Lithium: Event-Driven Network Control

Nick Feamster, Hyojoon Kim, Russ Clark
Georgia Tech

Andreas Voellmy
Yale University

Software Defined Network *Management*

- Software defined networking (SDN) makes it easier for network operators to evolve network capabilities

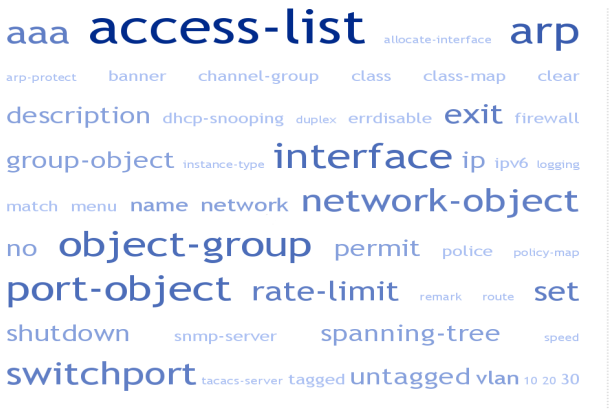
- Can SDN also help network operators manage their networks, once they are deployed?
  - Campus/Enterprise networks
  - Home networks

Big Problem: Configuration Changes Frequently

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

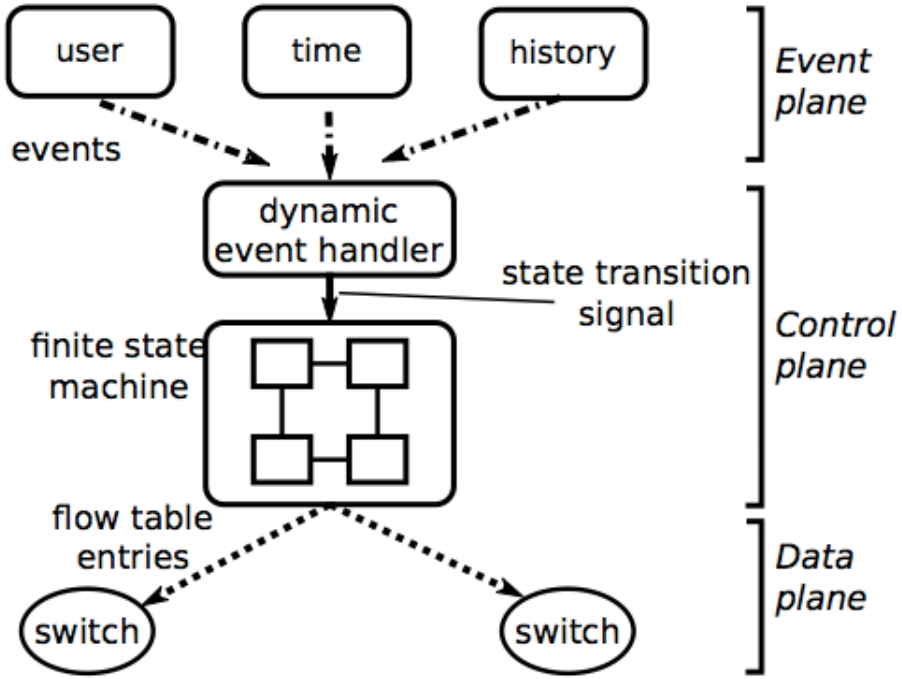


But, Network Configuration is Really Just Event Processing!

- 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

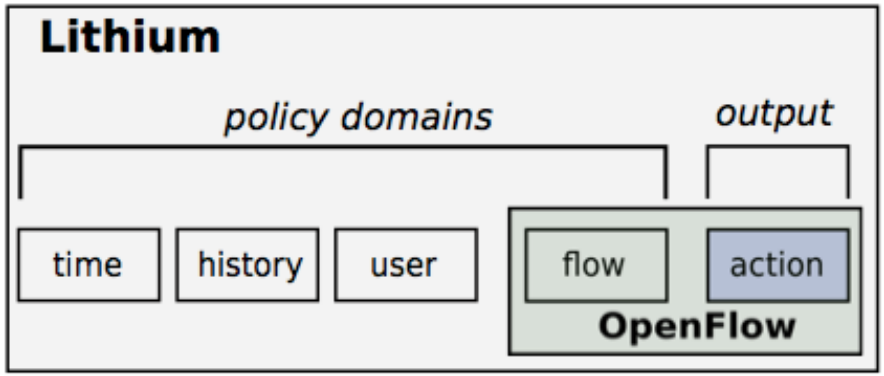
Main Idea: Express network policies as event-based programs.

