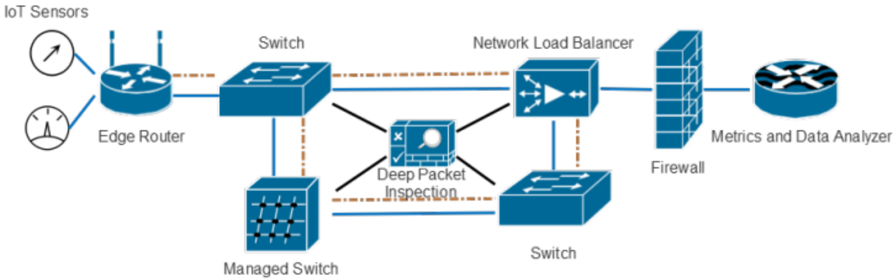
Software Defined Network

Internet of Things

Rahul Shandilya

Traditional internetworking

In this configuration, the data plane and control plane are unified. When the system needs to add or remove another node or set up a new data path, many of the dedicated systems need to be updated with new VLAN settings, QoS parameters, access control lists, static routes, and firewall pass-throughs.



Limitations of the conventional network architectures

- **Complex Network Devices**: Conventional networks are getting increasingly complex with more and more protocols being implemented to improve link speeds and reliability. Interoperability is limited due to the lack of standard and open interfaces. The conventional networks were well suited for static traffic patterns and had a large number of protocols designed for specific applications. For IoT applications which are deployed in cloud computing environments, the traffic patterns are more dynamic.

Limitations of the conventional network architectures

- **Complex Network Devices**: Conventional networks are getting increasingly complex with more and more protocols being implemented to improve link speeds and reliability. Interoperability is limited due to the lack of standard and open interfaces. The conventional networks were well suited for static traffic patterns and had a large number of protocols designed for specific applications. For IoT applications which are deployed in cloud computing environments, the traffic patterns are more dynamic.
- **Management Overhead**: Network managers find it increasingly difficult to manage multiple network devices and interfaces from multiple vendors. Upgradation of network requires configuration changes in multiple devices (switches, routers, firewalls, etc.)

Limitations of the conventional network architectures

- **Complex Network Devices**: Conventional networks are getting increasingly complex with more and more protocols being implemented to improve link speeds and reliability. Interoperability is limited due to the lack of standard and open interfaces. The conventional networks were well suited for static traffic patterns and had a large number of protocols designed for specific applications. For IoT applications which are deployed in cloud computing environments, the traffic patterns are more dynamic.
- **Management Overhead**: Network managers find it increasingly difficult to manage multiple network devices and interfaces from multiple vendors. Upgradation of network requires configuration changes in multiple devices (switches, routers, firewalls, etc.)
- **Limited Scalability**: The virtualization technologies used in cloud computing environments has increased the number of virtual hosts requiring network access. The analytics components of IoT applications run distributed algorithms on a large number of virtual machines that require huge amounts of data exchange between virtual machines. Such computing environments require highly scalable and easy to manage network architectures with minimal manual configurations, which is becoming increasingly difficult with conventional networks.

Software Defined Network

Defintion

Software-Defined Networking (SDN) is an emerging paradigm that promises to make complex IP network more manageable and flexible by breaking vertical integration(the control and data planes are bundled together), separating the network's control logic from the underlying routers and switches, promoting (logical) centralization of network control, and introducing the ability to program the network.

Software Defined Network

Defintion

Software-Defined Networking (SDN) is an emerging paradigm that promises to make complex IP network more manageable and flexible by breaking vertical integration(the control and data planes are bundled together), separating the network's control logic from the underlying routers and switches, promoting (logical) centralization of network control, and introducing the ability to program the network.

- The separation of concerns introduced between the definition of network policies, their implementation in switching hardware, and the forwarding of traffic, is key to the desired flexibility

Software Defined Network

Defintion

Software-Defined Networking (SDN) is an emerging paradigm that promises to make complex IP network more manageable and flexible by breaking vertical integration(the control and data planes are bundled together), separating the network's control logic from the underlying routers and switches, promoting (logical) centralization of network control, and introducing the ability to program the network.

- The separation of concerns introduced between the definition of network policies, their implementation in switching hardware, and the forwarding of traffic, is key to the desired flexibility
- By breaking the network control problem into tractable pieces, SDN makes it easier to create and introduce new abstractions in networking, simplifying network management and facilitating network evolution.

Characteristics

- The control plane is decoupled from the data plane. Data plane hardware becomes simple packet-forwarding devices.

