Table of Contents

SDN Overview and Architecture

What are the major benefits of SDN?

→ Early days
- Simple forwarding and routing via software
- Networks small with large shared domains
- Plug and play devices
- Manual configuration of devices
- Computer occupied entire room

→ Evolution
- Developments implemented distributed environment with intelligence in each device
- Coordination between devices lead to collective decisions
- Goals:
  - Simplicity
  - Ease of use
  - Automatic recovery

→ Move control software off the device, and into a centrally located compute resource

→ Central device can see entire network, and make optimal decisions

→ SDN attempts to segregate network activities in following manner:
- Forwarding, filtering, prioritisation
- Control
- Application

→ SDN promotes research and innovation
- Open software environments promote rapid pace of advancement
  - e.g. Linux, KVM, MySQL, Postgres
- Hardware commoditisation and openness leads to innovation

→ Inadequacies in networks
- Growing need for network to respond to frequent/immediate changes
- Network changes take days, VM changes take minutes

→ Data centre needs
- Scalability – limits of MAC and number of VLANs problematic
- Network virtualisation – abstract over physical network, “anytime anywhere”
- Automation – dynamically instantiate networks, as well as tear down
- Multi-tenancy – need to have multiple paths
  - Shortest path – speed and efficiency
  - Alternate and redundant paths – resiliency, high availability, load balancing

→ Fundamental traits of SDN:
- Plane separation
  - Separation of forwarding and control planes
  - Forwarding plane:
    - Logic to deal with incoming packets
    - Actions:
      - Forward – lookup in hardware ASIC (application-specific integrated circuit)
      - Drop
      - Consume
      - Replicate – multicast
  - Control plane:
    - Contains protocols, logic, algorithms
    - Has global knowledge of network
    - Determines how the forwarding tables in the data plane are programmed/configured
    - Legacy switch – co-located with data plane
    - SDN – separated out to its own centralised controller

        - Application plane:
            - Incorporates SDN applications, which communicate the network requirements to the SDN controller
    - Simplified device and centralised control
        - One controller versus thousands of lines of complex code across multiple switches
        - Software-based controller manages the network using higher level policies
            - Primitive instructions sent to simplified devices allow them to make fast decisions on incoming packets
    - Network automation and virtualisation
        - Distributed state abstraction
            - Provides a global network view (controller)
            - Network programmer shielded from the complexity of many machines with their own state collaborating to solve network problems (data plane)
        - Forwarding abstraction
            - Network programmer can specify forwarding behaviour without having to know vendor specific hardware
        - Configuration abstraction
            - Overall goals of the network without getting lost in the details of the physical network
    - Openness
        - Standard
        - Non-proprietary
        - Well documented

What role does the ONF (Open Networking Foundation) have in relation to SDN?

→ Established by Deutsche Telekom, Facebook, Google, Microsoft, Verizon, and Yahoo
→ Responsible for overseeing OpenFlow standards
→ Board is composed of network operators (not vendors)
→ Sustaining SDN interoperability
    - Independent body to ensure implementations in OF space adheres to standards
    - Defines, clarifies, expands standards
    - Guarantees interoperability among OF via:
        - Plugfests
        - Certification programs
        - Education and consulting

Why did the fore-runners of SDN (e.g. Active Networking, ForCES, 4D) not have the impact that OpenFlow has had?

→ Early efforts, prior to OpenFlow/SDN
    - DCAN (Devolved Control of ATM Networks)
        - Separate control/management from switch
    - Open Signalling
        - Proposed open APIs – key is separate control software from switching hardware
        - GSMP (General Switch Management Protocol)
            - Centralised controller establish/release ATM connections
    - Deviation from autonomous and distributed forwarding decisions

        - Cisco's Tag Switching
        - MLPS (Multiprotocol Label Switching)
- → Network Access Control
    - Controls access to the network
    - Based on policies by network admin
    - Exchange of credentials for access/rights
    - NAC method
        - RADIUS (Remote Authentication Dial-In User Service)
            - Used to provide auto reconfig of network
            - Originally designed AAA (authentication, authorisation, accounting)
        - COPS (Common Open Policy Service)
- → Orchestration
    - Early attempts called Orchestrators
        - Generalised command applied to heterogenous devices via:
            - APIs
            - CLI
            - SNMP
- → Virtualisation Manager Network Plugins
    - Still used SNMP/CLI
        - Prone to error
        - Static configuration capability
- → ForCES
    - Separation of forwarding and control planes
    - Hardware-based forwarding at foundation of network element
    - Software-based control elements above
- → 4D centralised network control
    - Named for four planes:
        - Decision
        - Dissemination
        - Discovery
        - Data
    - Concentrates control plane operations into an independent system dedicated to this purpose
    - Challenges of centralised architectures:
        - Latency
        - Scale
        - High Availability
        - Security
- → Lead to birth of OpenFlow
    - Developed to allow researchers to experiment with new protocols in everyday networks