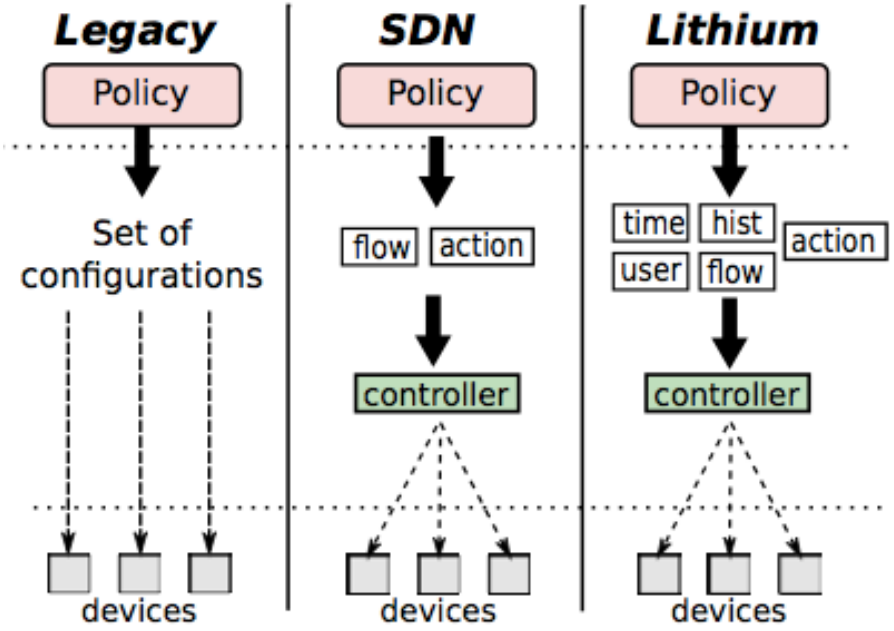


Resonance: Inference-Based Access Control for Enterprise Networks. Nayak, Reimers, Feamster, Clark. *ACM SIGCOMM Workshop on Enterprise Networks*. August 2009.

