SOFTWARE-DEFINED NETWORKING

Aditya Akella
Computer Sciences, UW-Madison
Why SDN?

Let’s start with a simple example..
Traditional Computer Networks

Forward, filter, buffer, mark, rate-limit, and measure packets

Data plane: Packet streaming
Traditional Computer Networks

Track topology changes, compute routes, install forwarding/filtering rules

Control plane:
Distributed algorithms
Traditional Computer Networks

Management plane: Human time scale

Collect measurements and configure the equipment
Shortest-Path Routing

- **Management**: set the link weights
- **Control**: compute shortest paths
- **Data**: forward packets to next hop
Inverting the Control Plane

- Traffic engineering
  - Change link weights
  - ... to induce the paths
  - ... that alleviate congestion
Transient Anomalies

- Distributed protocol
  - Temporary disagreement among the nodes
  - ... leaves packets stuck in loops
  - Even though the change was *planned*!
Other mgmt/control plane functions: access control, Quality-of-Service, overlays, service interposition, billing, DDoS protection

Non-routing state, managed using ad hoc mechanisms

Many boxes (routers, switches, firewalls, ...), with different interfaces.
What Ails the Network?

- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Distributed nature of control plane
- Ad hoc management approaches
- Slow protocol standardization

Impacts performance, security, reliability, cost...
Innovation is hard
SDN/OPENFLOW NETWORKS
Software Defined Networking

Logically-centralized control

Smart, slow

API to the data plane (e.g., OpenFlow)

Dumb, fast

Switches
Controller Architecture

Control Logic

Network graph and forwarding abstraction

State distribution mechanisms

Forwarding element integration

Northbound API

Southbound API

Events from switches:
- Topology changes,
- Traffic statistics,
- Arriving packets

Commands to switches:
- (Un)install rules,
- Query statistics,
- Send packets
Data-Plane: Simple Packet Handling

- Simple packet-handling rules
  - Pattern: match packet header bits
  - Actions: drop, forward, modify, send to controller
  - Priority: disambiguate overlapping patterns
  - Counters: #bytes and #packets

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.* → forward(2)
3. src=10.1.2.3, dest=.*.*.*.* → send to controller
The SDN Stack: More Detail

Applications
- Simple Switch
- CloudNaaS
- Stratos
- ...

Controller
- NOX
- Beacon
- Trema
- Maestro
- ...

Slicing Software
- FlowVisor
- FlowVisor Console

Commercial Switches
- HP, NEC, Pronto, Juniper, and many more

OpenFlow Switches
- Software Ref. Switch
- NetFPGA
- Broadcom Ref. Switch
- OpenWRT
- PCEngine WiFi AP
- Open vSwitch
Example SDN Applications

Wisconsin Projects
- Stratos
- CloudNaaS
- OpenNF

Commercial products
- Network virtualization: Nicira/VMWare, Azure, Google, CloudNaaS
- Traffic Engineering: Google’s B4, Microsoft’s SWAN

Public Demos
- Dynamic access control
- VM mobility/migration
- Network virtualization
- Power management
- Load balancing
- Traffic Engineering
Dynamic Access Control

- Inspect first packet of each connection
- Consult the access control policy
- Install rules to block or route traffic
Seamless Mobility/Migration

- See host sending traffic at new location
- Modify rules to reroute the traffic
SDN/OpenFlow in the Wild

- Open Networking Foundation
  - Creating Software Defined Networking standards
  - Google, Facebook, Microsoft, Yahoo, Verizon, Deutsche Telekom, and many other companies
- Commercial OpenFlow switches
  - Cisco, HP, NEC, Quanta, Dell, IBM, Juniper, ...
- Controllers/Languages
  - NOX, Beacon, Floodlight, Nettle, ONIX, POX,
  - Frenetic, MAPLE, Aspera, Pyretic
- Network deployments
  - Many campuses (including us), two research backbone networks
  - Commercial deployments
Software Defined Networking

- Simpler management and network control
  - No need to “invert” control-plane operations
- Faster pace of innovation
  - Less dependence on vendors and standards
  - Mechanism reuse
- Easier interoperability
  - Compatibility only in “wire” protocols
- Simpler, cheaper equipment
  - Minimal software
The End

Questions?
Abstractions in Networking

Layers decompose data delivery into tractable pieces

But... Layering abstractions deal mostly with "data plane"
What about Control Plane?

Data Plane: forwards packets given state

Control Plane: establishes state
- Distributed routing protocols
- Access control
- Quality-of-service
- Overlays
- Service interposition
- Billing
- DDoS protection
- ...
Hard to meet control requirements:

- Rely on ad hoc mgmt/config., distributed state exchange/processing

- Automated mgmt exist, but minimal mechanism reuse, composition is hard
Enter SDN

A unified approach to control plane management that embodies a set of clean abstractions so that rich control functions can be designed with minimal new design
Control Platform: handles state collection/distribution; hides complexity, heterogeneity

Rich Control Logics: distributed programs that read from/write to state at controller
SDN for Clouds

Example killer app: **Network virtualization**

- Multi-tenancy Control Logic
- Network graph and forwarding abstraction
- State distribution mechanisms
- Forwarding element integration

Control Platform Or “Controller”

Isolated virtual networks per tenant
SDN for Clouds

- Many “hot” startups
  - Better controllers
  - Controller/switch co-design
  - Improving network virtualization
  - Security
  - Other, rich control logics
  - <Insert your idea here>
Opportunities: Research

- Abstractions
  Currently: logical network graph. Too simplistic? What about data? Services?

- Control logics
  E.g.: Managing SLAs; security (DDoS protection)

- L3-L7 services
  API for tenants; composition
  How to provision? Scale? Manage (e.g., failures)?

- Broader issues
  Current: One-size-fit-all. But, needs differ, e.g., Facebook vs. Azure
  Interface with storage
Opportunities: Learn About SDN

- SDN boot camp
  - Read SDN literature: documentation, papers, blogs, white papers...
  - Play with NOX, Beacon, Floodlight
  - Write/run applications
  - Deploy/test on our OpenFlow testbed

- We’ll start small: ~6-8 students

- Regular meetings
  - Structured up front, not so much later
- akella@cs.wisc.edu or agember@cs.wisc.edu regarding boot camp
- Come see me regarding SDN research
The Internet: A Tremendous Success

- Brilliance of under-specifying
  - Network: best-effort packet delivery
  - Hosts: arbitrary applications

- Enables innovation in applications
  - Web, P2P, VoIP, social networks, virtual worlds

- But, change is easy only at the edge... 😞
Inside the Network?

- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces

- Over specified
  - Slow protocol standardization

- Few people can innovate
  - Equipment vendors write the code
  - Long delays to introduce new features

Impacts performance, security, reliability, cost...
Do We Need Innovation

Many boxes (routers, switches, firewalls, ...), with different interfaces.
How Hard are Networks to Manage?

- Operating a network is expensive
  - More than half the cost of a network
  - Yet, operator error causes most outages
- Buggy software in the equipment
  - Routers with 20+ million lines of code
  - Cascading failures, vulnerabilities, etc.
- The network is “in the way”
  - Especially a problem in data centers
  - ... and home networks
Shortest-Path Routing

- **Management**: set the link weights
- **Control**: compute shortest paths
- **Data**: forward packets to next hop