Software Defined Networking Part 1

Introduction

1. Need for software defined networking
    - Separating control from data plane
    - Increasing challenges networks have been facing
    - What led to SDN?
    - How are SDN controllers architected?
2. Next SDN lecture
    - Data plane programmability
    - P4 language: A high-level language for protocol independent packet processors
    - SDN paradigm has been used for traffic engineering, security, and data center network applications
    - How would SDN work within an Internet Exchange Point?

What Led Us to SDN?

1. Motivation
    - SDN arose as part of the process to make computer networks more programmable
    - Computer networks are very complex and especially difficult to manage
2. Diversity of equipment
    - Wide range of equipment
        - Routers, switches, middleboxes (firewalls, NATs, load balancers, IDSs)
    - Network has to handle different software adhering to different protocols for each of these equipment
    - Network management is very complex
3. Proprietary technologies
    - Routers and switches tend to run software which is closed and proprietary
        - Configuration interfaces vary between vendors
    - These characteristics of computer networks made them highly complex, slow to innovate, and drove up the costs of running a network
    - SDN offers ways to redesign networks to make them more manageable
        - Simple idea: Separation of tasks
        - Divide network into control/data planes to simplify management and expedite innovation

Quiz 1

1. The main reason why SDNs were created was because of the increase of internet users.
    - False
2. SDNs divide the network in two planes: control plane and data plane, to ease management and speed up innovation.
    - True

A Brief History of SDN: The Milestones

1. The history of SDN can be divided into three phases:
    - Active networks
    - Control and data plane separation
    - OpenFlow API and network operating systems
2. Active Networks
    - Mid 1990s to early 2000s
    - Growth of Internet required standardization of new protocols by Internet Engineering Task Force (IETF)
    - Active Networks aimed to open up network control through a network API that exposed resources/network nodes and supported customization of functionalities for subsets of packets passing through the network nodes

    - Contrary to the idea that the simplicity of the network core was crucial to success
  - Two programming models
    - Capsule model: Carried in-band packets
    - Programmable router/switch model: Established by out-of-band mechanisms
  - Technology pushes:
    - Reduction in computation cost
    - Advancement in programming languages
    - Advances in rapid code compilation and formal methods
    - Funding from agencies (DARPA) for a collection of promoted interoperability among projects
  - Use pulls:
    - Network service provider frustration concerning the long timeline to develop and deploy new network services
    - Third party interests to add value by implementing more individual control
    - Researchers interest in having a network that would support large-scale experimentation
    - Unified control over middleboxes
  - Active network contributions to SDN
    - Programmable functions in the network to lower the barrier to innovation
    - Network virtualization, and the ability to demultiplex software programs based on packet headers
    - The vision of a unified architecture for middlebox orchestration

3. Control and Data Plane Separation
  - Lasted from 2001-2007
  - Steady increase in traffic volume so network reliability, predictability, and performance became more important
  - Technology pushes:
    - Higher link speeds in backbone networks led vendors to implement packet forwarding directly in the hardware, separating it from the control plane
    - ISPs found it hard to meet demands for reliability and new services (such as VPNs) and struggled to manage increasing size and scope
    - Servers had substantially more memory and processing resources so a single server could store all routing states and compute routing decisions for a large ISP network
    - Open source routing software lowered the barrier to creating prototype implementations of centralized routing controllers
  - Innovations
    - Open interface between control and data planes
    - Logically centralized control of the network
  - Differences from active networking
    - Focused on spurring innovation by and for network administrators rather than end users and researchers
    - Emphasized programmability in the control domain rather than the data domain
    - Worked towards network-wide visibility and control rather than device- level configurations
  - Use pulls:
    - Selecting between network paths based on the current traffic load
    - Minimizing disruptions during planned routing changes
    - Redirecting/dropping suspected attack traffic
    - Allowing customer networks more control over traffic flow
    - Offering value-added services for virtual private network customers
  - Contributions to SDN
    - Logically centralized control using an open interface to the data plane
    - Distributed state management

4. OpenFlow API and network operating systems
  - Lasted from 2007-2010
  - OpenFlow was born out of the interest in the idea of network experimentation at scale
    - Balance fully programmable networks and practicality of ensuring real world deployment

  - Built on existing hardware; limited its flexibility but enabled immediate deployment
  - Was adopted in the industry, unlike its predecessors
- Technology pushes:
  - Switch chipset vendors had already started to allow programmers to control some forwarding behaviors
  - More companies could build switches without having to design and fabricate their own data plane
  - Early OpenFlow versions built on technology that the switches already supported (just a firmware upgrade)
- Use pulls:
  - Meet the need of conducting large scale experimentation on network architectures
  - OpenFlow was useful in data-center networks for managing traffic at large scales
  - Companies invested in programmers to write control programs and less in proprietary switches that could not support new features easily
- Contributions to SDN:
  - Generalizing network devices and functions
  - Vision of a network operating system
  - Distributed state management techniques