Extending the Control Model

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



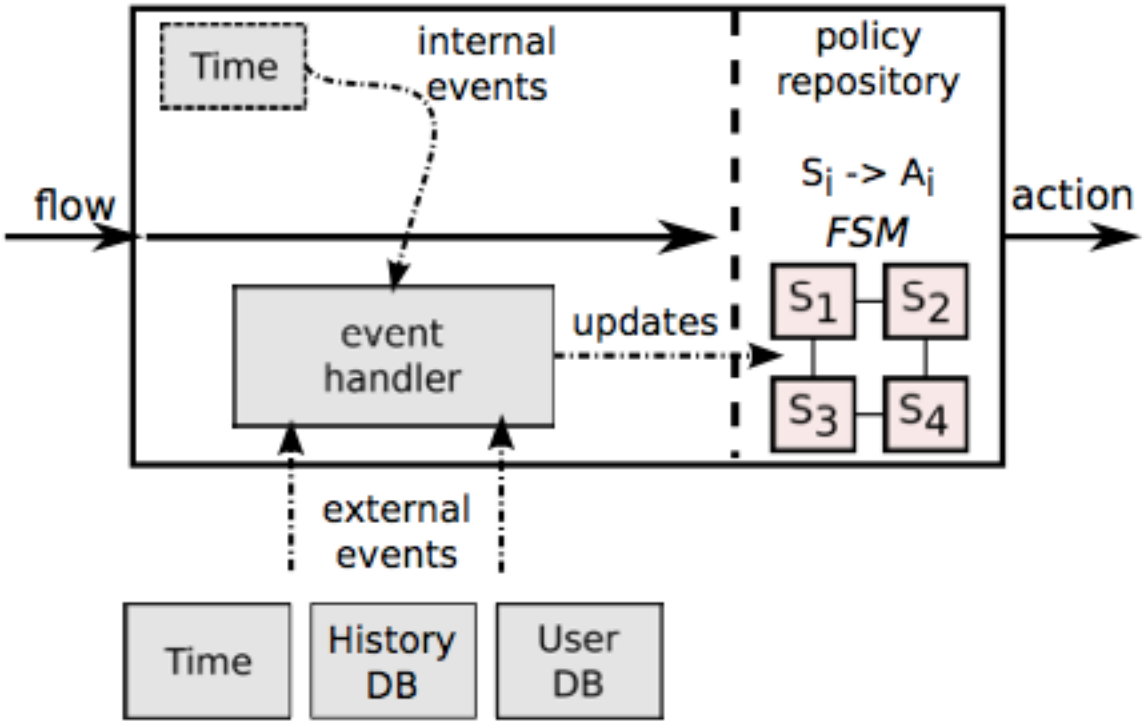


Event-Driven Control Domains

Domain: A condition on which traffic forwarding rules can be expressed.

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

Dynamic Event Handler



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

Representing Network State: 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

