Outline

- SDN - The Basic Idea and Definition
- Concept and Architecture
- OpenFlow
- Controller OPEN DAYLIGHT
- New Results and Summary
Traditional Networks

The 1st step

IP/MPLS Control Plane
- Distributed protocols and algorithms
- Logical separation from data plane, but in the same device

Data Plane Connectivity

Heterogeneous Data Plane

GMPLS Control Plane
- Distributed protocols and algorithm
- Central management functions
- Strict separation between control and data plane

Software Defined Networking (SDN) - First Idea

Logically Centralized Control
- Strict separation between control and data plane
- Smart control devices
- Very fast but cheap data plane switches

New Control
- Simpler management
- Less dependence on vendors
  - faster innovation
  - easier interoperability
- Simple and cheap equipment
  - minimal software
  - standard hardware

Open Flow (OF)
“API to the control”

Data Plane Connectivity

Heterogeneous Data Plane

API: application programming interface
SDN - The Principle -

- SDN is a new approach to networking
  - Not about “architecture”: IP, TCP, etc.
  - But about design of network control (routing, TE, ...)

- SDN is defined by three abstractions
  - Distribution
  - Forwarding
  - Configuration

---

SDN Controller Platform

Northbound API

Southbound API: Open interface to packet forwarding

Packet Forwarding

---

SDN Definition

- SDN is a framework to allow network administrators to automatically and dynamically manage and control a large number of network devices, services, topology, traffic paths, and packet handling (quality of service) policies using high-level languages and APIs.

- Management includes provisioning, operating, monitoring, optimizing, and managing (faults, configuration, accounting, performance, and security) in a multi-tenant environment.

Multi-tenancy: architecture in which a single instance of a software application serves multiple customers
Outline

- SDN - The Basic Idea and Definition
- Concept and Architecture
- OpenFlow
- Controller OPEN DAYLIGHT
- New Results and Summary

Network OS

- Consistent, up-to-date global network view

Control Program A  Control Program B

Network OS

Open interface to packet forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding
Network OS

Network OS
- Distributed system that creates a consistent, up-to-date global network view
  - Is responsible for the network state abstraction
  - Runs on servers in the network
  - Examples: OpenDayLight, Floodlight, POX, Pyretic, … + more

- Communicates with forwarding elements
  - Uses forwarding abstraction to:
    - Get state information from forwarding elements
    - Send control directives to forwarding elements

Forwarding Abstraction

- Purpose: Abstract away forwarding hardware details
  - Switches have two “brains”
    - Need a forwarding abstraction for both
      - Management CPU (smart but slow)
      - Forwarding ASIC (fast but dumb)

- Flexible forwarding function
  - Behavior specified by control plane
  - Built from basic set of forwarding primitives

- Minimal abstraction level
  - Streamlined for speed and low-power hardware
  - Allows the control program to be not vendor-specific

- OpenFlow is an example of such an abstraction
Control Program

- Network OS provides abstract view of global network map
- Control program operates on this abstract view of the network
  - **Input:** global network view (graph/database)
  - **Output:** configuration of each network device
- Control program should express desired behavior
  - It should not be responsible for implementing that behavior on physical network infrastructure
- Doesn’t have to be a distributed system, can be one centralized entity
  - Abstraction hides details of distributed network states
- Next step is to ease control program specification
  - Specification Abstraction using a specific Abstract Network View

Network Virtualization
Control: Specification Abstraction

- Natural abstraction: simplified abstract model of network
  - Simple model which only contains enough details to specify goals
  - Will give a control specific abstract view of the network

- Define a centralized control function which is working on the abstract network view
  \[ \text{Configuration} = \text{Function(specific abstract view)} \]

- Requires a new shared control layer:
  - which maps abstract configuration to physical configuration
  - usually called “Network Virtualization”

Network Virtualization

Specify behavior $f(\text{view1})$, $f(\text{view2})$, $f(\text{view3})$

Compile to topology Virtualization

Transmits to switches Network OS
Virtualization Examples

- Dynamic implementation of an Access Control List (ACL)
  - who can communicate to who (security policy)

- Data plane isolation:
  - who can hear my broadcasts

- Routing separation
  - Only specify routing to the degree you need for a service, e.g. some flows over satellite, others over landline

- Traffic Engineering:
  - specify service in terms of quality of service, not routes
  - Virtualization layer “compiles” these requirements
  - Produces suitable configuration of actual network devices
  - Network OS then transmits these settings to physical boxes

