Introduction to Software Defined Networking (SDN)

Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@cse.wustl.edu

These slides and audio/video recordings of this class lecture are at:
http://www.cse.wustl.edu/~jain/cse570-15/

SDN = Separation of Control and Data Planes
SDN = Standard Southbound API
SDN = Centralization of control plane
SDN = OpenFlow
Overview

1. What is SDN?
2. SDN Controllers
3. Alternative APIs: XMPP, PCE, ForCES, ALTO
4. RESTful APIs and OSGi Framework

Note: This is the second module of three modules on OpenFlow, SDN and NFV in this course.
Origins of SDN

- SDN originated from OpenFlow
- Centralized Controller
  - Easy to program
  - Change routing policies on the fly
  - Software Defined Network (SDN)
- Initially, SDN=
  - Separation of Control and Data Plane
  - Centralization of Control
  - OpenFlow to talk to the data plane
- Now the definition has changed significantly.
Three Features that Define SDN

1. **Abstract the Hardware**: No dependence on physical infrastructure. Software API.
2. **Programmable**: Shift away from static manual operation to fully configurable and dynamic
3. **Centralized Control of Policies**: Policy delegation and management
What = Why We need SDN?

1. **Virtualization**: Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc. Abstraction ⇒ Virtualization.

2. **Orchestration**: Should be able to control and manage thousands of devices with one command.

3. **Programmable**: Should be able to change behavior on the fly.

4. **Dynamic Scaling**: Should be able to change size, quantity
   Virtualization ⇒ Scaling

5. **Automation**: To lower OpEx minimize manual involvement
   - Troubleshooting
   - Reduce downtime
   - Policy enforcement
   - Provisioning/Re-provisioning/Segmentation of resources
   - Add new workloads, sites, devices, and resources
Why We need SDN? (Cont)

6. **Visibility**: Monitor resources, connectivity

7. **Performance**: Optimize network device utilization
   - Traffic engineering/Bandwidth management
   - Capacity optimization
   - Load balancing
   - High utilization
   - Fast failure handling

8. **Multi-tenancy**: Tenants need complete control over their addresses, topology, and routing, security

9. **Service Integration**: Load balancers, firewalls, Intrusion Detection Systems (IDS), provisioned on demand and placed appropriately on the traffic path
Why We need SDN? (Cont)

10. **Openness**: Full choice of “How” mechanisms
   ⇒ Modular plug-ins
   ⇒ Abstraction:
      - Abstract = Summary = Essence = General Idea
        ⇒ Hide the details.
      - Also, abstract is opposite of concrete
        ⇒ Define tasks by APIs and not by how it should be done.
        E.g., send from A to B. Not OSPF.

Ref: Open Data Center Alliance Usage Model: Software Defined Networking Rev 1.0,”
Software Defined Anything (SDx)

- Tsunami of software defined things
  - Software Defined Networking (SDN)
  - Software Defined Datacenter (SDDC)
  - Software Defined Storage (SDS)
  - Software Defined Compute (SDC)
  - Software Defined Infrastructure (SDI)
Centralized vs. Distributed

- Fast Response to changes
- Fast Consistency
- Less overhead ⇒ Scalable
- Single Point of Failure

- Time to converge
- Slow consistency
- Not scalable
- Fault Tolerant
What SDN is Not?

- All of these are mechanisms.
- SDN is *not* about a mechanism.
- It is a framework ⇒ Many solutions
Four Confusions About SDN

1. Policies vs. Control:
Control = All bits and messages not sent by the user
In IP, control includes all header bits and all routing messages.

2. Separation of Control Plane:
Elements have only data plane and have no brains.

3. SDN vs. OpenFlow:
OpenFlow is the father of SDN but not SDN.

4. Need OpenFlow:
- OpenFlow is micro-management.
- It is not scalable.
- For large infrastructure, need scalable solutions.
Separation vs. Centralization

Separation of Control Plane

Centralization of Policies

Micromanagement is not scalable
Current SDN Debate: What vs. How?

1. SDN is easy if control is centralized but not necessary. Distributed/hierarchical solutions may be required for fail-safe operation.