Quiz 2

1. The Active Networks phase consisted mainly of creating a programming interface that exposed resources/network nodes and supported customization of functionalities for subsets of packets passing through the network.
   - True
2. One of the main differences between the Active Networks phase and the separation of the Control and Data plane phase is that the former is focused on network-wide visibility and control and the latter is focused on device-level configurations.
   - False
3. An OpenFlow switch has a table of packet-handling rules, and whenever it receives a packet, it determines the highest priority matching rule, performs the action associated with it and increments the respective counter.
   - True
4. One of the downfalls of OpenFlow when it was first created was that it was hard to deploy and scale it easily.
   - False

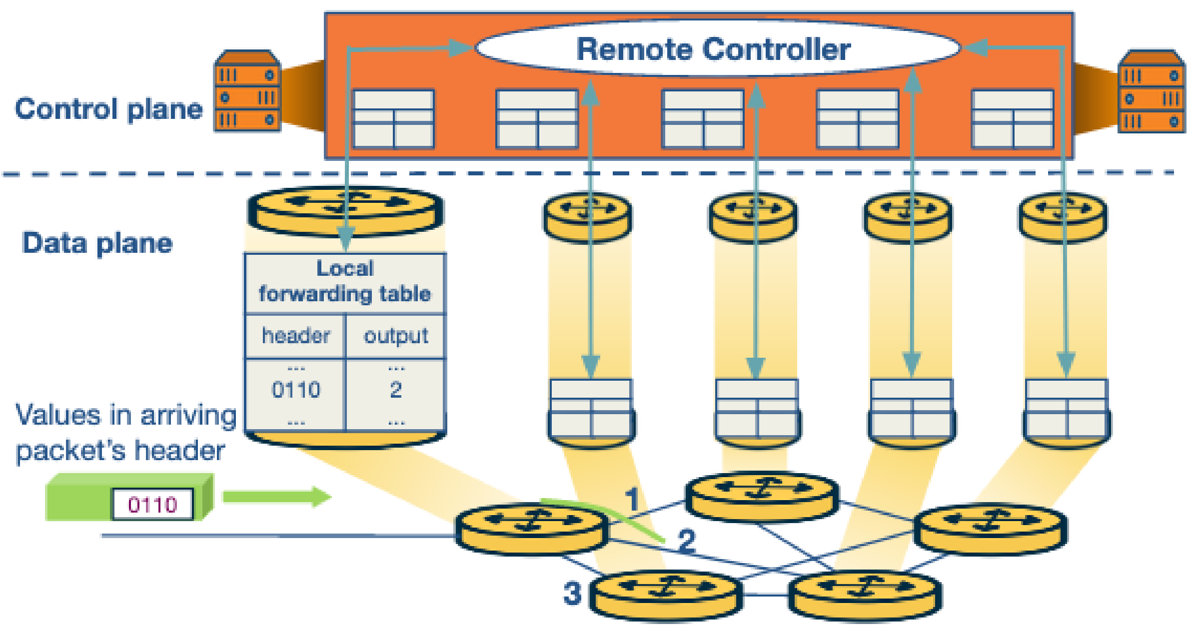
Why Separate the Data Plane from the Control Plane?

1. Control vs Data plane
   - Control plane contains logic that controls the forwarding behavior of routers such as routing protocols and network middlebox configurations
   - Data plane performs actual forwarding as dictated by the control plane
     - IP forwarding and layer 2 switching
2. Reasons for Separation
   - Independent evolution and development
     - Limiting interplay of routing and forwarding allows us to develop them more easily
   - Control from high-level software program
     - Allows for easier debugging and checking the behavior of the network
   - Software can develop independently from hardware
3. Other opportunities from separating
   - Data centers
     - SDN allows for easier management of thousands of servers and VMs
   - Routing

- BGP constrains routes; SDN allows for easier updating of the router's state and more control over path selection
- Enterprise networks
  - Easier to protect a network from DDoS attacks with SDN by dropping traffic at strategic locations
- Research networks
  - Can coexist with production networks

Control and Data Plane Separation

1. Functions of network layer
   - Forwarding
     - Router looks at header of an incoming packet and consults the forwarding table to determine the outgoing link to send the packet to
     - Implemented in hardware
     - Function of data plane
   - Routing
     - Determines the path from the sender to the receiver across the network
     - Implemented in software
     - Function of control plane
2. Control/Data Plane Separation
   - In the traditional approach, routing and forwarding are tightly coupled
   - In the SDN approach, a remote controller computes and distributes the forwarding tables to be used by every router
     - Physically separated from the router
   - Routers solely responsible for forwarding
   - Remote controller solely responsible for computing and distributing the forwarding tables



Remote Controller

Quiz 3

1. SDNs use software to control the routers' behavior (e.g., the path selection process).

2. With the separation of the control plane and the data plane, any change to the forwarding functions on a router is independent from the routing functions of the control plane.
    - True
3. In the SDN approach, the controller that computes and distributes the forwarding tables to be used by the routers is physically separate from the routers.
4. Software implementations in SDN controllers are increasingly open and publicly available, which speeds up innovation in the field.