Characteristics

- The control plane is decoupled from the data plane. Data plane hardware becomes simple packet-forwarding devices.
- All forwarding decisions are flow-based rather than destination-based. A flow is a set of packets that match a criterion or filter. All packets in a flow are treated with the same forwarding and service policies. Flow programming allows for easy scaling and flexibility with virtual switches, firewalls, and middleware.

Characteristics

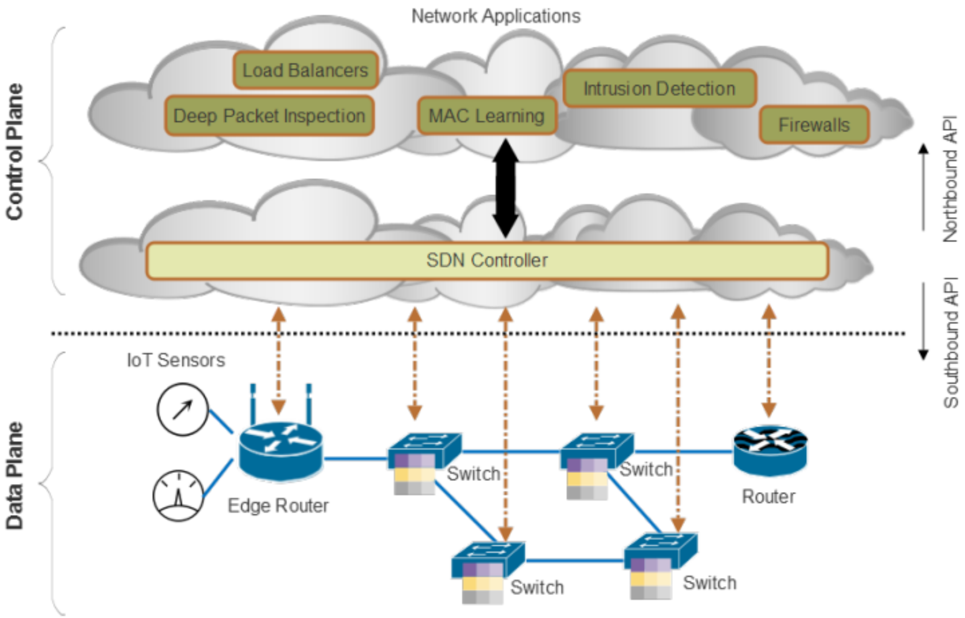
- The control plane is decoupled from the data plane. Data plane hardware becomes simple packet-forwarding devices.
- All forwarding decisions are flow-based rather than destination-based. A flow is a set of packets that match a criterion or filter. All packets in a flow are treated with the same forwarding and service policies. Flow programming allows for easy scaling and flexibility with virtual switches, firewalls, and middleware.
- The control logic is also known as the SDN Controller. This software version of legacy hardware is capable of running on commodity hardware and cloud-based instances. Its purpose is to command and govern the simplified switching nodes.

Characteristics

- The control plane is decoupled from the data plane. Data plane hardware becomes simple packet-forwarding devices.
- All forwarding decisions are flow-based rather than destination-based. A flow is a set of packets that match a criterion or filter. All packets in a flow are treated with the same forwarding and service policies. Flow programming allows for easy scaling and flexibility with virtual switches, firewalls, and middleware.
- The control logic is also known as the SDN Controller. This software version of legacy hardware is capable of running on commodity hardware and cloud-based instances. Its purpose is to command and govern the simplified switching nodes.
- The control logic is also known as the SDN Controller. This software version of legacy hardware is capable of running on commodity hardware and cloud-based instances. Its purpose is to command and govern the simplified switching nodes.

Characteristics

- The control plane is decoupled from the data plane. Data plane hardware becomes simple packet-forwarding devices.
- All forwarding decisions are flow-based rather than destination-based. A flow is a set of packets that match a criterion or filter. All packets in a flow are treated with the same forwarding and service policies. Flow programming allows for easy scaling and flexibility with virtual switches, firewalls, and middleware.
- The control logic is also known as the SDN Controller. This software version of legacy hardware is capable of running on commodity hardware and cloud-based instances. Its purpose is to command and govern the simplified switching nodes.
- The control logic is also known as the SDN Controller. This software version of legacy hardware is capable of running on commodity hardware and cloud-based instances. Its purpose is to command and govern the simplified switching nodes.
- Network application software can reside over the SDN controller through a northbound interface. This software can interact and manipulate the data plane with services such as deep packet inspection, firewalls, and load balancers.