2. Complete removal of control plane may be harmful. Exact division of control plane between centralized controller and distributed forwarders is yet to be worked out.
Current SDN Debate: What vs. How? (Cont)

3. SDN is easy with a standard southbound protocol like OpenFlow but one protocol may not work/scale in all cases
   1. Diversity of protocols is a fact of life.
   2. There are no standard operating systems, processors, routers, or Ethernet switches.
4. If industry finds an easier way to solve the same problems by another method, that method may win. E.g., ATM vs. MPLS.


### Flavors of SDN

1. **OpenDaylight**: Multi-Protocol Southbound
2. **Bare Metal Switches + Network Operating System**
   a. Switches from Dell, Edgecore, HP, Penguin, QCT, Agema, Supermicro
   b. Open Network Install Environment (ONIE)
   c. Network operating system: Alcatel-Lucent, Arista, Big Switch, Broadcom, Brocade, Cisco, Cumulus, Dell, Ericsson, Extreme, HP, Juniper, OCP, Pica8, Pluribus
3. **Network Virtualization/Overlay**: VMWare’s NSX
4. **ONF SDN**: OpenFlow southbound

All provide: Abstraction, Programmability, and Centralization


Source: Alan J Weissberger
Washington University in St. Louis
ONOS

- Open Network Operating System:
  Distributed OpenFlow OS for a large WAN
- 8-10 instances in a cluster.
  Each Instance responsible for a part of a network

Ref: [http://onosproject.org/](http://onosproject.org/)

Washington University in St. Louis

Ref: [http://www.cse.wustl.edu/~jain/cse570-15/](http://www.cse.wustl.edu/~jain/cse570-15/)

©2015 Raj Jain
Floodlight

- **Java** based OpenFlow controller based on Beacon runs within a JVM. Developers from Big Switch Networks
- **Indigo:** Software to make switch hardware OpenFlow compatible
- **Floodlight** is the core of Big Switch Controller from Big Switch Networks


Washington University in St. Louis
http://www.cse.wustl.edu/~jain/cse570-15/

©2015 Raj Jain
A number of real-world networking applications

- **Neutron plug-in** for OpenStack cloud management system
- **Static Flow Pusher**: Allows users to manually insert flows
- **Circuit Pusher**: Creates permanent entries on all switches along the path
- **Firewall**: Enforces access control list (ACL) rules on packets
- **Big Virtual Switch**: Automates network provisioning for a large scale data centers. Includes provisioning, multi-tenant partitioning

Ref: [http://www.projectfloodlight.org/floodlight/](http://www.projectfloodlight.org/floodlight/)
OpenDaylight: Multi-Protocol SDN

Northbound APIs
- DLUX GUI
- REST APIs

AAA Authentication Filter
- OSGi Framework
- OpenStack Neutron
- Neutron Northbound

Network Service Functions
- DOCSIS Svc
- Topology Mgr
- GBP Svc
- SDNI Aggregator
- AAA
- OVSDB Neutron

- Reservation
- Stats Mgr
- L2 Switch
- SFC
- OVSDB

- DIDM
- Switch Mgr
- LACP
- TSDR
- Neutron

- Topo Processing
- FRM
- LISP Svc
- USC Mgr
- VTN Mgr

Service Abstraction Layer (SAL)
- Plugin Mgr, Capability Abstractions, Flow Programming, Inventory, etc.

Southbound Protocol Plugins
- USC
- OVSDB
- CAPWEB
- CoAP
- HTTP
- LISP
- NETCONF
- PCEP
- OpenFlow
- ALTO
- BGP
- OPFLEX
- PCMM/COPS
- SNBI
- SNMP
- SXP

Network Elements
- Network Element
- Network Element
- Network Element

Overlay Tunnels (VxLAN, NVGRE, …)

Washington University in St. Louis

http://www.cse.wustl.edu/~jain/cse570-15/

©2015 Raj Jain
OpenDaylight SDN Controller Platform (OSCP)