The SDN Architecture

1. Components of and SDN network
    - SDN-controlled network elements
        - Called the infrastructure layer
        - Responsible for forwarding of traffic in a network based on the rules computed by the SDN control plane
    - SDN controller
        - Logically centralized entity that acts as an interface between the network elements and the network-control applications
    - Network-control applications
        - Programs that manage the underlying network by collecting information about the network elements with the help of the SDN controller
2. Defining Features in an SDN Architecture
    - Flow-based forwarding: Rules for forwarding packets can be computed based on any number of header field values in transport, network, or link layers
        - OpenFlow allows up to 11 header fields to be considered
    - Separation of data and control plane
    - Network control functions: Controller maintains information about network devices and the network-control applications monitor and control the devices
    - Programmable network: Network-control applications act as the brain of the SDN control plane by managing the network
        - Traffic engineering, security, automation, analytics, etc.

Quiz 4

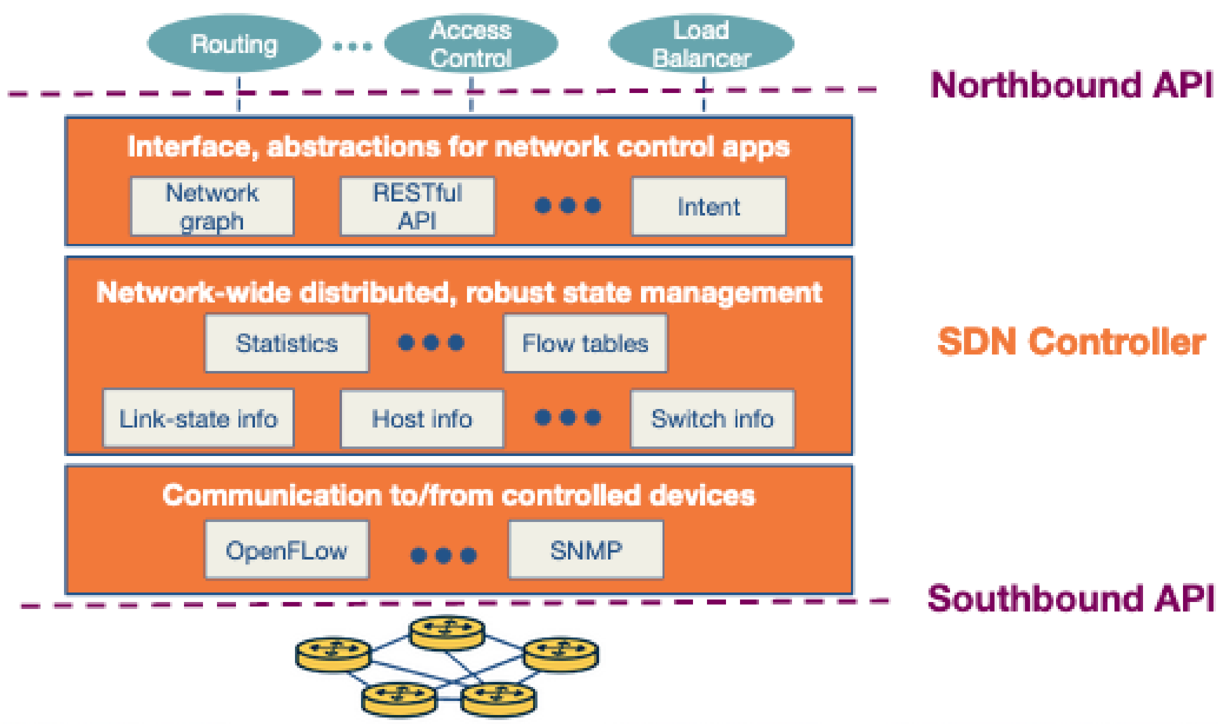
1. The network-control applications use the information about the network devices and elements, provided by the controller, to monitor and control the network devices.
    - True
2. In an SDN, the controller is responsible for the routing of the traffic, and the SDN-controlled network elements such as the switches are responsible for the forwarding of the traffic.
3. Traffic forwarding can be based on any number of header field values in various layers like the transport-layer, network-layer and link-layer.
    - True
4. SDN controllers operate on the control plane.