SDN has seen an increased use of open-source software. How have network equipment vendors responded to this?

- → Closed environments encourage vendor lock-in
    - Standards and competitive advantage
- → Power of collective
    - Software world allows small players to contribute
        - OS (e.g. Linux)
        - Databases (e.g. MySQL)
        - Servers (e.g. Apache)
        - Security (e.g. OpenSSL)

        - File sharing (e.g. BitTorrent)
- → Danger of collective
    - Solutions need scrutiny over commercial counterparts
    - There may be competing solutions to do same task
    - Release impact from timeliness
    - Support
        - Create your own
        - Trust the collective community
    - Open source licensing
- → Urgent need for SDN
    - Short term solutions:
        - Cisco's policy-based routing
        - SDN providers using rich API on top of their legacy switches with a centralised controller
- → Existing SDN device implementations
    - Available from NEMs (Network Equipment Manufacturer)
        - Cisco, HP, IBM have adopted OF support for some of their legacy switches
    - New class of switches – white-box switches
        - Built on low cost merchant silicon switching chips and commodity CPU/memory
        - No control plane
        - Uses OVS (Open vSwitch) code with OF logic

Some vendors provide SDN functionality via APIs. In what ways is this approach good or bad?

- → SDN applications
    - Built on top of the SDN controller
    - Interfaces with the SDN controller to
        - Set proactive flows (aka static flows)
        - Handle packets that have been passed up to it (e.g. no matching flow action)
            - May define a reactive flow passed down to device when packet passed up
        - Reactive flows from other stimuli
            - IDS (intrusion detection system)
            - Traffic loads
        - OpenFlow is the standard for open SDN
        - SDN devices composed of:
            - API for communication with controller
                - SDN providers using rich API on top of their legacy switches with a centralised controller
            - Abstraction layer
                - 1 or more flow tables
            - Packet processing function/logic
                - Actions to take on incoming packets based on highest priority match
        - Flow tables
            - Fundamental data structure in SDN
        - Packets
            - Traditional switches – inspect packet fields and take action
                - Forward to specific port
                - Drop
                - Flood the packet

        - SDN is not that much different except that the logic is rendered more generic and more programmable
        - Matching logic can match on:
            - IP address
            - Subnet
            - MAC address
            - UDP/TCP port
            - Other fields
            - Can use wildcards
    - Interfaces
        - Low-level API – access to network devices
        - High-level API – abstraction of the network
        - Events from the network, as well as external events, go to the application which in turn executes methods exposed to it by the controller
        - Southbound interface is standardised
        - Northbound interface is not

→ Alternate SDN methods

- Many vendors are offering SDN by enhancing their existing APIs
- RESTful API – dominant method of making API calls REST using HTTP
- Benefits of SDN via APIs:
    - Works with legacy switches
    - Improves agility and automation
    - Easier to write orchestration tools at software later
    - Facilitates centralised control
    - Increased openness of SDN via API approach
- Limitations of SDN via APIs:
    - No controller
        - Network programmer must interact directly with switch
        - Network programmer must sync with control
            - Complexity of synchronising central controller with distributed control place
    - Even with controller, there may be no abstraction layer
    - SDN-like vendor APIs only work with that vendor's device
    - More expensive
    - Propreitary
- SDN via device APIs:
    - Enhanced APIs
        - Beyond CLI/SNMP
    - Popular method
        - NETCONF
- SDN via controller APIs:
    - Apps developed using APIs exposed by the controller
    - Example controller is OpenDaylight (see southbound APIs)
- SDN via policy APIs:
    - Allows for declarative perspective (what to do vs how to do)
- SDN via hypervisor-based overlay networks
    - Virtualised network overlay physical network
    - SDN application given access to these networks/ports
    - HV inserts traffic into the virtual network
    - Endpoints unaware, works because of encapsulation via tunnel
    - VTEP (virtual tunnel endpoint)
        - Takes forwarded packet, and decapsulates for forwarding
        - VTEPs source and destination host devices
        - Virtual network capability added to HV by extending with virtual switch
    - Tunnelling mechanism known as MAC-in-IP

- Entire frame (from MAC address inward) encapsulated in unicast IP frame

What risks are involved in implementing SDN on a company's network?

