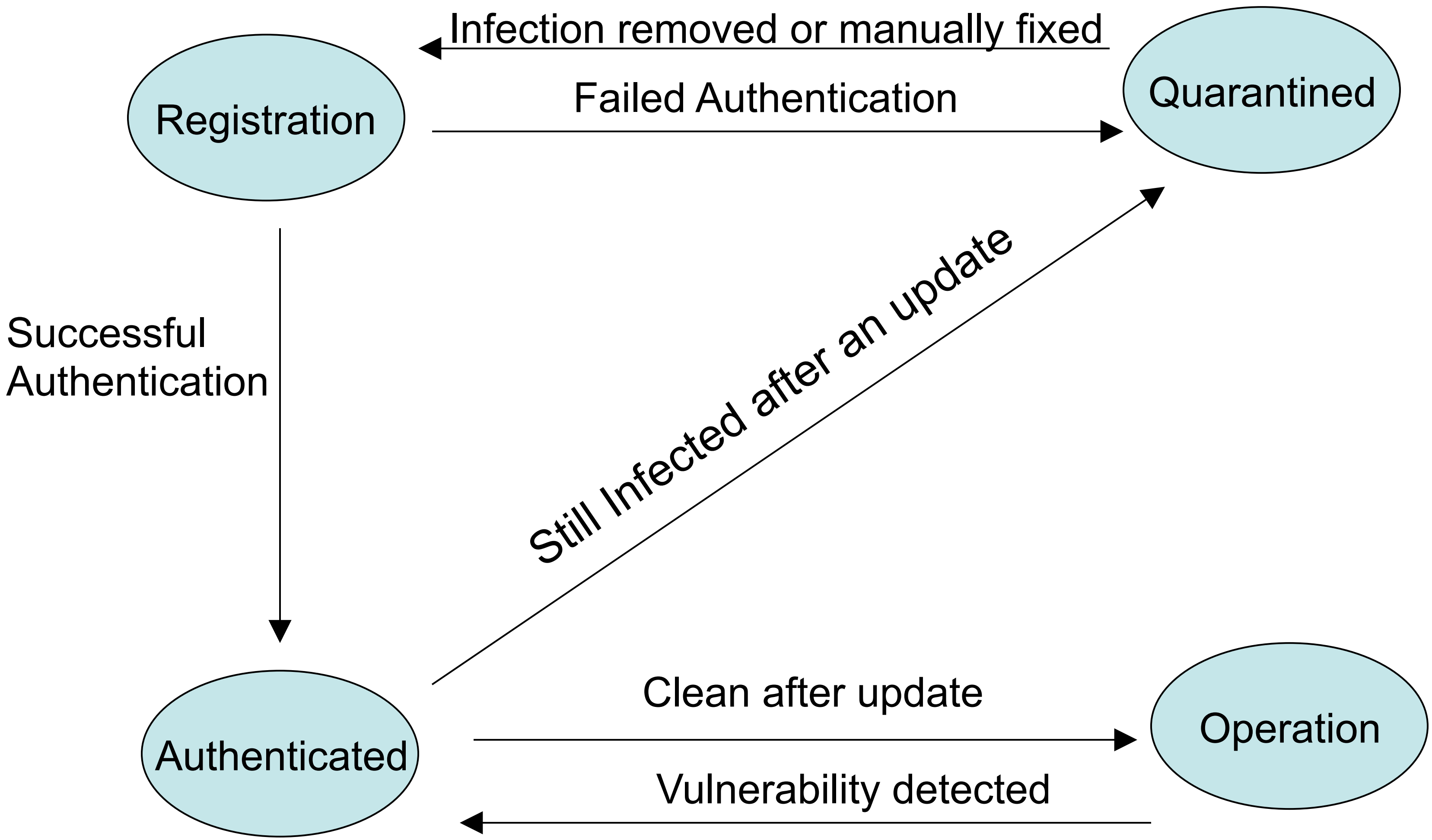


Problems with Current Approach

- 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