Outline

- SDN - The Basic Idea and Definition
- Concept and Architecture
- OpenFlow
- Controller OPEN DAYLIGHT
- New Results and Summary
Open Flow – Basic Architecture

- Data and Control path properties
  - + Speed, scale, fidelity of vendor hardware
  - + Flexibility and control of software
  - Vendors don’t need to expose implementation
  - Leverages hardware inside most switches today (ACL tables)

OpenFlow Controller

OpenFlow Protocol (SSL/TCP)

OpenFlow Protocol

Control Path (Software)

OpenFlow Client (Software)

Flow Switch

Data Path (Hardware)

Open Flow – Data plane forwarding functions

- Simple packet-handling rules: Match any packet header against flow table
- Allow any flow granularity
- Forward at specific bit rate
- Actions:
  - If header-bits = x → send to port 1
  - If header-bits = y → overwrite header with z, send to port 5 and 6
  - If header-bits = v → drop packet
  - If header-bits = "?" → send to controller
### Flow Routing vs. Aggregation

**Flow-Based**
- Every flow is individually set up by controller
- Exact-match flow entries
- Flow table contains one entry per flow
- Good for fine grain control, e.g. campus networks

**Aggregated**
- One flow entry covers large groups of flows
- Wildcard flow entries
- Flow table contains one entry per category of flows
- Good for large number of flows, e.g. backbone
Topology Discovery

Link Layer Discovery Protocol (LLDP)

- Open-Flow Controller sends LLDP frame to switches
- Switches send this frame over active data ports and to the controller
- Controller is able to discover the actual topology

Unknown Packet Flow

- First packet of flow has no entry in the flow table
  - Flow switch sends unknown packet as Packet_in type to controller
- Controller inspects packet and calculates path
- Controller sends FlowMod packet with flow table entries to all switches on the path
  - Dynamic assigned table entries have a restricted lifetime (soft state)
Controller Delay and Overhead

- Controller sends FlowMod packet with flow table entry and packet as Packet_out to initial switch
- Initial switch sends first and further packets on the established path

Drawback:
- Controller is much slower than the switch
  - Processing packets leads to delay and overhead
- Need to keep most packets in the fast path using static configuration

Reactive vs. Proactive

Reactive
- First packet of flow triggers controller to insert flow entries
- Efficient use of flow table
- Every flow incurs small additional flow setup time
- If control connection lost, switch has limited utility

Proactive
- Controller pre-populates flow table in switch
- Zero additional flow setup time
- Loss of control connection does not disrupt traffic
- Essentially requires aggregated (wildcard) rules
Centralized vs Distributed Control

- Centralized and distributed controller models are possible with OpenFlow
- Challenge: Controller placement and signaling load

**Centralized Control**

- Controller Application
- Network OS
- OpenFlow Switch
- Controller

**Distributed Control**

- Controller Application
- Network OS
- OpenFlow Switch
- Controller

Outline

- SDN - The Basic Idea and Definition
- Concept and Architecture
- OpenFlow
- Controller OPEN DAYLIGHT
- New Results and Summary
OPEN DAYLIGHT

- Not just a controller

  “The OpenDaylight Project is a collaborative open source project that aims to accelerate adoption of Software-Defined Networking (SDN) and create a solid foundation for Network Functions Virtualization (NFV) for a more transparent approach that fosters new innovation and reduces risk”

- Industry supported platform

  - Robust and extensible open source codebase
  - Common abstractions for northbound capabilities
  - Integration with OpenStack
  - Cloud applications
  - Production-level performance
  - Heavy industry support
Outline

- SDN - The Basic Idea and Definition
- Concept and Architecture
- OpenFlow
- Controller OPEN DAYLIGHT
- New Results and Summary
**SDN Limitations and Challenges**

- Limitation: Non-flow-based per-packet networking
  - Control of the per-packet next-hop selection in wireless mesh

- Limitation: New forwarding primitives
  - SDN provides a nice way to integrate them through extensions

- Limitation: New packet formats/field definitions
  - A generalized OpenFlow (2.0) is on the horizon to solve this

- Challenge: Optical Circuits
  - Effort is underway to apply OpenFlow model to circuits

- Challenge: Low-setup-time of individual flows
  - Can push down flows proactively to avoid delays

---

**SDN and Optical Space Division Multiplexing**

- A first experimental trial by N. Amaya et. Al.
  “Software defined networking (SDN) over space division multiplexing (SDM) optical networks: features, benefits and experimental demonstration”

---

**Acronyms**

- MCF: Multi-core Fibre
- BW: Bandwidth
- CF: Centre Frequency
- MF: Modulation Format
- DFB: laser bank
Questions?