→ Reactive vs Proactive
- ○ Service disruption
  - Reactive programming may be more vulnerable to service disruption if connectivity to controller is lost
  - Loss of controller has less impact in proactive model, if failure mode specifies that operation should continue
    - Failure mode config set to fail closed → controller loss flushes flow tables
- ○ Latency
  - Proactive model – no additional latency for flow setup, bc they are prepopulated
- ○ Drawback
  - Most flows are wildcarded
    - Aggregate flows, and less fine granularity of control
- ○ Internal vs external applications
  - Int run inside the OSGi container of controller
  - Ext run outside that contains (can run anywhere, typically use RESTful APIs provided by controller)

→ New security challenges
- ○ Possibility of a compromised SDN controller attack at the control plane layer (SDN controller is essentially brain of SDN architecture)
- ○ If attacker gains access, they can tell switches to drop all traffic, or send useless traffic to victim
- ○ Control plane susceptible to DDoS (distributed denial-of-service) attack
  - Possible defence is to implement multiple physical SDN controllers, instead of just one
  - One of these controllers can act as the master of the switches, and when under heavy load, can direct some of it towards other controllers → load balancing
- ○ Data plane also susceptible to DDoS attack
  - Flood switch with large payload, causing packets to get dropped

Briefly describe the role of each component of a Software-Defined Network.

→ SDN applications
- ○ Run above the SDN controller
- ○ Interface to controller on Northbound API, to:
  - Configure flows to route packets between two endpoints
  - Balance traffic loads across multiple paths/destined endpoints
  - React to changes in network topo (e.g. link failures)
  - Redirect traffic for inspection, authentication, segregation
- ○ Strength of SDN is centralised control, and richer applications (e.g. firewall, load balancers, etc.)

→ SDN application responsibilities
- ○ Once controller has finished initialising devices, the applications spends most of its time responding to events
- ○ External stimuli

        - Network monitoring systems (e.g. NetFlow, IDS, BGP peers)
    - End-user device discovery
    - Network device discovery
    - Handle incoming packet

How does SDN facilitate the operation of a multi-tenant data centre network?

→ Use of VMs rather than dedicating entire server rack to single application/customer
- With 12-tuple, network administrators can configure the SDN controller to route traffic based on any combination of headers with each configuration called a flow
- Large amount of possible permutations afforded to the network administrator when configuring flows provides a lot of granularity
- A single customer may have multiple services hosted with a multi-tenant data centre
- Network administrators can isolate the traffic from other tenants within the same data centre by configuring flows based on the ingress port, source port, destination port and any other combination of headers that would refer exclusively to the service hosted by that customer
- If another tenant within the data centre runs a similar service or application, the network administrator can instruct the SDN controller to route traffic based on the same headers but with differing values of said headers
- Each tenant's traffic is successfully isolated from one another without disrupting the network's performance

In a Software-Defined network, what are 'flows' and 'flow-rules'?

→ Controller populates flow table
- Controller will update the switch with new flow entries as new packet patterns are received, so that the switch can deal with them locally

→ SDN openness
- Standard
- Non-proprietary
- Well documented
- These all keep north and south bound interfaces to the SDN controller open

→ The flow represents packets transferred from one network endpoint to another

→ SDN controller defines flows, and maintains this info on SDN devices

→ Flow is unidirectional

→ Flow table resides on network device
- Contains flow entries and associated actions for matching packet arriving on device
- Built and maintained by the controller
- Actions on packet can include forward, drop, send to controller (depends on version of OF, and switch config)
- Flow is a simple programming expression, as a result of a potentially complex computation done in the controller (can't be done speedily)

→ SDN devices scale the number of flows:
- At edge, flows applied to individual users and traffic types
- At core, policies deep in the network apply to aggregate flows (e.g. VLAN, MPLS, LSP)

Outline the fundamental characteristics of a Software-Defined Network.

→ …

How is TCAM (Ternary Content-Addressable Memory) relevant to the operation of a Software-Defined Network?