**

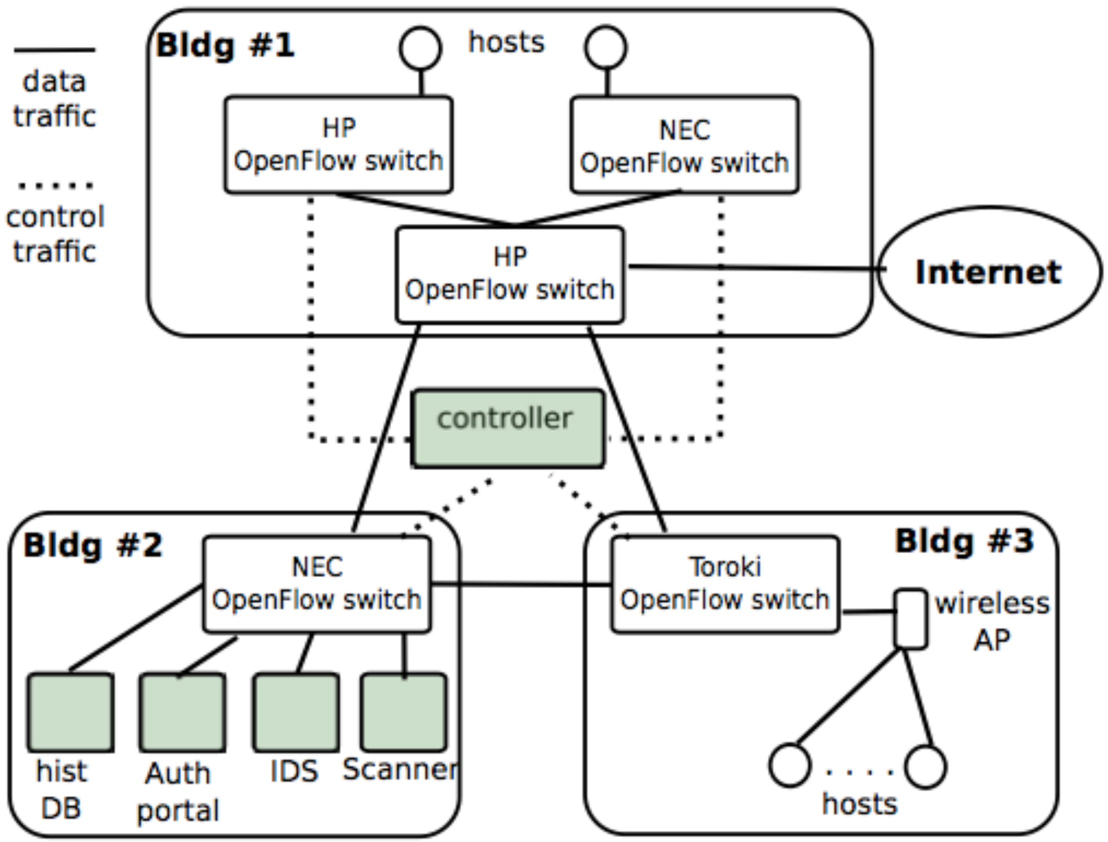
Express policies that incorporate network dynamics.

