Lecture 6 Overview

- Project discussion
- Software Defined Networking overview
- Onix discussion
Evolution of Computing

Specialized Operating System

Specialized Hardware

Specialized Applications

Open Interface

Windows (OS) or Linux or MacOS

App

Open Interface

Microprocessor

CSE 222A – Lecture 6: Software Defined Networks
Evolution of Networking?

Specialized Features
Specialized Control Plane
Specialized Hardware

App
Control Plane
or
Open Interface
Control Plane
or
Control Plane
Open Interface
Merchant Switching Chips
Software Defined Networking

Abstract Network View

Network Virtualization

Global Network View

Network OS

Packet Forwarding

Control Programs

Packet Forwarding

Control Programs

Packet Forwarding

Control Programs

CSE 222A – Lecture 6: Software Defined Networks
OpenFlow

“If header = \(x\), send to port 4”
“If header = \(y\), overwrite header with \(z\), send to ports 5,6”
“If header = \(?\), send to me”
Basic OF Operations

- Match arbitrary bits in headers:
  - Match on any header, or new header
  - Allows any flow granularity

- Action
  - Forward to port(s), drop, send to controller
  - Overwrite header with mask, push or pop
  - Forward at specific bit-rate

Match: 1000x01xx0101001x
What is a Network OS?

- ONIX provides abstractions for network management

- Basic functionalities:
  - State distribution primitives between controllers and network elements.
  - Virtualized network elements

- A global view & control of the network
  - Very different from Active Networking vision (node local)
Onix Architecture

Server 1
Network Control Logic
NIB
Switch Import / Export
Distribution I / E

Server N
Network Control Logic
NIB
Distribution I / E
Switch Import / Export

Management Connectivity Network Infrastructure
Managed Physical Network Infrastructure
Design Requirements

- **Generality**: Support a wide range of network management applications
- **Scalability**: No inherent limitations due to the platform
- **Reliability**: Graceful failure handling
- **Performance**: Sufficient performance
Onix API

- Developers program against a network graph
- Nodes represent physical network entities
Network Information Base

- The NIB is the focal point of the system
  - State for applications to access
  - External state changes imported into it
  - Local state changes exported from it
Scalability

- Physical controller becomes a bottleneck:
  - Memory: to keep NIB
  - CPU and bandwidth: to process events

- This is a classic distributed systems problem
  - Partitioning
  - Aggregation
Partitioning For Scale

- Multiple dimensions available to applications:
  - Onix instances with different computations tasks
  - Onix instances have only subsets of the NIB
  - Switches connect to a subset of Onix instances
Aggregating for Scale

- Reduce fidelity of information before disseminating within the cluster
Aggregating for Scale

- Reduce fidelity of information before disseminating within the cluster
Consistency

- Onix provides two storage options
  - Replicated transactions (SQL) storage
  - One-hop memory-based DHT

- Do we need strong consistency for forwarding state between the controller and routers?

- Do we need strong consistency for NIB stored in controllers?

- Is it reasonable to ask applications for consistency preference and resolving conflicts?
Reliability

- **Network Element & Link Failures**: Applications' responsibility

- **Connectivity Infrastructure Failures**: Assumed reliable

- **Onix Failures**: Onix provides distributed coordination facilities provided for app failover
Discussion

- What are the security implications of Onix?
- Is Onix flexible and general enough for networking management?
- Are there applications you can run on a centralized network but not on a distributed one?
For Next Class...

- Read and review Vahdat’s Fat Tree paper

- Submit project groups by Friday morning
  - Email to Danny group members & idea(s)