→ SDN hardware switches
- Promise of operating much faster than software counterparts
- L2 and L3 forwarding tables in
  - CAM (content addressable memory)
    - L2 – MAC level forwarding
      - Implement CAM with precise indices using 48-bit address
  - TCAM (ternary content-addressable memory)
    - L3 – IP level routing
    - Can handle more complex matching
    - Can use wildcards
    - More than just IP address
    - Facilitates the implementation of PBR (policy-based routing)
- Challenges for SDN developer
  - How to best translate flows into CAMs, TCAMs, hash tables
  - Which flow entries go in software and which in hardware?
  - How to deal with hardware limitations?
  - How to track statistics on individual flows?
    - TCAMs can't track when multiple flows matched
  - These challenges impact the quality, functionality, and efficiency of SDN switches

Describe the operation of SDN in a hypervisor-based overlay network environment.

→ SDN via hypervisor-based overlay networks
- Virtualised network overlay physical network
  - Overlay networks don't solve all problems
    - Issues in physical infrastructure have to be manually handled
    - No traffic prioritisation
    - Does not address desire to open network devices for innovation/simplification
- SDN application given access to these networks/ports
- HV inserts traffic into the virtual network
- Endpoints unaware, works because of encapsulation via tunnel
- VTEP (virtual tunnel endpoint)
  - Takes forwarded packet, and decapsulates for forwarding
  - VTEPs source and destination host devices?
  - Virtual network capability added to HV by extending with virtual switch
- Tunnelling mechanism known as MAC-in-IP
  - Entire frame (from MAC address inward) encapsulated in unicast IP frame

→ Overlay networks don't solve all problems
- Issues in physical infrastructure have to be manually handled
- No traffic prioritisation
- Does not address desire to open network devices for innovation/simplification

Outline the different SDN devices that are available for building a Software-Defined Network.

→ SDN controller

- Maintains view of entire network
- Implements policy decisions
  - Routing
  - Forwarding
  - Load balancing
- Controls SDN switches
- Own set of common applications
  - Learning switch
  - Router
  - Basic firewall
- These are SDN applications, but bundled with the controller

SDN Protocols, Verification and Troubleshooting

Why did OpenFlow version 1.0 include support for multiple queues per port?

→ OF 1.0 released in 2009, considered to be the initial release
- Legacy – multiple queues per port
  - Served by scheduling algorithms
  - Different QoS levels for packets
  - OF embraces this concept
- Packet matching
  - Referred to as 12-tuple
  - Match field may be wildcarded using a bit mask
- Flow table – core of OF switch
  - V1.0 does not specify which fields are required
  - ONF clarified matching by V1.0 conformance
    - Full conformance – all 12 fields used
    - L2 conformance – only L2 matching fields used
    - L3 conformance – only L3 matching fields used
- Action fields – what switch should do with the packet
  - Most common action specified a physical port to forward
    - Packet forwarding?
    - Enqueue – selects particular queue associated with port
    - Modify field – modify field on header (e.g. VLAN headers)
    - When there are multiple actions, processed in order
  - 5 virtual ports defined in 1.0:
    - LOCAL – forward to OF control software
    - ALL – flood packet to ports, except input port
    - CONTROLLER – send packet to OF controller
    - IN_PORT – send packet back out the input port
    - TABLE – arrives a PACKET_OUT with message action
  - 2 additional ports specified in 1.0:
    - NORMAL – sends packet to legacy forwarding logic in switch
    - ALL/FLOOD – sends packet to all ports, except ingress port
- Richer functionality in today's OF version has outpaced reality of today's hardware

What security is provided for OpenFlow connections between controllers and switches, and why is it not always used?

→ Controller-Switch secure channel
- Path used for comms between OF controller and OF device
- Secured through TLS-based asymmetrical encryption
  - Unencrypted TCP connections allowed
  - Don't use in tightly controlled data centre
  - V1.0 has one secure channel; newer releases have many
  - Message between controller/switch starts with OF header

'The connection from OpenFlow controller to switch can be in-band or out-of-band'. What does this mean?