Lithium State Machine for Campus Network



Deployment: Campus Network

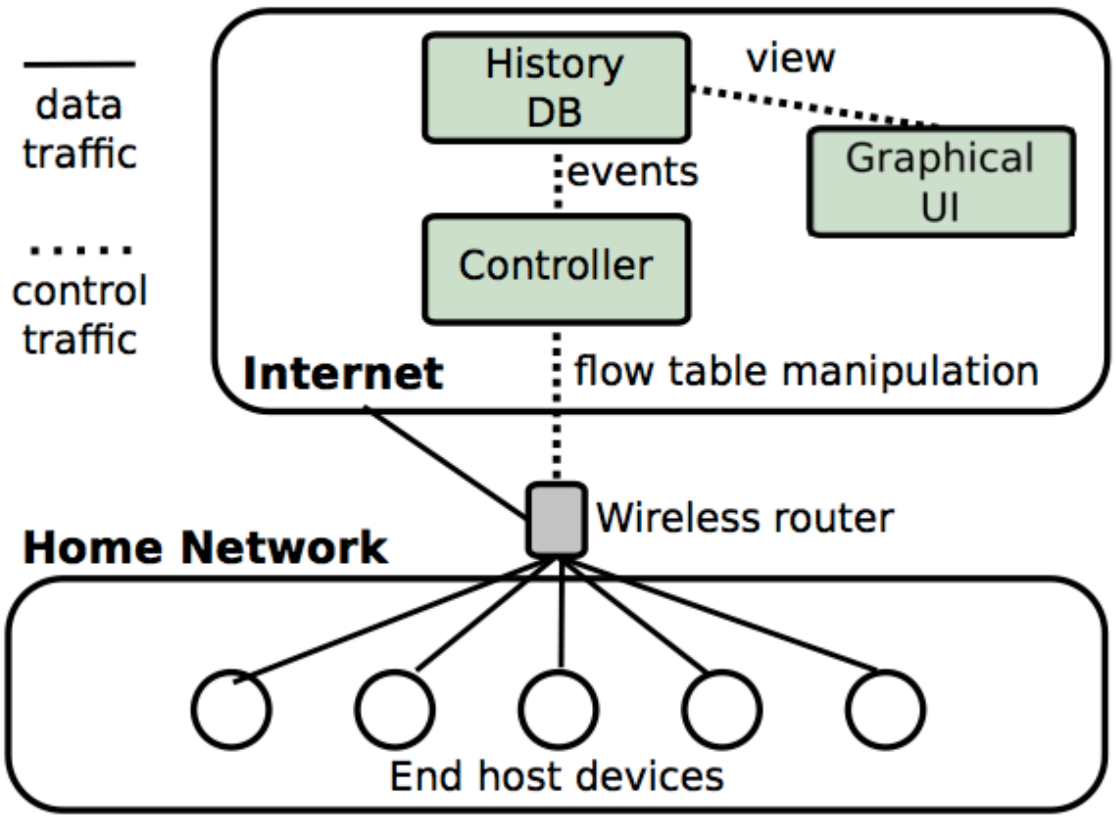
- Software-defined network in use across three buildings across the university
- Redesign of network access control
- Also deployed at other universities



Example From Home Networks: Usage Control

- Network management in homes is challenging
- One aspect of management: **usage control**
  - Usage cap management
  - Parental control
  - Bandwidth management
- **Idea:** Outsource network management/control
  - Home router runs OpenFlow switch
  - Usage reported to off-site controller
  - Controller adjusts behavior of traffic flows

Deployment: Home Networks

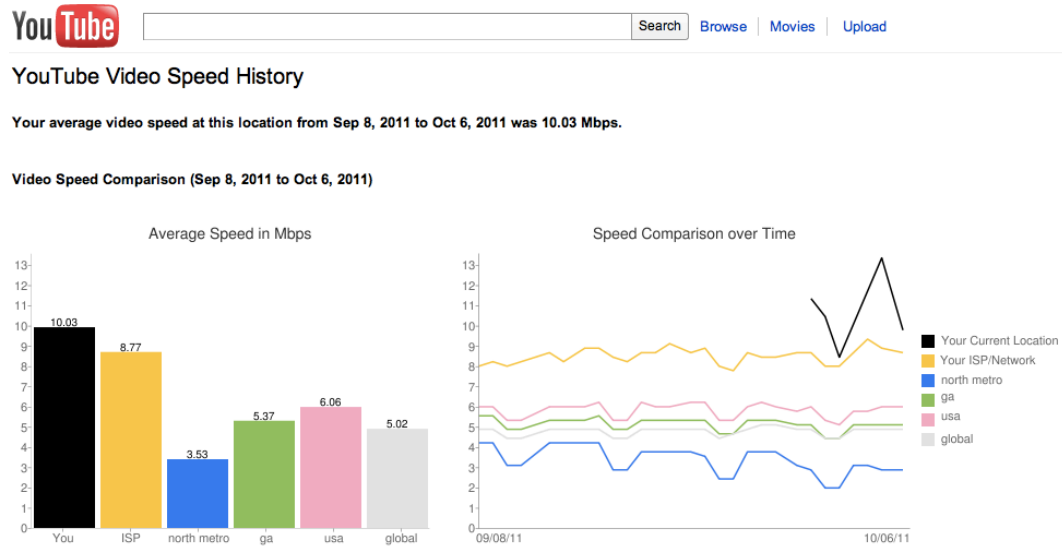


- User monitors behavior and sets policies with UI
- Lithium controller manages policies and router behavior