The SDN Controller Architecture

1. SDN Controller Layers
    - Communication layer
        - Communicating between controller and network elements
    - Network-wide state-management
        - Information about network state
    - Interface to the network-control application layer
        - Communicating between controller and applications

- SDN controller is implemented by distributed servers to achieve fault tolerance, high availability, and efficiency

2. Communication Layer
   - Protocol through which the SDN controller and network controlled elements communicate
   - Devices send locally observed events to the controller to provide a view of the current network state
     - Device joining/leaving the network
   - OpenFlow is an example of this protocol
   - Communication between the SDN controller and the controlled devices is known as the "southbound" interface
3. Network-wide State-Management Layer
   - Network-state is any information about the state of the hosts, links, switches, and other controlled elements in the network
4. Interface to the Network-Control Application Layer
   - "Northbound" interface
   - SDN controller interacts with network-control applications
   - Network-control applications can read/write network state and flow tables in the controller's state-management layer
   - REST interface is an example of a northbound API



Components of an SDN Controller

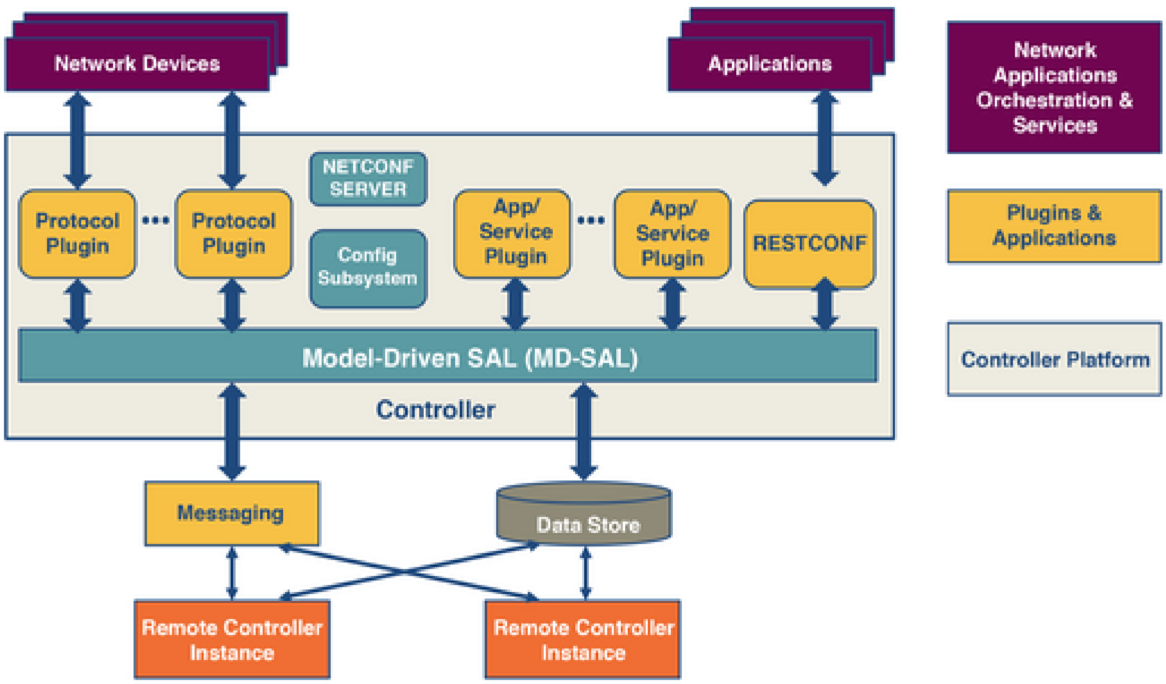
Quiz 5

1. The northbound interface is used by the controller and the network-control applications to interact with each other.
   - True
2. A REST interface is an example of a southbound API.
   - False

3. SDN controllers that are implemented by centralized servers are more likely to achieve fault tolerance, high availability and efficiency.
    - False

OpenDayLight Architecture Overview

1. Controller Architecture
    - Southbound interface: Controller uses to communicate with network devices
    - Northbound interace: SDN applications use to communicate with controller
    - Model Driven Service Abstraction Layer (MD-SAL): Abstraction layer provided by OpenStack for developers to add new features to the controller
        - Config datastore: Manages the representation of the network
        - Operation datastore: Has the true representation of the network state based on data from the managed network elements
        - Message bus provides a way for various services and protocol drivers to notify and communicate with each other



OpenDayLight