→ OpenFlow switch
- Flow table
  - Forward

        - Drop
        - Pass to controller
    - ASICs – hardware has to be part of SDN discussions
    - OpenFlow-only
        - Only forwards packets according to OF logic
    - OpenFlow-hybrid
        - Can use OF logic, as well as packets in legacy mode as IP router/ethernet switch
        - Probably the norm as the migration to fully SDN implementation
- → OpenFlow controller
    - Legacy switches
    - Long standing control and data plane
    - OF control plane different
        - Programs different data plane elements with common language
        - On separate hardware device than data plane
        - Programs multiple forwarding elements from single instance of control plane
- → Controller-Switch secure channel
    - Connections in-band or out-of-band
        - Out-of-band secure channel relevant only to OF hybrid switch

What is the function of barrier messages in OpenFlow?

- → BARRIER_REQUEST – tells switch to complete all commands received previously
- → BARRIER_REPLY – switch tells controller it has finished

The OpenFlow specification categorises message types as 'Symmetric', 'Async' and 'Immutable' (among others). What does each of these three categories mean?

- → Symmetric/Immutable:
    - Symmetric messages sent by controller or switch without being solicited
    - Immutable means the message types will not change going forward
    - HELLO – ensures secure channel established
    - ECHO_REQUEST – ensure other side is alive
    - ECHO_REPLY – ensure other side is alive
    - VENDOR – vendor specific implementations
- → Async:
    - Async messages sent from switch to controller without being solicited by the controller
    - PACKET_IN
    - FLOW_REMOVED – switch tells controller a flow is removed
    - PORT_STATUS – communicates changes in port status
    - ERROR – notifies controller of problems
- → Other notable:
    - SET_CONFIG – set config params on switch
    - FEATURES – how controller requests features supported by switch
    - GET_CONFIG – controller retrieves switch config
    - STATS – controller receives statistics from switch
    - QUEUE_GET_CONFIG – learns how queues are configured
    - OFPR_NO_MATCH – switch sends packet to controller bc no match found

- OFPR_ACTION – controller has configured flow on switch to send packer to controller when a certain flow is matched
- BUFFER_ID – used to pull full packet from switch

What support does OpenFlow provide for MPLS and VLAN tags?

→ Complete support for multiple levels of VLAN tags
- Support popping and pushing

→ Support for MPLS tagging
→ PUSH – new header of specified type inserted in front of outermost header, can use SET action to set value in header
→ POP
→ V1.0 allows modification of outermost header

How many controllers does OpenFlow allow to be in use in a network at one time?

→ Northbound API
- 20 controllers available with unique APIs

→ Multiple controllers, config switch to have connections to each
→ 3 roles:
- Equal
- Slave
- Master

How has the OpenFlow protocol evolved over its multiple versions?

→ OpenFlow
- Subset of technologies under SDN umbrella
- Defines the communications protocol between control and data planes
  - Between OF controller and OF switch
  - Messaging in place to allow controller fine-grained control over user traffic
- Defines some behaviour of the data plane
- Is the only non-proprietary, general purpose protocol for programming the forwarding plane of SDN switches
- Pop – remove item from LIFO list (last in, first out)
- Push – add item to LIFO list
- Stack – pop/push used with this, CS term for LIFO ordered list
- Created in 2008 by people at Stanford University
- ONF formed in 2011 to accelerate SDN
  - Took over responsibility for OF oversight
- OF designers recognised that switches built around ASICs
- Flow is a set of packets transferred from one network endpoint (or set of) to another endpoint (of set of)
  - Endpoint may be IP address, TCP/UDP port pairs, VLAN endpoints, L3 tunnel endpoints, input ports, etc.
- Protocol states what actions should take place by the data plane
- Packets follow matching flow:
  - Forward to one or more ports
  - Drop
  - Pass packet back to controller
- Versioning
  - Some early features no longer in current releases

        - Backwards compatibility important
        - Richer functionality in today's OF version has outpaced reality of today's hardware
    - V1.3 additions
        - Auxiliary connections
            - Multiple parallel comms existed on single switch to multiple controllers
            - Now, allow multiple connections per comms channel
            - Use PACKET_IN messages directly from ASIC to controller
        - PBB (provider backbone bridging tagging)
            - Allows LANs to be L2 bridged across provider domains
    - V1.4 additions
        - More significant features
        - New error codes
        - New port descriptor which contains fields to support optical ports
        - Bundles
            - More transactional control vs BARRIER_REQUEST/REPLY
        - No component message should be acted upon until entire bundle has been received/committed
        - Enhanced support for multiple controllers
            - Flow monitoring
        - Optical port support
        - Flow table synchronisation
    - V1.5 additions
        - Enhanced L4-L7 support
        - Pipeline processing enhancements
        - Egress tables
            - Flow table matching flow coming in ingress table
        - Enhanced support for multiple controllers
        - Improved OF interoperability

Against what parts of a packet does OpenFlow allow matches?