Network Management Challenges

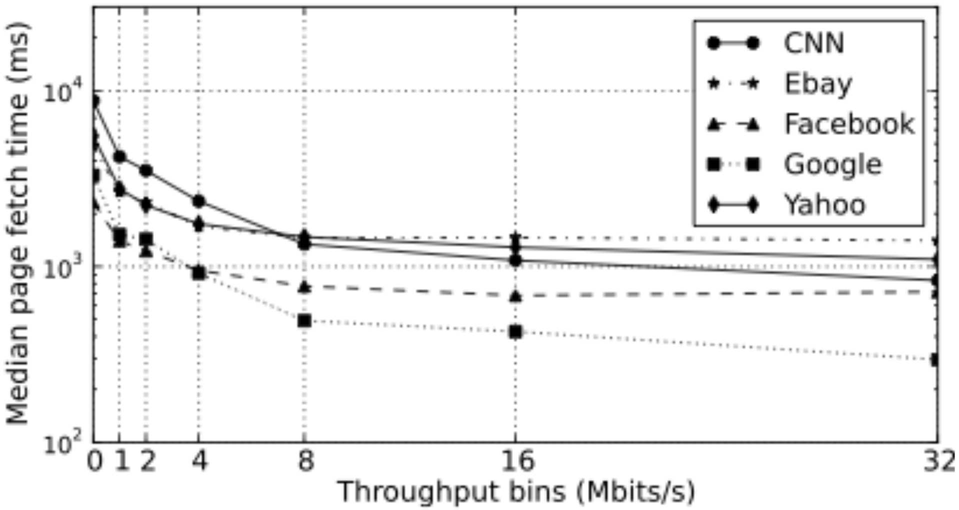
Challenge	Approach	System
Frequent Changes	Event-Based Network Control	Lithium
Minimal Visibility and Control	Programmable Measurement Platform	BISmark
Low-Level Configuration	High-Level Policy Language	Procera

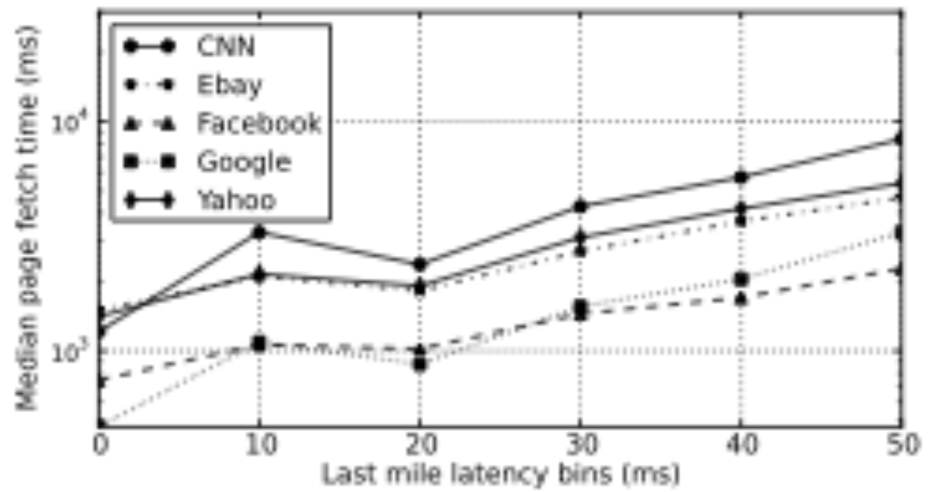
Little Visibility Into Performance



- **Access ISPs**
  - What performance are customers seeing?
  - Can they gain better visibility into downtimes?
  - Can visibility into problems help reduce service calls?
- **Content Providers**
  - How do content routing or traffic engineering decisions affect end user performance
- Also, consumers and regulators

Making the Web Faster (Why Latency Matters)





Latency can dominate page load times, especially for service plans with higher throughput.