Network Applications
Load Balancers
Deep Packet Inspection
MAC Learning
Intrusion Detection
Firewalls
Control Plane
Northbound API
SDN Controller
Southbound API
Data Plane
IoT Sensors
Edge Router
Switch
Switch
Router
Switch
Switch

Key Elements

- *Centralized Network Controller*: With decoupled control and data planes and centralized network controller, the network administrators can rapidly configure the network, SDN applications can be deployed through programmable open APIs. This speeds up innovation as the network administrators no longer need to wait for the device vendors to embed new features in their proprietary hardware.

Key Elements

- *Centralized Network Controller*: With decoupled control and data planes and centralized network controller, the network administrators can rapidly configure the network, SDN applications can be deployed through programmable open APIs. This speeds up innovation as the network administrators no longer need to wait for the device vendors to embed new features in their proprietary hardware.
- Programmable Open APIs: SDN architecture supports programmable open APIs for interface between the SDN application and control layers (Northbound interface). With these open APIs various network services can be implemented, such as routing, quality of service (QoS), access control, etc.

Key Elements

- *Centralized Network Controller*: With decoupled control and data planes and centralized network controller, the network administrators can rapidly configure the network, SDN applications can be deployed through programmable open APIs. This speeds up innovation as the network administrators no longer need to wait for the device vendors to embed new features in their proprietary hardware.
- Programmable Open APIs: SDN architecture supports programmable open APIs for interface between the SDN application and control layers (Northbound interface). With these open APIs various network services can be implemented, such as routing, quality of service (QoS), access control, etc.
- Standard Communication Interface (OpenFlow): SDN architecture uses a standard communication interface between the control and infrastructure layers (Southbound interface). OpenFlow, which is defined by the Open Networking Foundation (ONF) is the broadly accepted SDN protocol for the Southbound interface. With OpenFlow, the forwarding plane of the network devices can be directly accessed and manipulated.

SDN Benefits

- **Service chaining**: This allows a customer or provider to sell services a la carte. Cloud network services such as firewalls, deep packet inspection, VPNs, authentication services, and policy brokers can be linked and used on a subscription basis. Some customers may want a full set of features, others may not choose any or may change their configuration routinely. Service chaining allows for significant flexibility in deployments.

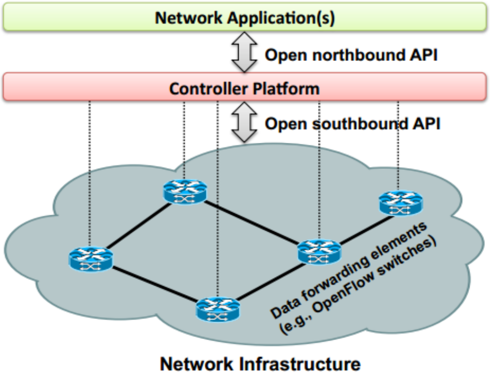
SDN Benefits

- **Service chaining**: This allows a customer or provider to sell services a la carte. Cloud network services such as firewalls, deep packet inspection, VPNs, authentication services, and policy brokers can be linked and used on a subscription basis. Some customers may want a full set of features, others may not choose any or may change their configuration routinely. Service chaining allows for significant flexibility in deployments.
- **Dynamic load management**: An SDN enjoys the flexibility of cloud architecture, and by design it can scale resources dynamically depending on load. This type of flexibility is crucial for the IoT as architects need to plan for capacity and scale as the number of things grows exponentially. Only virtual networking in the cloud provides the ability to scale capacity when needed.