- Packet matching
    - Referred to as 12-tuple
    - Match field may be wildcarded using a bit mask
    - Can match on:
        - Switch input port
        - VLAN ID
        - VLAN priority
        - Ethernet source addr
        - Ethernet dest addr
        - Ethernet frame type
        - IP source addr
        - IP dest addr
        - IP protocol
        - IP TOS bits (Type of Service)
        - TCP/UDP source port
        - TCP/UDP dest port

What are the purposes of these OpenFlow messages: FLOW_MOD, PORT_STATUS, PACKET_IN, PACKET_OUT?

→ FLOW_MOD – controller modifies flow entries in switch
→ PORT_STATUS – communicates changes in port status
→ PACKET_IN – packets encapsulated, forwarded to controller
→ PACKET_OUT – controller sends packets to switch for forwarding through data plane

The OpenFlow specification defines 'meters'. How are these useful in a network?

→ Per-flow meters
  - Structure designed to support definition of complex meters in future OF versions
  - Each meter band has configured bandwidth rate/type
  - Band selected based on highest bandwidth that is lower than current measured rate
  - No "required" types; "optional" types:
    - DROP
    - DSCP (differentiated services code point)

How does OpenFlow fit in the context of legacy protocols such as IEEE 802.3 and 802.1Q?

→ 802.3
  - ???

→ 802.1Q
  - Bridging VLANs though SP networks
  - Known as provider bridging, Q-in-Q
  - Based on 802.1Q tags
  - Implemented in V1.2 by encoding flow entry to match VLAN tag from customer L2 network

In the context of OpenFlow, what are 'standard' and 'reserved' ports?

→ Virtual ports
  - Introduced categories of standard and reserved ports
  - Standard ports
    - Physical ports
    - Switch-defined virtual ports
      - Used for more complex processing outside of header modification, examples:
        - Forwarding a packet to a tunnel
        - LAG (link aggregation)
  - Reserved ports
    - ???

Why is verification important on a Software-Defined Network, and what tools are available?

→ Debugging – ndb, OFRewind, NetSight

- → Verification – NICE, OFLOPS, VeriFlow
  - ○ Veriflow
    - System that resides between the controller and the switches
    - Verifies the correctness of each flow entry update before it is applied
    - Advantage of approach is that it does not require knowledge of all the network programs themselves, since it verifies correctness based on observations of flow rules as they are sent from the controller to the switches
- → Correctness – VeriCON

What approach should be taken to troubleshooting a Software-Defined Network, and what tools are available?

- → Tools: Mininet, Wireshark?
- → NICE (no bugs in controller execution)
  - ○ Proposal to have a tool for modelling behavior which will detect forwarding loops, etc.
- → Network debugger
  - ○ With higher level languages to program CPUs, we need a basic debugger functionality at the packet level

An OpenFlow controller just received an OFPT_FLOW_REMOVED message from a switch. In what situations does this happen?

- → OFPT – OpenFlow message type
- → FLOW_REMOVED – switch tells controller a flow is removed

SDN vSwitches and Controllers

In a flow-rule entry in a switch's flow-table, what is the purpose of the IDLE_TIMEOUT value?

- → Flow tables tend to continually evolve, based on packets being processed by switch, and by flows ageing out
  - ○ Flows need to be reprogrammed too frequently → set idle timeouts for flows
    - ▪ Idle timer used to clear out flow entries after they have terminated/gone inactive
      - • Important to consider nature of flow, and what counts as inactivity
    - ▪ Idle timeouts too short → results in too many packets sent to controller
    - ▪ Idle timeouts too long → can result in flow table overflow

In SDN, what is the difference between a southbound and a northbound interface? Give examples of each.

- → Southbound interface is standardised
- → Northbound interface is not
- → Several initiatives around standardising NBI, such as one group as the ONF

From where does a controller receive information that triggers changes to flow-rules?

- → Controller populates flow table
  - ○ Controller will update the switch with new flow entries as new packet patterns are received, so that the switch can deal with them locally

If an SDN switch loses the connection to its controller, what are the possible options for how the switch behaves?