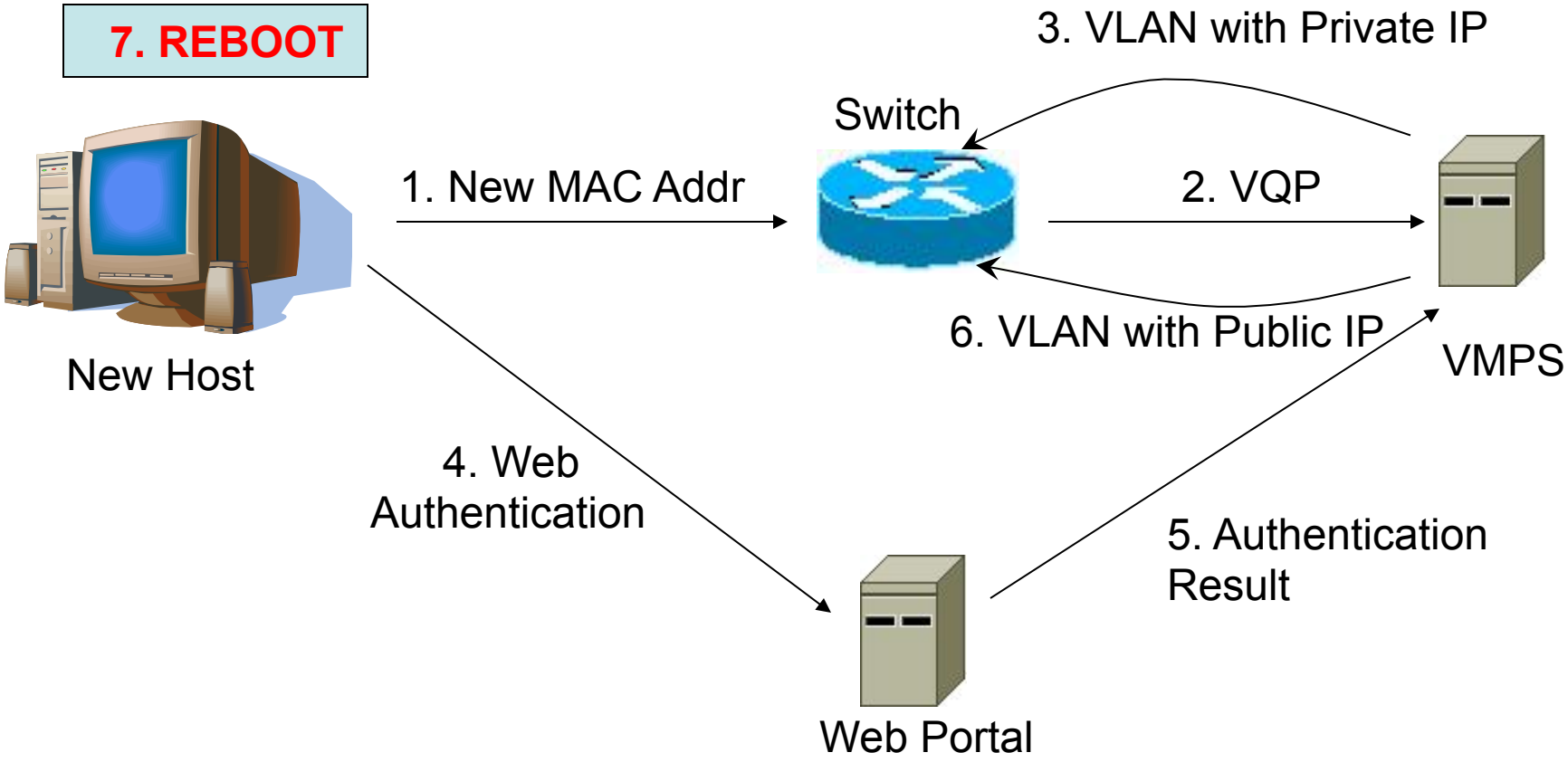
Current Implementation

- NOX Version 0.6.0
- **Lithium plumbing:** About 1,300 lines of C++
- **Policies**
  - Enterprise network access control: 145 lines of C++
  - Home network usage cap management: 130 lines
- **Ongoing:** Policies without general-purpose programming

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

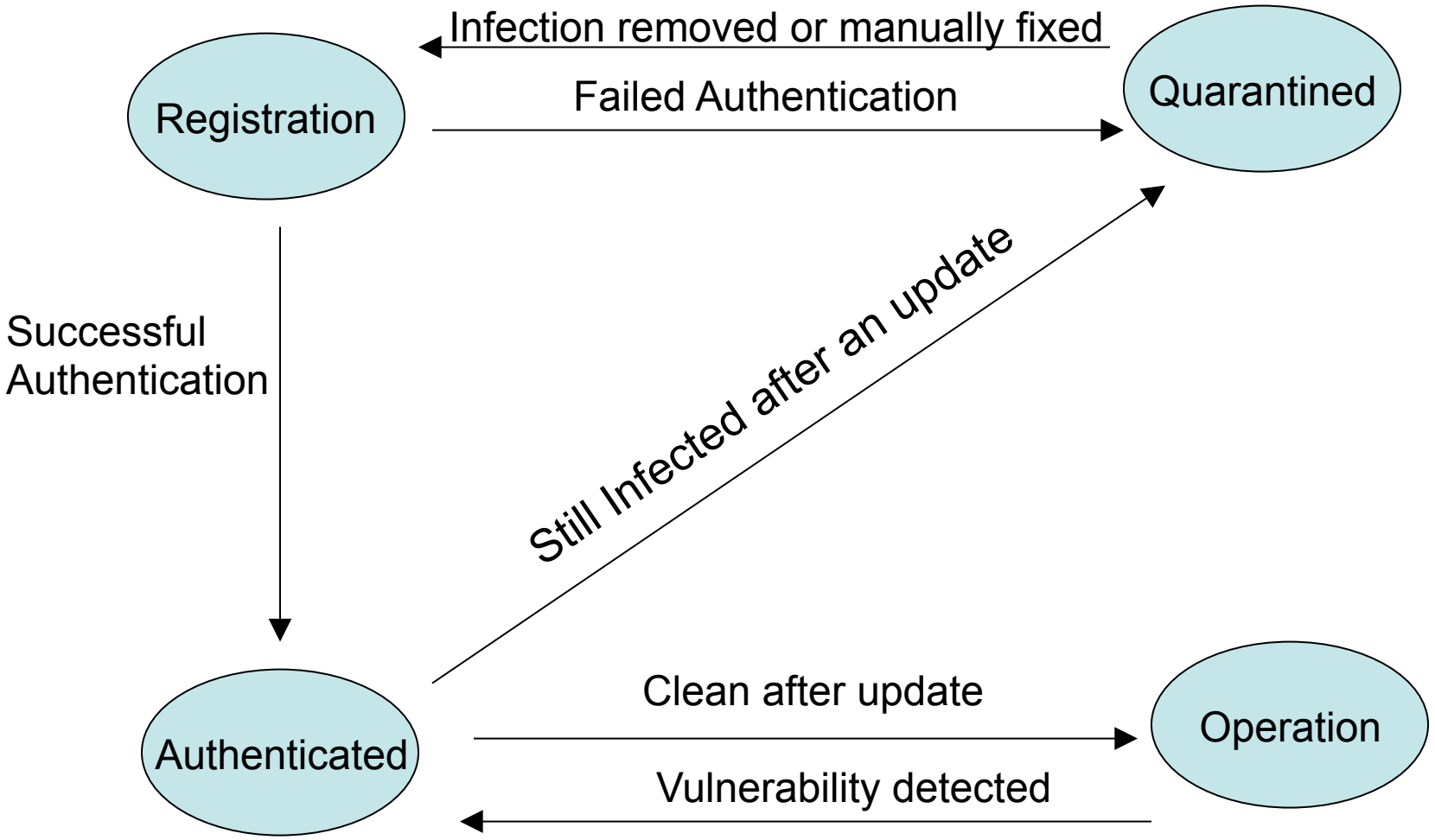
Example from Campus Network: Enterprise Access Control