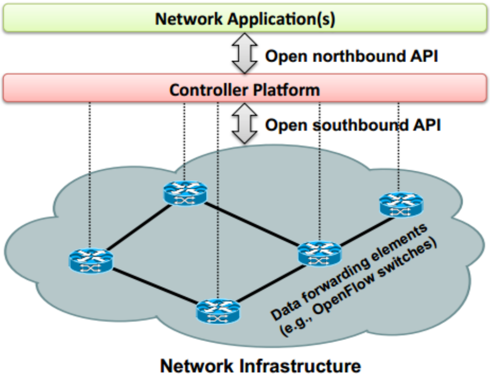
SDN Benefits

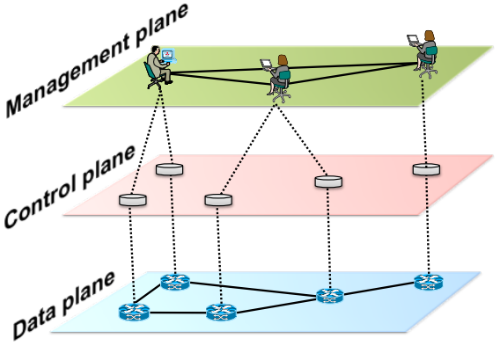
- **Service chaining**: This allows a customer or provider to sell services a la carte. Cloud network services such as firewalls, deep packet inspection, VPNs, authentication services, and policy brokers can be linked and used on a subscription basis. Some customers may want a full set of features, others may not choose any or may change their configuration routinely. Service chaining allows for significant flexibility in deployments.
- **Dynamic load management**: An SDN enjoys the flexibility of cloud architecture, and by design it can scale resources dynamically depending on load. This type of flexibility is crucial for the IoT as architects need to plan for capacity and scale as the number of things grows exponentially. Only virtual networking in the cloud provides the ability to scale capacity when needed.
- **Bandwidth calendaring**: This allows an operator to partition data bandwidth and usage to specified times and days. This is pertinent to IoT as many edge sensors only report data periodically or at a certain time of day. Sophisticated bandwidth sharing algorithms can be constructed to time slice capacity.

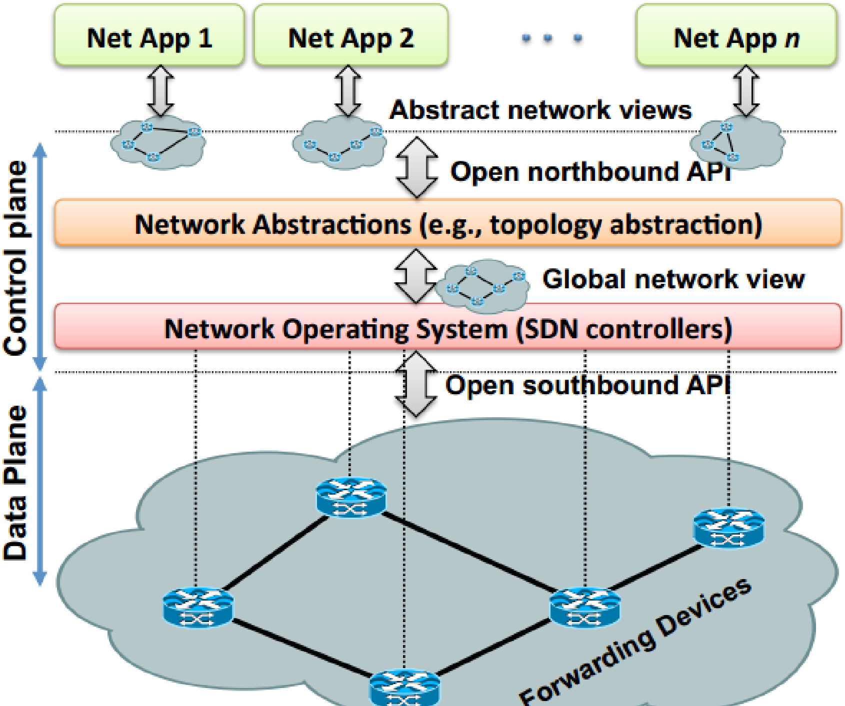
SDN Architecture



SDN Architecture






Net App 1
Net App 2
Net App n
Abstract network views
Open northbound API
Network Abstractions (e.g., topology abstraction)
Global network view
Network Operating System (SDN controllers)
Open southbound API
Control plane
Data Plane
Forwarding Devices

Elements of SDN

Forwarding Devices (FD):

Hardware- or software-based data plane devices that perform a set of elementary operations. The forwarding devices have well-defined instruction sets (e.g., flow rules) used to take actions on the incoming packets (e.g., forward to specific ports, drop, forward to the controller, rewrite some header). These instructions are defined by southbound interfaces (e.g., OpenFlow, ForCES) and are installed in the forwarding devices by the SDN controllers implementing the southbound protocols.

Elements of SDN

Forwarding Devices (FD):

Hardware- or software-based data plane devices that perform a set of elementary operations. The forwarding devices have well-defined instruction sets (e.g., flow rules) used to take actions on the incoming packets (e.g., forward to specific ports, drop, forward to the controller, rewrite some header). These instructions are defined by southbound interfaces (e.g., OpenFlow, ForCES) and are installed in the forwarding devices by the SDN controllers implementing the southbound protocols.