- → Controller connection failure
  - ○ Emergency flow cache included in V1.0 to handle these failures
    - ▪ Fail secure mode
      - • Continues to forward packets
    - ▪ Fail standalone mode
      - • OF pipeline processing stops
      - • Continues to operate in native switch or router mode
    - ▪ Switch ban be built with support for both, or one of the above
      - • Support for both → switch will be configured to enter one of the modes
    - ▪ When comms with controller restored, switch resumes normal ops
  - ○ Loss of the controller will have less impact in the proactive model if the failure mode specifies that operation should continue.
    - ▪ Failure mode configuration is set to fail closed → controller loss flushes flow tables

What are the advantages and disadvantages of a RESTful interface to an SDN controller?

→ REST (Representational State Transfer)
- Popular for SDN networks – asynchronous notifications not performed by RESTful API
  - Asymmetric – application (requester), controller (responder)
- Problem:
  - Not straightforward for controller to asynch notify application of an event, such as the arrival of packet from switch
  - Latency – requires reverse-direction control relationship, with controller being the requester, and application being responder
- Java APIs – can register listener, then receive callbacks from controller when packets arrive
  - Callbacks from with passed args, packet, associated metadata
- Reactive applications tend to be written in native lang of controller
- RESTful APIs can be high-level, providing a level of abstraction above network switches, so that SDN application interacts with network components, such as virtual networks, rather than with switches themselves
- RESTful API interface, referred to as a flow pusher, allows application to set flows on switches

What is 'Intent-Based Networking'?

→ Use tools that SDN provides, along with AI and ML, and they will setup network as desired

In the context of SDN, what are the differences between a software switch and a hardware switch?

→ SDN software switches
- Easiest way to deploy an SDN
- Due to standards, more consistent implantation versus hardware versions
- Flow entries can be a lot larger than hardware versions
- Often found in hypervisors of a virtualisation system
- Can implement flow tables in standard computer memory
- Feature limitations and table size issues are not a major concern
- Performance will still be a consideration, since implementations in hardware will generally be faster

→ Hardware switches
- SDN application major issue regarding flow capacity is how many flow entries the hardware ASIC is capable of holding
  - Some less 1,000 flow entries
  - Some have dependencies on types of flows
- Type of applications
  - Traffic prioritisation application or a TCP port-specific may use relatively few flow table entries
  - Access control application may use many flow entries.
- Topology of the application
- Switch or AP at edge may use much fewer flow entries
- Hardware implementations often have feature limitations
  - OpenFlow specification mapped to the capabilities of the hardware
  - Some features may not be supported

- e.g. some ASICs can perform NORMAL forwarding when packet has had header modified
  - Hardware limitations may be by trial and error
    - Application developer must be aware of these types of limitations before designing a solution

'A network administrator should use either all software switches or all hardware switches in their organisation's network, but not both'. Do you agree? Explain your answer.

→ Questions to Consider
- What is the basic nature of the application?
- What is the type and nature of network with which the application will be dealing?
- What is the intended deployment of the controller with respect to the switches it controls?
- Are the SDN needs purely related to the data centre and virtualization?
- Will the application run in a greenfield environment, or will it need to deal with legacy switches that do not support SDN?
- What type of API's are available on legacy switches and routers? What is their level of capability?
- What is the level of programming ability present in your development team(s)?
- How many switches will be managed by your application?

→ With SDN, the controller can figure out how to integrate a new device into the network. While this is a huge advantage for organizations that attempt to be agile, it can cause problems with visibility. When admins add or remove multiple devices, networking or otherwise, it can be difficult to maintain real-time awareness over the networks, which can lead to significant security issues. For example, it may be easier for hackers to add devices to an SDN-enabled network if there's a lack of proper network monitoring.

What functions must an SDN controller provide?