- Multi-company collaboration under Linux foundation
- Many projects including OpenDaylight Controller
- **NO-OpenFlow** (Not Only OpenFlow): Supports multiple southbound protocols via plug-ins including OpenFlow
- Dynamically linked in to a Service Abstraction Layer (SAL) Abstraction ⇒ SAL figures out how to fulfill the service requested by higher layers irrespective of the southbound protocol
- Modular design using **OSGI framework**
- A rich set of North-bound APIs via **RESTful** services for loosely coupled applications and OSGI services for co-located applications using the same address space

Ref: [https://wiki.opendaylight.org/view/Main_Page](https://wiki.opendaylight.org/view/Main_Page)
Examples Alternative APIs

- Southbound APIs: XMPP (Juniper), OnePK (Cisco)
- Northbound APIs: I2RS, I2AEX, ALTO,
- Overlay: VxLAN, TRILL, LISP, STT, NVO3, PWE3, L2VPN, L3VPN
- Configuration API: NETCONF
- Controller: PCE, ForCES

XMPP

- Extensible Messaging and Presence Protocol
- **Extensible** ⇒ Using XML
- Similar to SMTP email protocol but for near real-time communication
- Each client has an ID, e.g., john@wustl.edu/mobile (John’s mobile phone)
- Client sets up a connection with the server ⇒ Client is online
- **Presence**: Server maintains contact addresses and may let other contacts know that this client is now on-line
- **Messaging**: When a client sends a “chat” message to another clients, it is forwarded to these other clients
- Messages are “pushed” (⇒ real-time) as opposed to “polled” as in SMTP/POP emails.

Washington University in St. Louis
http://www.cse.wustl.edu/~jain/cse570-15/ ©2015 Raj Jain
XMPP (Cont)

- XMPP is IETF standardization of Jabber protocol
- RFC 6121 defines XMPP using TCP connections. But HTTP is often used as transport to navigate firewalls
- All messages are XML encoded
  - Not efficient for binary file transfers
  - Out-of-band binary channels are often used with XMPP.
- A number of open-source implementations are available
- Variations of it are widely used in most instant messaging programs including Google, Skype, Facebook, …, many games
- Used in IoT and data centers for management. Network devices have XMPP clients that respond to XMPP messages containing CLI management requests ⇒ You can manage your network using any other XMPP client, e.g., your mobile phone
- Arista switches can be managed by XMPP, Juniper uses XMPP as a southbound protocol for SDN

Ref: http://en.wikipedia.org/wiki/XMPP
Washington University in St. Louis http://www.cse.wustl.edu/~jain/cse570-15/ ©2015 Raj Jain
XMPP in Data Centers

- Everything is an XMPP entity.
  It has its own contact list and authorizations.

Ref: https://github.com/ArchipelProject/Archipel/wiki/Architecture-%26-Concepts

Washington University in St. Louis  http://www.cse.wustl.edu/~jain/cse570-15/ ©2015 Raj Jain
Path Computation Element (PCE)

- MPLS and GMPLS require originating routers to find paths that satisfy multiple constraints including not using any backup routers and having a given bandwidth etc.
- This may require more computer power or network knowledge than a router may have.
- IETF PCE working group has developed a set of protocols that allow a Path computation client (PCC), i.e., router to get the path from path computation element (PCE)
- PCE may be centralized or may be distributed in many or every router.

What is the 1 Gbps route to New York not going through Boston?
PCE (Cont)

- PCE separates the route computation function from the forwarding function.
- Both functions may be resident in the same box or different boxes.
- 25+ RFCs documenting protocols for:
  - PCE-to-PCC communication
  - PCE-to-PCE communication (Multiple PCEs)
  - PCE discovery

Ref: http://datatracker.ietf.org/wg/pce/
Ref: http://en.wikipedia.org/wiki/Path_computation_element
Washington University in St. Louis http://www.cse.wustl.edu/~jain/cse570-15/
**Forwarding and Control Element Separation (ForCES)**

- IETF working group since July 2001
- Control Elements (CEs) prepare the routing table for use by forwarding elements (FEs).
- Each CE may interact with one or more FEs
- There may be many CEs and FEs managed by a CE manager and a FE manager
ForCES (Cont)

- Idea of control and data plane separation was used in BSD 4.4 routing sockets in early 1990s. It allowed routing tables to be controlled by a simple command line or by a route daemon.