Data Plane (DP)

Forwarding devices are interconnected through wireless radio channels or wired cables. The network infrastructure comprises the interconnected forwarding devices, which represent the data plane.

Elements of SDN

Forwarding Devices (FD):

Hardware- or software-based data plane devices that perform a set of elementary operations. The forwarding devices have well-defined instruction sets (e.g., flow rules) used to take actions on the incoming packets (e.g., forward to specific ports, drop, forward to the controller, rewrite some header). These instructions are defined by southbound interfaces (e.g., OpenFlow, ForCES) and are installed in the forwarding devices by the SDN controllers implementing the southbound protocols.

Data Plane (DP)

Forwarding devices are interconnected through wireless radio channels or wired cables. The network infrastructure comprises the interconnected forwarding devices, which represent the data plane.

Southbound Interface (SI):

The instruction set of the forwarding devices is defined by the southbound API, which is part of the southbound interface. Furthermore, the SI also defines the communication protocol between forwarding devices and control plane elements. This protocol formalizes the way the control and data plane elements interact.

Control Plane (CP):

Forwarding devices are programmed by control plane elements through well-defined SI embodiments. The control plane can therefore be seen as the "network brain". All control logic rests in the applications and controllers, which form the control plane.

Control Plane (CP):

Forwarding devices are programmed by control plane elements through well-defined SI embodiments. The control plane can therefore be seen as the "network brain". All control logic rests in the applications and controllers, which form the control plane.

Northbound Interface (NI):

The network operating system can offer an API to application developers. This API represents a northbound interface, i.e., a common interface for developing applications. Typically, a northbound interface abstracts the low level instruction sets used by southbound interfaces to program forwarding devices

Control Plane (CP):

Forwarding devices are programmed by control plane elements through well-defined SI embodiments. The control plane can therefore be seen as the "network brain". All control logic rests in the applications and controllers, which form the control plane.

Northbound Interface (NI):

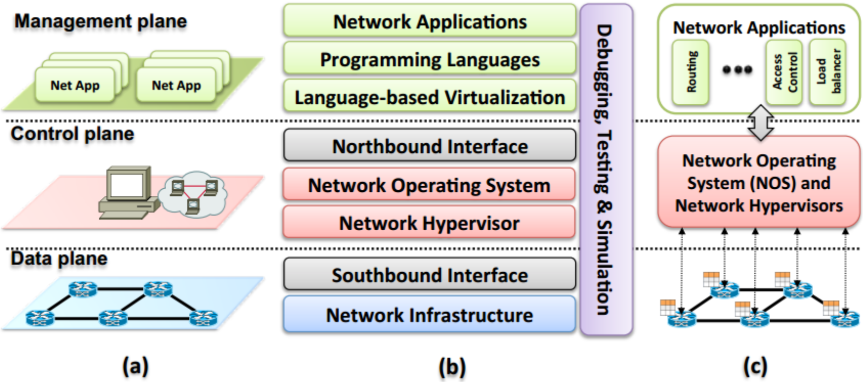
The network operating system can offer an API to application developers. This API represents a northbound interface, i.e., a common interface for developing applications. Typically, a northbound interface abstracts the low level instruction sets used by southbound interfaces to program forwarding devices

Management Plane (MP):

The management plane is the set of applications that leverage the functions offered by the NI to implement network control and operation logic. This includes applications such as routing, firewalls, load balancers, monitoring, and so forth. Essentially, a management application defines the policies, which are ultimately translated to southbound-specific instructions that program the behavior of the forwarding devices

SDN Layered Architecture

An SDN architecture can be depicted as a composition of different layers. While some of them are always present in an SDN deployment, such as the southbound API, network operating systems, northbound API and network applications, others may be present only in particular deployments, such as hypervisor- or language-based virtualization.



Layer I: Infrastructure

- An SDN infrastructure, similarly to a traditional network, is composed of a set of networking equipment (switches, routers and middlebox appliances).

Layer I: Infrastructure

- An SDN infrastructure, similarly to a traditional network, is composed of a set of networking equipment (switches, routers and middlebox appliances).
- The difference is that those traditional physical devices are now simple forwarding elements without embedded control or software to take autonomous decisions.

Layer I: Infrastructure

- An SDN infrastructure, similarly to a traditional network, is composed of a set of networking equipment (switches, routers and middlebox appliances).
- The difference is that those traditional physical devices are now simple forwarding elements without embedded control or software to take autonomous decisions.
- Configuration and communication compatibility and interoperability among different data and control plane devices are ensured by standard open interfece like OpenFlow.