Problems with Current Approach

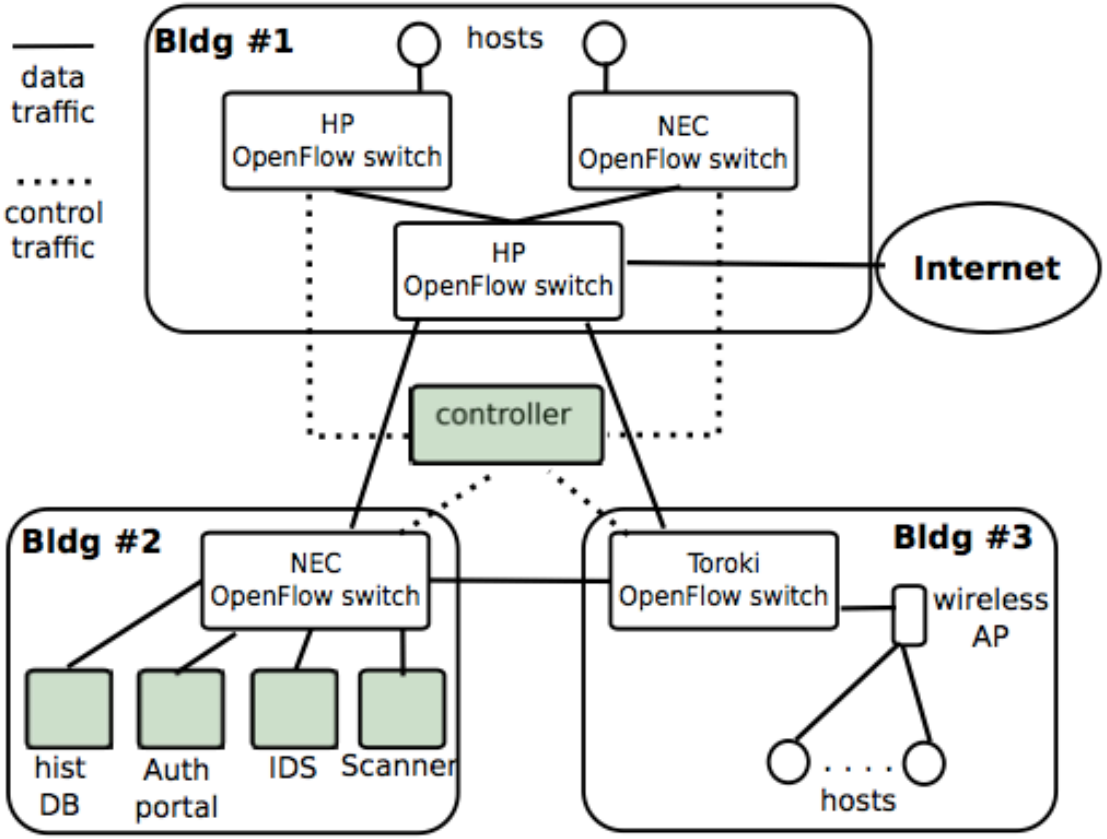
- Access control is **too coarse-grained**
  - Static, inflexible and prone to misconfigurations
  - Need to rely on VLANs to isolate infected machines
- **Cannot dynamically remap** hosts to different portions of the network
  - Needs a DHCP request which for a windows user would mean a reboot
- **Cannot automatically process changes** to network conditions.