- ForCES protocol supports exchange of:
  - Port type, link speed, IP address
  - IPv4/IPv6 unicast/multicast forwarding
  - QoS including metering, policing, shaping, and queueing
  - Packet classification
  - High-touch functions, e.g., Network Address Translation (NAT), Application-level Gateways (ALG)
  - Encryptions to be applied to packets
  - Measurement and reporting of per-flow traffic information

Ref: http://datatracker.ietf.org/doc/rfc3654/?include_text=1
Sample ForCES Exchanges

Ref: http://datatracker.ietf.org/doc/rfc3746/?include_text=1
Application Layer Traffic Optimization (ALTO)

- IETF working group to optimize P2P traffic
  ⇒ Better to get files from nearby peers
- Provide guidance in peer selection
- ALTO Server: Has knowledge of distributed resources
- ALTO Client: Requests information from servers about the appropriate peers
- Ratio Criteria: Topological distance, traffic charges, …
- ALTO Server could get information from providers or from nodes about their characteristics, e.g., flat-rate or volume based charging
- A client may get the list of potential peers and send it to the server, which can return a ordered list
- Also need a protocol for ALTO server discovery

Ref: J. Seedorf and E. Berger, “ALTO Problem Statement,” http://datatracker.ietf.org/doc/rfc5693/?include_text=1

Washington University in St. Louis - http://www.cse.wustl.edu/~jain/cse570-15/
ALTO Extension

- Now being extended to locate resources in data centers
- Need to be able to express
  - resource (memory, storage, CPU, network) availability
  - Cost of these resources
  - Constraints on resources, e.g., bandwidth
  - Constraints on structure, e.g., Power consumption
- ALTO client gets the info from various providers
- Issue of privacy of resource and cost info for the provider
OpenDaylight Tools

1. **Applications**: Provides Virtual Network Segments (VNS) for each tenant
   1. OpenDaylight Network Virtualization (ONV):
   2. OpenDaylight Virtual Tenant Network (VTN)

2. **Services**:
   1. Defense4All: Security

3. **Northbound APIs**:
   1. REST
   2. **Dlux**: Northbound API using AngularJS, an extension of HTML by Google for dynamic views
OpenDaylight Tools (Cont)

4. **Southbound APIs:**
   1. OpenFlow Plug-in + Protocol Library (V1.0, V1.1,…)
   2. Locator ID Separation Protocol (LISP) Mapping Service
   3. SNMP4SDN
   4. BGP Link State Path Control Element Protocol

5. **Overlay:**
   1. Open Distributed Overlay Virtual Ethernet (DOVE):
      Like VxLAN but does not use IP Multicast

6. **Configuration:**
   1. OpenDaylight YANG Tools: NETCONF
   2. Open vSwitch Database (OVSDB) Integration
   3. Affinity Metadata Service
Affinity Metadata Service

- API to create an abstract topology and implementation independent description of infrastructure needs and behaviors of network workloads
- Allows intent to be specified in application and service terms independent of where and how the workloads attach to the network.
- SDN controllers and application can use “affinity” information to automatically provision the VMs and network for the user
- Users don’t need to know about bridges, routers, VLANs, and tunnels

Ref: https://wiki.opendaylight.org/view/Project_Proposals:Affinity_Metadata_Service
Washington University in St. Louis http://www.cse.wustl.edu/~jain/cse570-15/ ©2015 Raj Jain
Key SDN Related Software

- Mininet (Current)
- OpenVirteX
- Ryu (current)
- Trema
- RouteFlow (Last commit March 19, 2014)
- Luxoft Twister
Mininet

- Widely used open source network emulation environment.
- Can simulate a number of end-hosts, switches, routers, links on a Linux.
- Used for rapid prototyping of software define networks.
- Built-in Open vSwitch, and a OpenFlow capable switch.
- Command line launcher and Python API for creating networks of varying sizes, e.g., `mn --topo tree,depth=2,fanout=3`.
- Useful diagnostic commands like iperf, ping, and other commands in a host, e.g., `mininet> h11 ifconfig -a`.
- Mininet code for several popular commercial switches are available.