Layer I: Infrastructure

- An SDN infrastructure, similarly to a traditional network, is composed of a set of networking equipment (switches, routers and middlebox appliances).
- The difference is that those traditional physical devices are now simple forwarding elements without embedded control or software to take autonomous decisions.
- Configuration and communication compatibility and interoperability among different data and control plane devices are ensured by standard open interfece like OpenFlow.

Layer I: Infrastructure

- An SDN infrastructure, similarly to a traditional network, is composed of a set of networking equipment (switches, routers and middlebox appliances).
- The difference is that those traditional physical devices are now simple forwarding elements without embedded control or software to take autonomous decisions.
- Configuration and communication compatibility and interoperability among different data and control plane devices are ensured by standard open interfece like OpenFlow.

Layer II: Southbound Interfaces

- Southbound interfaces (or southbound APIs) are the connecting bridges between control and forwarding elements, thus being the crucial instrument for clearly separating control and data plane functionality.

Layer I: Infrastructure

- An SDN infrastructure, similarly to a traditional network, is composed of a set of networking equipment (switches, routers and middlebox appliances).
- The difference is that those traditional physical devices are now simple forwarding elements without embedded control or software to take autonomous decisions.
- Configuration and communication compatibility and interoperability among different data and control plane devices are ensured by standard open interfece like OpenFlow.

Layer II: Southbound Interfaces

- Southbound interfaces (or southbound APIs) are the connecting bridges between control and forwarding elements, thus being the crucial instrument for clearly separating control and data plane functionality.
- As a central component of its design the southbound APIs represent one of the major barriers for the introduction and acceptance of any new networking technology therefore standard SDN southbound API like OpenFlow Emerges .

The *OpenFlow* protocol provides three information sources for network operating systems.

- First, event-based messages are sent by forwarding devices to the controller when a link or port change is triggered.

The *OpenFlow* protocol provides three information sources for network operating systems.

- First, event-based messages are sent by forwarding devices to the controller when a link or port change is triggered.
- Second, flow statistics are generated by the forwarding devices and collected by the controller.

The *OpenFlow* protocol provides three information sources for network operating systems.

- First, event-based messages are sent by forwarding devices to the controller when a link or port change is triggered.
- Second, flow statistics are generated by the forwarding devices and collected by the controller.
- Third, packet-in messages are sent by forwarding devices to the controller when they do not known what to do with a new incoming flow or because there is an explicit "send to controller" action in the matched entry of the flow table.

The *OpenFlow* protocol provides three information sources for network operating systems.

- First, event-based messages are sent by forwarding devices to the controller when a link or port change is triggered.
- Second, flow statistics are generated by the forwarding devices and collected by the controller.
- Third, packet-in messages are sent by forwarding devices to the controller when they do not known what to do with a new incoming flow or because there is an explicit “send to controller” action in the matched entry of the flow table.

The *OpenFlow* protocol provides three information sources for network operating systems.

- First, event-based messages are sent by forwarding devices to the controller when a link or port change is triggered.
- Second, flow statistics are generated by the forwarding devices and collected by the controller.
- Third, packet-in messages are sent by forwarding devices to the controller when they do not known what to do with a new incoming flow or because there is an explicit “send to controller” action in the matched entry of the flow table.

Layer III: Network Hypervisors

- Hypervisors enable distinct virtual machines to share the same hardware resources.

The *OpenFlow* protocol provides three information sources for network operating systems.

- First, event-based messages are sent by forwarding devices to the controller when a link or port change is triggered.
- Second, flow statistics are generated by the forwarding devices and collected by the controller.
- Third, packet-in messages are sent by forwarding devices to the controller when they do not known what to do with a new incoming flow or because there is an explicit "send to controller" action in the matched entry of the flow table.

Layer III: Network Hypervisors

- Hypervisors enable distinct virtual machines to share the same hardware resources.
- Virtual machines can be easily migrated from one physical server to another and can be created and/or destroyed on-demand, enabling the provisioning of elastic services with flexible and easy management.

The *OpenFlow* protocol provides three information sources for network operating systems.

- First, event-based messages are sent by forwarding devices to the controller when a link or port change is triggered.
- Second, flow statistics are generated by the forwarding devices and collected by the controller.
- Third, packet-in messages are sent by forwarding devices to the controller when they do not known what to do with a new incoming flow or because there is an explicit "send to controller" action in the matched entry of the flow table.