Where Programmability in the Home can Help

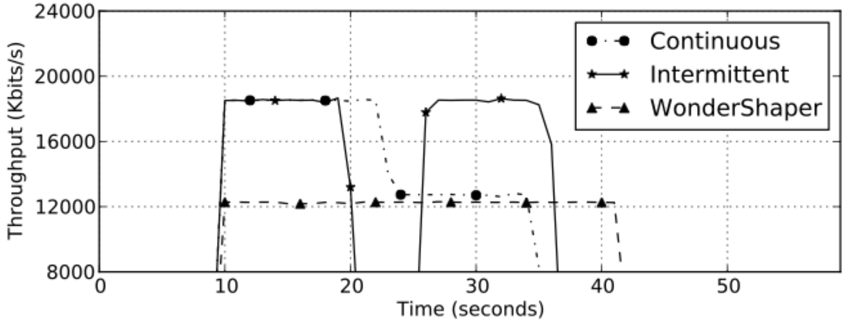
- Reconfiguration of the home network in response to different conditions (e.g., traffic shaping)

- Proactively (and programmatically?) prefetching and caching content, *before* the last mile.

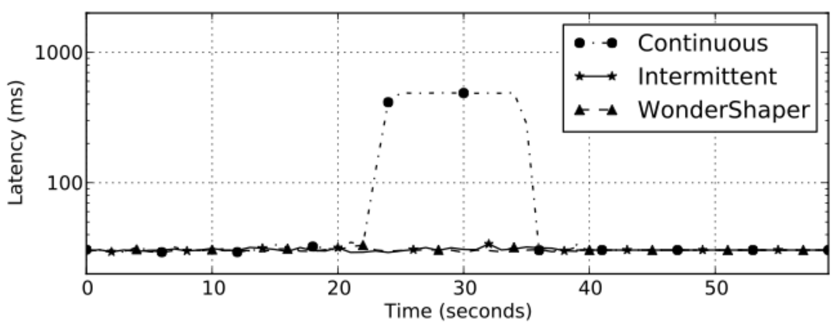
- **Triggered on-demand measurements (e.g., from content providers, ISPs, etc.)**

Programmable Traffic Shaping

- Intermittent or shaped traffic can achieve same levels of throughput, without incurring high latency