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

Example From Home Networks: Usage Control

comcast

It's official: Comcast starts 250GB bandwidth caps October 1

By Jacqui Cheng | Published 3 years ago

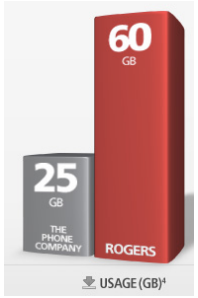
Comcast has announced that it will in fact be introducing bandwidth caps to all residential customers. The cap, which will go into effect as of October 1, will be 250GB per month. Comcast justifies the decision by saying that it's "an extremely large amount of data," and that a very large majority of customers will never cross it.

at&t

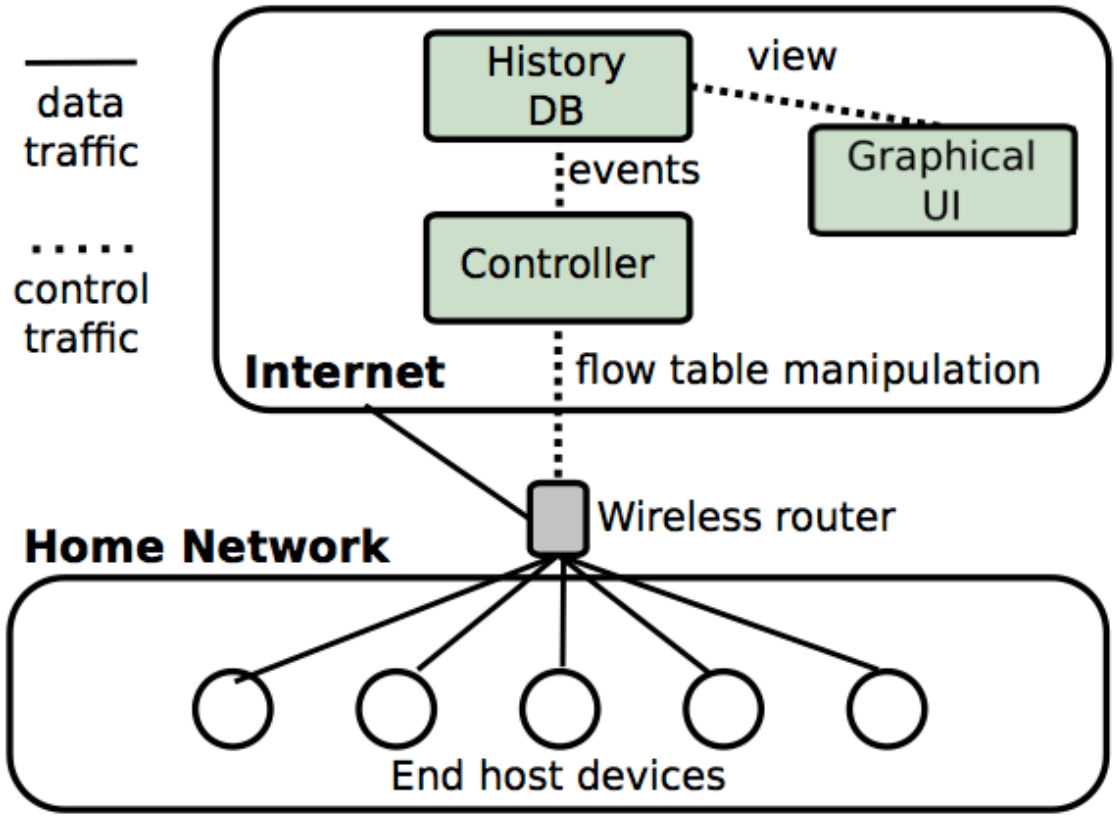
Is AT&T's new 150GB DSL data cap justified?