Ref: [https://github.com/mininet/mininet](https://github.com/mininet/mininet)
OpenVirteX (OVX)

- Transparent Proxy between OpenFlow switches and multiple OpenFlow Controllers. Slices defined by header fields.
- Creates network slices that can be managed by different controllers ⇒ Isolates slices from each other
- All control traffic goes through OVX ⇒ Slight latency

Ref: http://ovx.onlab.us/
http://www.cse.wustl.edu/~jain/cse570-15/
Ryu

- Component-based framework that integrates with OpenStack and supports OpenFlow
- Provides software component with well defined API for network management and control applications
- Supports various versions of OpenFlow, OF-Config, Nicira extensions
- Developed by NTT laboratories
- Can easily setup a multi-node OpenStack environment using pre-configured Ryu VM image file

Ref: [http://osrg.github.io/ryu/](http://osrg.github.io/ryu/)
©2015 Raj Jain
Trema

- Full-stack easy-to-use framework for developing OpenFlow controllers in **Ruby and C**
- Open source. Developed by NEC Research Lab.
- Modular extensible architecture
- Integrated development environment for testing and debugging

Ref: [http://github.com/trema/](http://github.com/trema/)
Ref: [http://trema.github.com/trema/](http://trema.github.com/trema/)
RouteFlow

- Provides virtualized IP routing services over OpenFlow enabled hardware
- IP routing engines (e.g., Quagga) in the networking devices generate the forwarding information base (FIB) into the Linux IP tables using OSPF, BGP, etc.
- RouteFlow Client processes collect the IP and ARP tables and translate into OpenFlow tuples that are installed in the OpenFlow devices in the forwarding plane

Ref: https://sites.google.com/site/routeflow/home

Washington University in St. Louis

http://www.cse.wustl.edu/~jain/cse570-15/

©2015 Raj Jain
RouteFlow (Cont)

- Key components: RouteFlow Client, RouteFlow Server, and RouteFlow Proxy

Ref: https://sites.google.com/site/routeflow/home

Washington University in St. Louis

http://www.cse.wustl.edu/~jain/cse570-15/

©2015 Raj Jain
Luxoft Twister

- Test automation framework to manage and drive test cases written in shell scripting languages.
- Supports TCL, Python, and Perl
- Web-based user interface
- Remote access capability

Ref: http://github.com/Luxoft/Twister
Open Source Routing Software

- Bird Internet Routing Daemon (BIRD):
  - TCP/IP routing daemon for Unix-like systems
  - Developed at Charles University, Prague
  - Provides BGP, RIP, OSPF for IPv4 and IPv6
  - Included in many Linux distributions
  - Used in several internet exchanges as a route server and has replaced Quagga because of its scalability issues

- Quagga: Includes OSPF, RIP, BGP, IS-IS on Unix-like OSs

- eXensible Open Router Platform (XORP):
  - Designed at ICSI in Berkeley
  - Supports OSPF, BGP, RIP, PIM, IGMP, OLSR
  - Generally replace by Quagga

RESTful APIs

- Software architecture style developed by W3C.
- Introduced by Roy Fielding in his PhD thesis.
- WWW uses this style. Very popular in other applications.
- Goals: Scalability, Generality, Independence, and allow intermediate components
- Client-Server Model: Clients and servers can be developed undependably.
- Server is stateless
- Responses can be cached for the specified time
- Intermediate Servers (Proxies) can respond. End point is not critical.
REST (Cont)

- Create, Read, Update, Delete (CRUD) Operations
- Uniform Interface: GET (Read), POST (Insert), PUT (write), DELETE
- Resources identified by global identifiers, e.g., URI in Web.
  E.g., GET http://odcp.org/rest/v1/model/controller-node
- Data Types: Controller node, Firewall rule, Topology configuration, Switch, Port, link, flow entry, VLAN, ...
- Data types can include commercial entities, such as, Big Virtual Switch from Big Switch Networks, vCenter from VMware, ...
- If optional-id and query parameters are omitted, the returned text includes all of the items of the given data type.