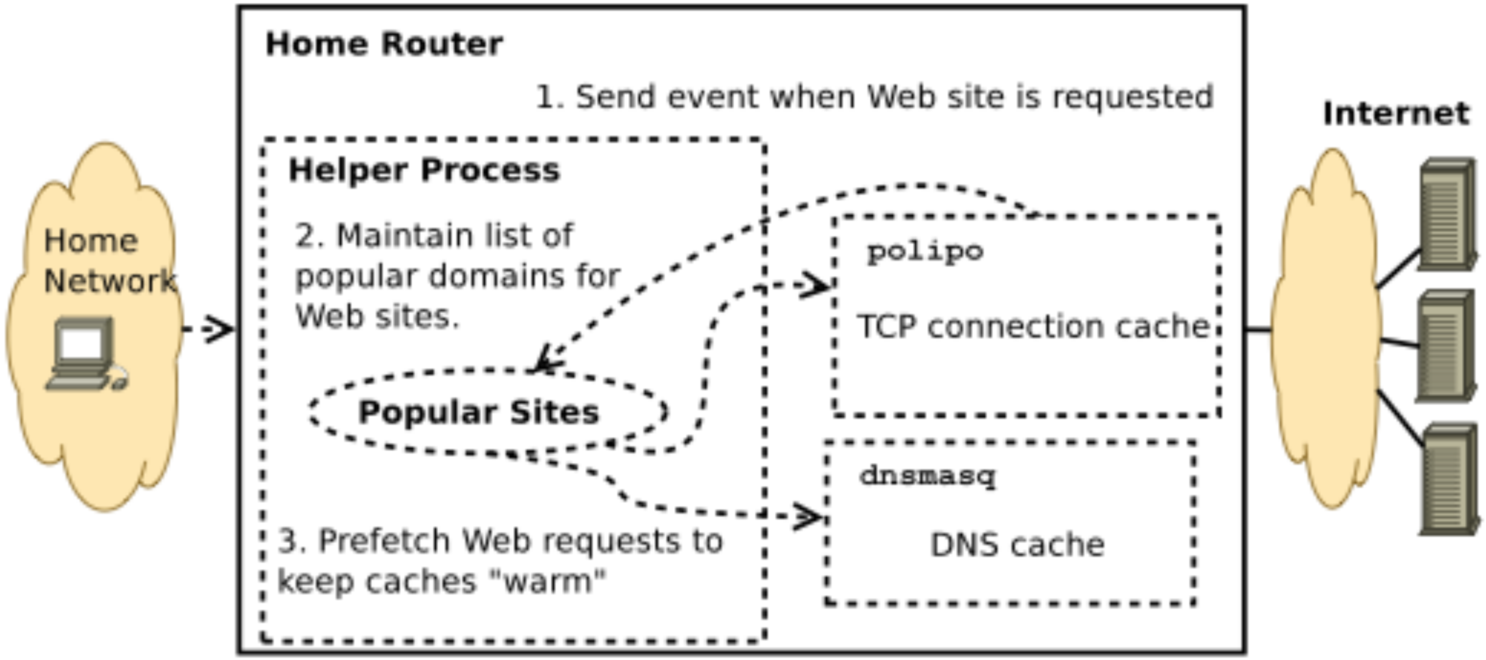


(a) Throughput.



(b) Latency.

Programmable Caching



Proactively caching content, connections, and DNS lookups can reduce page load times by 80%.
Even caching connections alone can reduce page load times by 40%.

Network Management Challenges

Challenge	Approach	System
Frequent Changes	Event-Based Network Control	Lithium
Minimal Visibility and Control	Programmable Measurement Platform	BISmark
Low-Level Configuration	High-Level Policy Language	Procera