Layer III: Network Hypervisors

- Hypervisors enable distinct virtual machines to share the same hardware resources.
- Virtual machines can be easily migrated from one physical server to another and can be created and/or destroyed on-demand, enabling the provisioning of elastic services with flexible and easy management.
- Unfortunately, virtualization has been only partially realized in practice.

Layer IV: Network Operating Systems / Controllers

- Networks have so far been managed and configured using lower level, device-specific instruction sets and mostly closed proprietary network operating systems (e.g., Cisco IOS and Juniper JunOS)

Layer IV: Network Operating Systems / Controllers

- Networks have so far been managed and configured using lower level, device-specific instruction sets and mostly closed proprietary network operating systems (e.g., Cisco IOS and Juniper JunOS)
- SDN is promised to facilitate network management and ease the burden of solving networking problems by means of the logically-centralized control offered by a network operating system (NOS)

Layer IV: Network Operating Systems / Controllers

- Networks have so far been managed and configured using lower level, device-specific instruction sets and mostly closed proprietary network operating systems (e.g., Cisco IOS and Juniper JunOS)
- SDN is promised to facilitate network management and ease the burden of solving networking problems by means of the logically-centralized control offered by a network operating system (NOS)
- The crucial value of a NOS is to provide abstractions, essential services, and common application programming interfaces (APIs) to developers.

Layer IV: Network Operating Systems / Controllers

- Networks have so far been managed and configured using lower level, device-specific instruction sets and mostly closed proprietary network operating systems (e.g., Cisco IOS and Juniper JunOS)
- SDN is promised to facilitate network management and ease the burden of solving networking problems by means of the logically-centralized control offered by a network operating system (NOS)
- The crucial value of a NOS is to provide abstractions, essential services, and common application programming interfaces (APIs) to developers.
- Generic functionality as network state and network topology information, device discovery, and distribution of network configuration can be provided as services of the NOS.

Layer IV: Network Operating Systems / Controllers

- Networks have so far been managed and configured using lower level, device-specific instruction sets and mostly closed proprietary network operating systems (e.g., Cisco IOS and Juniper JunOS)
- SDN is promised to facilitate network management and ease the burden of solving networking problems by means of the logically-centralized control offered by a network operating system (NOS)
- The crucial value of a NOS is to provide abstractions, essential services, and common application programming interfaces (APIs) to developers.
- Generic functionality as network state and network topology information, device discovery, and distribution of network configuration can be provided as services of the NOS.
- With NOSs, to define network policies a developer no longer needs to care about the low-level details of data distribution among routing elements, for instance

Layer V: Northbound Interfaces

- The northbound interface is mostly a software ecosystem, not a hardware one as is the case of the southbound APIs.

Layer V: Northbound Interfaces

- The northbound interface is mostly a software ecosystem, not a hardware one as is the case of the southbound APIs.
- Open and standard northbound interfaces are crucial to promote application portability and interoperability among the different the control platforms.

Layer V: Northbound Interfaces

- The northbound interface is mostly a software ecosystem, not a hardware one as is the case of the southbound APIs.
- Open and standard northbound interfaces are crucial to promote application portability and interoperability among the different the control platforms.
- A northbound API can be compared to the POSIX standard in operating systems, representing an abstraction that guarantees programming language and controller independence.

Layer V: Northbound Interfaces

- The northbound interface is mostly a software ecosystem, not a hardware one as is the case of the southbound APIs.
- Open and standard northbound interfaces are crucial to promote application portability and interoperability among the different the control platforms.
- A northbound API can be compared to the POSIX standard in operating systems, representing an abstraction that guarantees programming language and controller independence.

Layer V: Northbound Interfaces

- The northbound interface is mostly a software ecosystem, not a hardware one as is the case of the southbound APIs.
- Open and standard northbound interfaces are crucial to promote application portability and interoperability among the different the control platforms.
- A northbound API can be compared to the POSIX standard in operating systems, representing an abstraction that guarantees programming language and controller independence.

Layer VI: Language-based Virtualization

- Two essential characteristics of virtualization solutions are the capability of expressing modularity and of allowing different levels of abstractions while still guaranteeing desired properties such as protection

Layer V: Northbound Interfaces

- The northbound interface is mostly a software ecosystem, not a hardware one as is the case of the southbound APIs.
- Open and standard northbound interfaces are crucial to promote application portability and interoperability among the different the control platforms.
- A northbound API can be compared to the POSIX standard in operating systems, representing an abstraction that guarantees programming language and controller independence.