Ref: http://en.wikipedia.org/wiki/Representational_state_transfer
OSGi Framework

- Initially, Open Services Gateway initiative
- A set of specifications for dynamic application composition using reusable Java components called bundles
- Bundles publish their services with OSGi services registry and can find/use services of other bundles

Ref: [http://www.osgi.org/Technology/WhatIsOSGi](http://www.osgi.org/Technology/WhatIsOSGi)
Bundles can be installed, started, stopped, updated or uninstalled using a lifecycle API.

Modules defines how a bundle can import/export code.

Security layer handles security.

Execution environment defines what methods and classes are available in a specific platform.

A bundle can get a service or it can listen for a service to appear or disappear.

Each service has properties that allow others to select among multiple bundles offering the same service.

Services are dynamic. A bundle can decide to withdraw its service. Other bundles should stop using it.

⇒ Bundles can be installed and uninstalled on the fly.
Summary

1. SDN = Abstraction + Programmability + Centralization
2. OpenFlow originated SDN but now many different southbound and northbound APIs, intermediate services and tools are being discussed and implemented by the industry, e.g., XMPP, ForCES, PCE, ALTO
3. OpenDaylight, ONOS, and FloodLight are SDN Controllers. Differ on how much open.
4. Mininet for network simulation
5. REST = HTTP APIs
   OSGI framework for modularity
Reading List

- J. Seedorf and E. Berger, “ALTO Problem Statement,” http://datatracker.ietf.org/doc/rfc5693/?include_text=1
- http://www.osgi.org/Technology/WhatIsOSGi
- http://www.sdncentral.com/sdn-use-cases/
- http://datatracker.ietf.org/wg/pce/
- https://wiki.opendaylight.org/view/Main_Page
Wikipedia Links

- http://en.wikipedia.org/wiki/Path_computation_element
References


- OpenDaylight Components and Tools:
  - https://wiki.opendaylight.org/view/OpenDaylight_Network_Virtualization_%28ONV%29:Main
  - https://wiki.opendaylight.org/view/OVSDB_Integration:Design
  - https://wiki.opendaylight.org/view/Project_Proposals:Affinity_Metadata_Service
  - https://wiki.opendaylight.org/view/Project_Proposals:BGP_and_PCEP
  - https://wiki.opendaylight.org/view/Project_Proposals:Defense4All
  - https://wiki.opendaylight.org/view/Project_Proposals:Dlux
References (Cont)

- https://wiki.opendaylight.org/view/Project_Proposals:SNMP4SDN
- https://wiki.opendaylight.org/view/YANG_Tools:Main
- Open Data Center Alliance Usage Model: Software Defined Networking Rev 1.0,”
## Acronyms

- **ACI**  Application Policy Infrastructure
- **ACL**  Access Control List
- **AEX**  Application Information Exposure
- **ALG**  Application Level Gateway
- **ALTO**  Application Layer Traffic Optimization
- **ANDSF**  Access Network Discovery and Selection Function
- **API**  Application Programming Interface
- **APIC**  Application Policy Infrastructure Controller
- **ARP**  Address Resolution Protocol
- **ATIS**  Association for Telecom Industry Solutions
- **ATM**  Asynchronous Transfer Mode
- **AVNP**  Active Virtual Network Management Protocol
- **BGP**  Border Gateway Protocol
- **BNC**  Big Switch Network Controller
- **BSD**  Berkeley Software Distribution
- **BUM**  Broadcast, Unknown, and Multicast
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDN</td>
<td>Content Distribution Network</td>
</tr>
<tr>
<td>CDNI</td>
<td>Content Distribution Network Interconnection</td>
</tr>
<tr>
<td>CE</td>
<td>Control Element</td>
</tr>
<tr>
<td>CLI</td>
<td>Command Line Interface</td>
</tr>
<tr>
<td>CMS</td>
<td>Content Management System</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CRUD</td>
<td>Create, Read, Update, Delete</td>
</tr>
<tr>
<td>CSP</td>
<td>Cloud Service Provider</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Control Protocol</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>DOVE</td>
<td>Distributed Overlay Virtual Ethernet</td>
</tr>
<tr>
<td>DVS</td>
<td>Distributed Virtual Switch</td>
</tr>
<tr>
<td>EID</td>
<td>Endpoint Identifier</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Institute</td>
</tr>
<tr>
<td>FCAPS</td>
<td>Faults, configuration, accounting, performance, and security</td>
</tr>
<tr>
<td>FE</td>
<td>Forwarding Element</td>
</tr>
<tr>
<td>Acronyms (Cont)</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>FE</td>
<td>Forwarding Element</td>
</tr>
<tr>
<td>ForCES</td>
<td>Forwarding and Control Element Separation</td>
</tr>
<tr>
<td>GMPLS</td>
<td>Generalized Multi-Protocol Label Switching</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>I2AEX</td>
<td>Infrastructure to Application Information Exposure</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure as a Service</td>
</tr>
<tr>
<td>ID</td>
<td>Identifier</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institution of Electrical and Electronic Engineers</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IGP</td>
<td>Interior Gateway Protocol</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPv4</td>
<td>Internet Protocol version 4</td>
</tr>
</tbody>
</table>
## Acronyms (Cont)