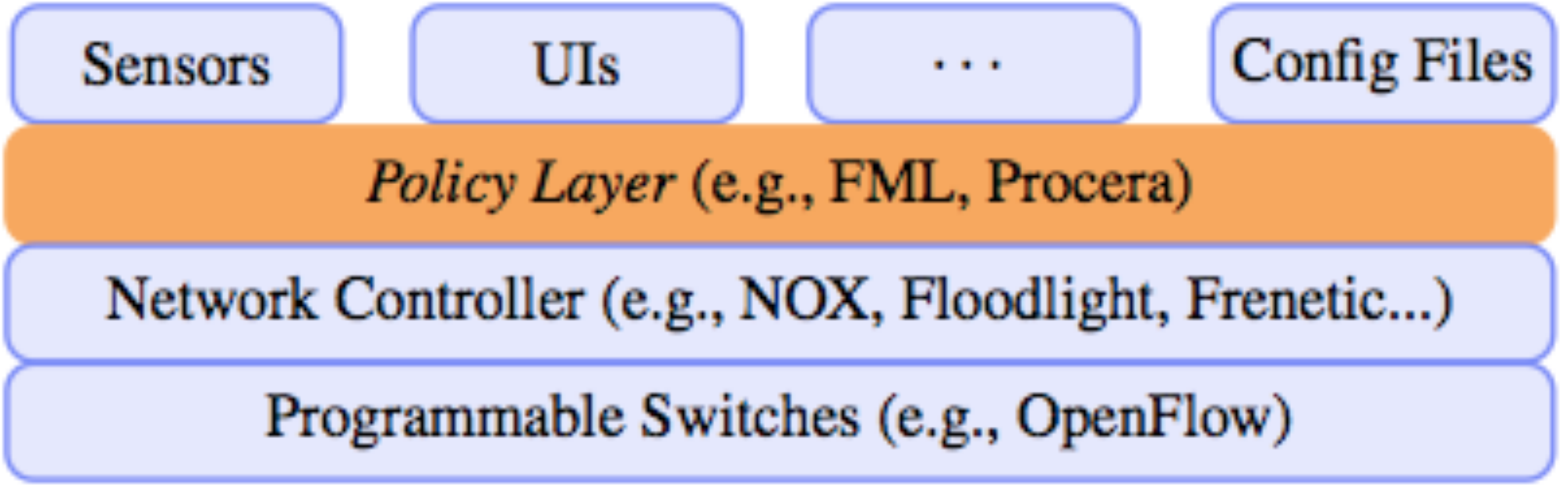
Language Design Requirements

- **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.

Procera Highlights

- **Flow constraint functions** that the network controller uses to constrain its own behavior
- **Functional reactive programming:** Declarative, expressive, compositional framework to describe reactive and temporal behaviors
- Customizable with a collection of primitive event streams

Procera System Architecture



- **Lowest layer:** Programmable Switches
- **Control layer:** Control and data
- **Policy layer:** “Supervisor” role
  - Process input signals
  - Events to network controllers

Why Procera?

- **Example:** Ban a device if its usage over the last five days exceeds 10 GB.
- Can't express this in existing languages, since FML doesn't incorporate usage data, provide arithmetic operations.
- So, extend FML with inequalities and a usage predicate?

```
deny(Us, D, As, Ut, Ht, At, P, R) <- over(D).
over(D) <- usage(D,T,B), T=5, B > 10.
```

- But, what about **permanent** bans? Requires *another* predicate!

Procera: Functional Reactive Programming

- Users describe time-varying values by describing how their current value depends on event histories and other values

$$
\begin{aligned}
&\mathbf{proc}\ \mathit{world} \rightarrow \mathbf{do} \\
&\quad \mathit{recent} \leftarrow \mathit{since}\ (\mathit{daysAgo}\ 5) \prec \mathit{add}\ (\mathit{usageEvents}\ \mathit{world}) \\
&\quad \mathit{usageTable} \leftarrow \mathit{group}\ \mathit{sum} \prec \mathit{recent} \\
&\quad \mathbf{returnA} \prec \mathit{usageTable}
\end{aligned}
$$

- Three components
  - Signals, functions, and events
  - Windowing and aggregation
  - Input signals and flow constraints

Constructs: Windowing & Aggregation

since dt	Windows a history to the past *dt* seconds
limit size	Windows a history to the last *size* number of events.
limitBy attribute size	Limits the input history by keeping only the last *size* number of events for each value of the attribute.
clockResetWindow next	Windowed history that is cleared at the time indicated by *next*.
accumList	Accumulates a sequence of events from a windowed history.
accumSet	Accumulates a set of events from a windowed history.
group op	Accumulates a dictionary from a history of key-value pairs, using *op* to combine values for the same key.
last1PerGroup	Accumulates a dictionary from a history, mapping keys to the last occurring value for the key.
add e,*remove e*, $ar_1 \oplus ar_2$	Event additions, removals, and combinations thereof.

Procera Constraints

- **Allow:** Forward the packet
- **Deny:** Do not allow the packet
- **Rate limit:** Rate limit to a given rate
- **Redirect:** Send to a specified host
- **Compose:** Constrain the flow according to multiple constraints

Procera Examples

- **Static Policy**

```
proc world → do
  returnA ⤙ λreq → allow
```

- **Device Registration**

```
proc world → do
  authDevs ← accumSet ⤙ add (authEvents world) ⊕
                          remove (deAuthEvents world)
  returnA ⤙
    λreq → if srcEthAddr req ‘member‘ authDevs
           then allow else deny
```

Summary

- Network management is difficult, and only becoming more difficult (and more important)
- Software-defined networking has made it possible to refactor network control
- We are exploring how to use SDN to simplify network management tasks
  - **Lithium:** Event-based network control
  - **BISmark:** Improved visibility and performance
  - Next steps: Languages