By Nate Anderson | Published 7 months ago

ROGERS

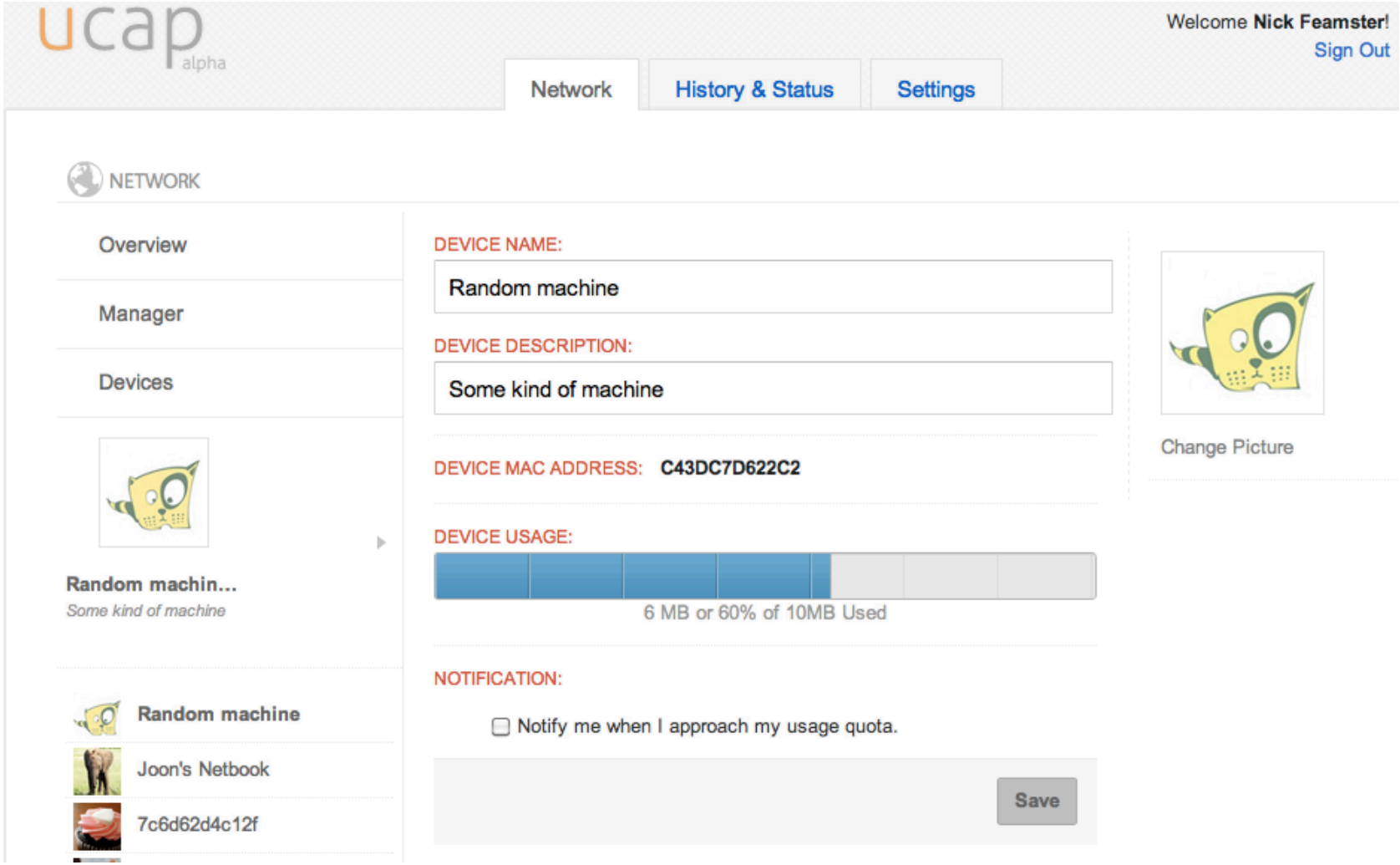


Deployment: Home Networks
(Outsourced Home Network Management)



- User monitors behavior and sets policies with UI (next slide)
- Lithium controller manages policies and router behavior

uCap: Intuitive Management Interface



Joint work with Boris de Souza, Bethany Sumner, Marshini Chetty.