→ SDN controller code modules
- Core features inside controller:
  - End user device discovery
  - Network device discovery
  - Network device topology management
  - Flow management
- Core functions – implemented internal to the controller
  - Device discovery and tracking
  - Topology discovery and tracking
  - Flow management (flow cache maintained to mirror flow tables in SDN switches)
  - Device management
  - Statistics tracking
- Interfaces
  - Low-level API – access to network devices
  - High-level API – abstraction of the network
  - Events from the network, as well as external events, go to the application which in turn executes methods exposed to it by the controller
  - Southbound interface is standardised
  - Northbound interface is not

→ Potential issues with the SDN controller
- Few large-scale deployments – lack of real life shake out

- Need wider array of applications with mix of equipment for confidence in architecture
- ONF needs to address issues as they come up
- Multiple SDN applications on a controller:
  - How do they collaborate/coordinate?
  - No standard Northbound API
  - Flow prioritisation
    - Which SDN applications should have the event first?
    - Should the application pass it to the next?
    - Flows in SDN device processed in priority order
    - Setting priority flows in one application is easy
    - What about across multiple SDN applications?

Why does an SDN controller need to keep a flow-rule cache?

→ …

A network manager you know is planning to use SDN applications running on a controller to replace some expensive hardware appliances. Give your advice on how she should proceed.

→ SDN is good when network changes a lot
→ ASICs – hardware specifically for networking
→ Switch Considerations
- One of the most important considerations in building an SDN application is the nature of the switches that are to be controlled by the application.
- Do all the switches support OpenFlow? What version of OpenFlow do they support? How much of that version do they in fact support?
- Is there a mix of OpenFlow and non-OpenFlow switches? If so, will some non-OpenFlow mechanism be used to provide the best simulation of an SDN environment? For example, do the non-OpenFlow switches support some type of SDN API that allows a controller to configure flows in the switch?
- For OpenFlow-supporting switches, are they hardware or software switches, or both? If both, what differences in behaviour and support will need to be considered?
- For hardware OpenFlow switches, what are their hardware flow-table sizes? If the switches are capable of handling flow table overflow in software, at what point are they forced to begin processing flows in software, and what is the performance impact of doing so? Is it even possible to process flows in software? What feature limitations are introduced because of the hardware?
- What is the mechanism by which the switch-controller communications channel is secured? Is this the responsibility of the network programmer?

→ Legacy protocol device considerations
- Devices using NETCONF and BGP-LS/PCEP requires knowledge of the devices

→ More power to the programmer increases the likelihood of mis-programming
→ Architecture of OpenFlow does not tightly synchronise the programming of multiple switches when the controller must simultaneously program flow entries in several switches

Compare two SDN controllers.

→ ONOS

- Provide ability to create packet listeners for receiving packets forwarded from switches
- ???

→ Floodlight
- Comes with sample applications, such as learning switch, load balancer
- Core modules come from seminal Beacon controller
- Provide ability to create packet listeners for receiving packets forwarded from switches

→ OpenDaylight
- OF functionality in controller is derived from Beacon code
- Most of controller capabilities provided by Cisco
- APIs – Java, RESTful
  - Topology provides APIs to retrieve the interswitch links in the network
  - Host Tracker provides APIs to retrieve the hosts (end nodes) in the network
  - Flow Programmer provides APIs for reading and writing flows on specific switches in the network
  - Static Routing provides APIs for reading and writing static routes (e.g., next-hop rules) on switches in the network
  - Statistics provides APIs for retrieving statistics for flows, ports, tables, and switches
  - Subnets provides APIs for retrieving information about subnets
  - Switch Manager provides APIs for retrieving information about switches

There are several SDN controllers currently available. What lead to there being more than one? How do you expect the situation to develop in the next few years, and why?

→ As applications become more complex, hybrid model of reactive-proactive applications is likely to become more common

→ Need to see a new generation of switching ASICs purposefully built with SDN and OpenFlow in mind

→ More Energy Efficient Switching Hardware
- SDN environment directly designed with more energy efficient switches
- Well-known methods of reducing power consumption used on servers are already being applied to switch design
  - One of most power-hungry components of a modern switch capable of flow-based policy enforcement is the TCAM
- Use prediction circuitry to identify a high percentage of packets to avoid power hungry TCAM lookups

Helpful Links

1. **Benefits and the Security Risk of Software-defined Networking**
   https://www.isaca.org/resources/isaca-journal/issues/2016/volume-4/benefits-and-the-security-risk-of-software-defined-networking

2. **SDN Management: Risks And Challenges**
   https://www.networkcomputing.com/networking/sdn-management-risks-and-challenges

3. **SDN Cheat Sheet**
   https://ipcisco.com/wp-content/uploads/Cheat-Sheets/SDN.pdf

4. **SDN Architecture Components**
   https://ipcisco.com/lesson/sdn-architecture-components/