- **IPv6**: Internet Protocol version 6
- **IRTF**: Internet Research Taskforce
- **IS-IS**: Intermediate System to Intermediate System
- **ISO**: International Standards Organization
- **LAN**: Local Area Network
- **LISP**: Locator-ID Separation Protocol
- **LS**: Link State
- **MAC**: Media Access Control
- **MPLS**: Multi-protocol Label Switching
- **NAT**: Network Address Translation
- **NFV**: Network Function Virtualization
- **NTP**: Network Time Protocol
- **NVGRE**: Network Virtualization using Generic Routing Encapsulation
- **NVO3**: Network Virtualization over L3
- **NVP**: Network Virtualization Platform
Acronyms (Cont)

- OF  OpenFlow
- OnePK  Open Network Environment Platform Kit
- ONF  Open Networking Forum
- ONV  OpenDaylight Network Virtualization
- OpEx  Operational Expenses
- OS  Operating System
- OSCP  OpenDaylight SDN Controller Platform
- OSGi  Open Services Gateway Initiative
- OSPF  Open Shortest Path First
- OVS  Open Virtual Switch
- OVSDB  Open Virtual Switch Database
- PCC  Path Computation Client
- PCE  Path Computation Element
- PCEP  Path Computation Element Protocol
- POP  Post Office Protocol
- PWE3  Pseudowire Emulation Edge to Edge
Acronyms (Cont)

- QoS  Quality of Service
- REST Representational State Transfer
- RFC Request for Comments
- RLOC Routing Locator
- RLOC Routing Locator
- RS Routing System
- SAL Service Abstraction Layer
- SDN Software Defined Networking
- SMTP Simple Mail Transfer Protocol
- SNMP Simple Network Management Protocol
- SSH Secure Socket Host
- STT Stateless TCP-like Transport
- TCP Transmission Control Protocol
- TE Traffic Engineering
- TIA Telecom Industry Association
- TRILL Transparent Interconnection of Lots of Links
Acronyms (Cont)

- URI Uniform Resource Identifier
- vBridge Virtual Bridge
- VIRL Virtual Internet Routing Lab
- VLAN Virtual Local Area Network
- VM Virtual Machine
- VNS Virtual Network Segment
- VPN Virtual Private Network
- vTep Virtual Tunnel End Point
- VTN Virtual Tenant Network
- VxLAN Virtual Extensible Local Area Network
- WAN Wide Area Network
- XML Extensible Markup Language
- XMPP Extensible Messaging and Presence Protocol
SDN Related Organizations and Projects

- Open Networking Foundation (ONF): www.opennetworking.org
- Telecom Industry Association (TIA): www.tiaonline.org
- Internet Engineering Task Force (IETF): www.ietf.org
- Open Data Center Alliance, http://www.opendatacenteralliance.org
- OpenStack Quantum: https://wiki.openstack.org/wiki/Quantum
- OpenDaylight: www.opendaylight.org
**SDN Web Sites**

- SDN Central, [http://www.sdncentral.com](http://www.sdncentral.com)