Evaluation

- **Qualitative:** Is Lithium *usable*?
  - More expressive
  - Fewer “touches”
  - More general
  - More portable

- **Quantitative:** Is Lithium *feasible*?
  - Forwarding performance and latency
  - Flow table size
  - Load on controller and switches (scalability)

Quantitative Evaluation

- **Forwarding speed**
  - Negligible (< 1%) decrease in throughput for production switches
  - Forwarding speed on home routers is limited by the user-space OpenWrt implementation
- **Flow table entries**
  - In a large campus network, the number of flows is always less than 25,000 (commodity switches can support 130,000 entries already)
- **Controller load**
  - Significant under heavy traffic load, but can be mitigated by distributing the controller

Needed: High-Level Language

- Network policies
  - Are **dynamic**
  - Depend on **temporal conditions** defined in terms of external events

- Need a way to configure these policies **without resorting to general-purpose programming** of a network controller

- Intuitive user interfaces can ultimately be built on top of this language

Language Design Goals

- **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.

The Need for Reactive Control

- Simple policies are doable in FML: “Ban the device if usage exceeds 10 GB in the last 5 days”

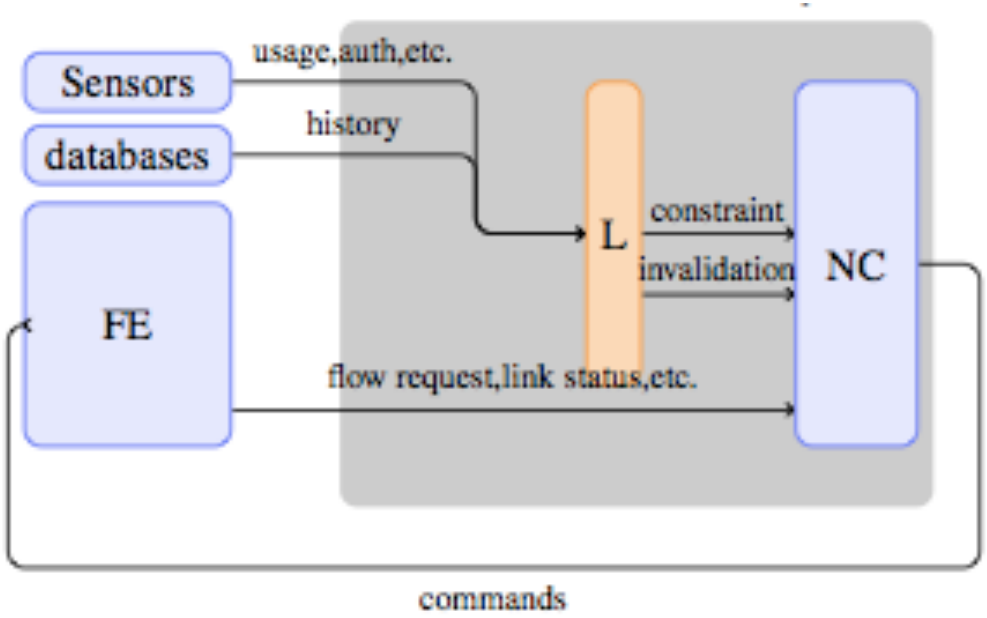
```
deny(Us, Hs, As, Ut, Ht, At, Prot, Req) <- over(Hs).
over(Hs) <- usage(Hs,lastDays(5),amt), amt > 10.
```

  - But, adding **temporal predicates** is difficult!
    - “Remove the ban if usage drops below 10 GB.”
    - “Remove the ban when an administrator resets.”

  - Each condition requires a new predicate.

```
over(Hs) <- usageOnceExceeded(Hs,lastDays(5),10).
```

Programming Reactive, Event-Based Network Control



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)
```

- **Controller:** signal functions and a flow constraint function
- Receives **input signals** from environment
- Periodically updates a **flow constraint function** that controls the forwarding elements

Summary

- Network management is difficult and error-prone
- **Lithium:** Event-based network control
- Deployment in two real-world settings (campus and home network)
- Developing a high-level control language that enables reactive control