Layer VI: Language-based Virtualization

- Two essential characteristics of virtualization solutions are the capability of expressing modularity and of allowing different levels of abstractions while still guaranteeing desired properties such as protection
- Pyretic is example of such language that offers this type of high-level abstraction of network topology. It incorporates this concept of abstraction by introducing network objects.

Layer V: Northbound Interfaces

- The northbound interface is mostly a software ecosystem, not a hardware one as is the case of the southbound APIs.
- Open and standard northbound interfaces are crucial to promote application portability and interoperability among the different the control platforms.
- A northbound API can be compared to the POSIX standard in operating systems, representing an abstraction that guarantees programming language and controller independence.

Layer VI: Language-based Virtualization

- Two essential characteristics of virtualization solutions are the capability of expressing modularity and of allowing different levels of abstractions while still guaranteeing desired properties such as protection
- Pyretic is example of such language that offers this type of high-level abstraction of network topology. It incorporates this concept of abstraction by introducing network objects.
- These objects consist of an abstract network topology and the sets of policies applied to it. Network objects simultaneously hide information and offer the required services.

Layer VII: Programming languages

In SDNs, high-level programming languages can be designed and used to:

- create higher level abstractions for simplifying the task of programming forwarding devices;

Layer VII: Programming languages

In SDNs, high-level programming languages can be designed and used to:

- create higher level abstractions for simplifying the task of programming forwarding devices;
- enable more productive and problem-focused environments for network software programmers, speeding up development and innovation;

Layer VII: Programming languages

In SDNs, high-level programming languages can be designed and used to:

- create higher level abstractions for simplifying the task of programming forwarding devices;
- enable more productive and problem-focused environments for network software programmers, speeding up development and innovation;
- promote software modularization and code reusability in the network control plane;

Layer VII: Programming languages

In SDNs, high-level programming languages can be designed and used to:

- create higher level abstractions for simplifying the task of programming forwarding devices;
- enable more productive and problem-focused environments for network software programmers, speeding up development and innovation;
- promote software modularization and code reusability in the network control plane;
- foster the development of network virtualization.

Layer VIII: Network Applications

- Network applications can be seen as the "network brains". They implement the control-logic that will be translated into commands to be installed in the data plane, dictating the behavior of the forwarding devices.

Layer VIII: Network Applications

- Network applications can be seen as the "network brains". They implement the control-logic that will be translated into commands to be installed in the data plane, dictating the behavior of the forwarding devices.
- Software-defined networks can be deployed on any traditional network environment, from home and enterprise networks to data centers and Internet exchange points. Such variety of environments has led to a wide array of network applications.

Layer VIII: Network Applications

- Network applications can be seen as the "network brains". They implement the control-logic that will be translated into commands to be installed in the data plane, dictating the behavior of the forwarding devices.
- Software-defined networks can be deployed on any traditional network environment, from home and enterprise networks to data centers and Internet exchange points. Such variety of environments has led to a wide array of network applications.
- Existing network applications perform traditional functionality such as routing, load balancing, and security policy enforcement, but also explore novel approaches, such as reducing power consumption.

Layer VIII: Network Applications

- Network applications can be seen as the "network brains". They implement the control-logic that will be translated into commands to be installed in the data plane, dictating the behavior of the forwarding devices.
- Software-defined networks can be deployed on any traditional network environment, from home and enterprise networks to data centers and Internet exchange points. Such variety of environments has led to a wide array of network applications.
- Existing network applications perform traditional functionality such as routing, load balancing, and security policy enforcement, but also explore novel approaches, such as reducing power consumption.
- Other examples include fail-over and reliability functionalities to the data plane, end-to-end QoS enforcement, network virtualization, mobility management in wireless networks, among many others.

Layer VIII: Network Applications

- Network applications can be seen as the “network brains”. They implement the control-logic that will be translated into commands to be installed in the data plane, dictating the behavior of the forwarding devices.
- Software-defined networks can be deployed on any traditional network environment, from home and enterprise networks to data centers and Internet exchange points. Such variety of environments has led to a wide array of network applications.
- Existing network applications perform traditional functionality such as routing, load balancing, and security policy enforcement, but also explore novel approaches, such as reducing power consumption.
- Other examples include fail-over and reliability functionalities to the data plane, end-to-end QoS enforcement, network virtualization, mobility management in wireless networks, among many others.
- The variety of network applications, combined with real use case deployments, is expected to be one of the major forces on fostering a broad adoption of SDN