SDN OPENFLOW CONTROLLERS: POX, NOX ARCHITECTURE.

NOX

- NOX is the original OpenFlow controller.
- It serves as a network control platform, that provides a high level programmatic interface for management and the development of network control applications.
- Its system-wide abstractions turn networking into a software problem.

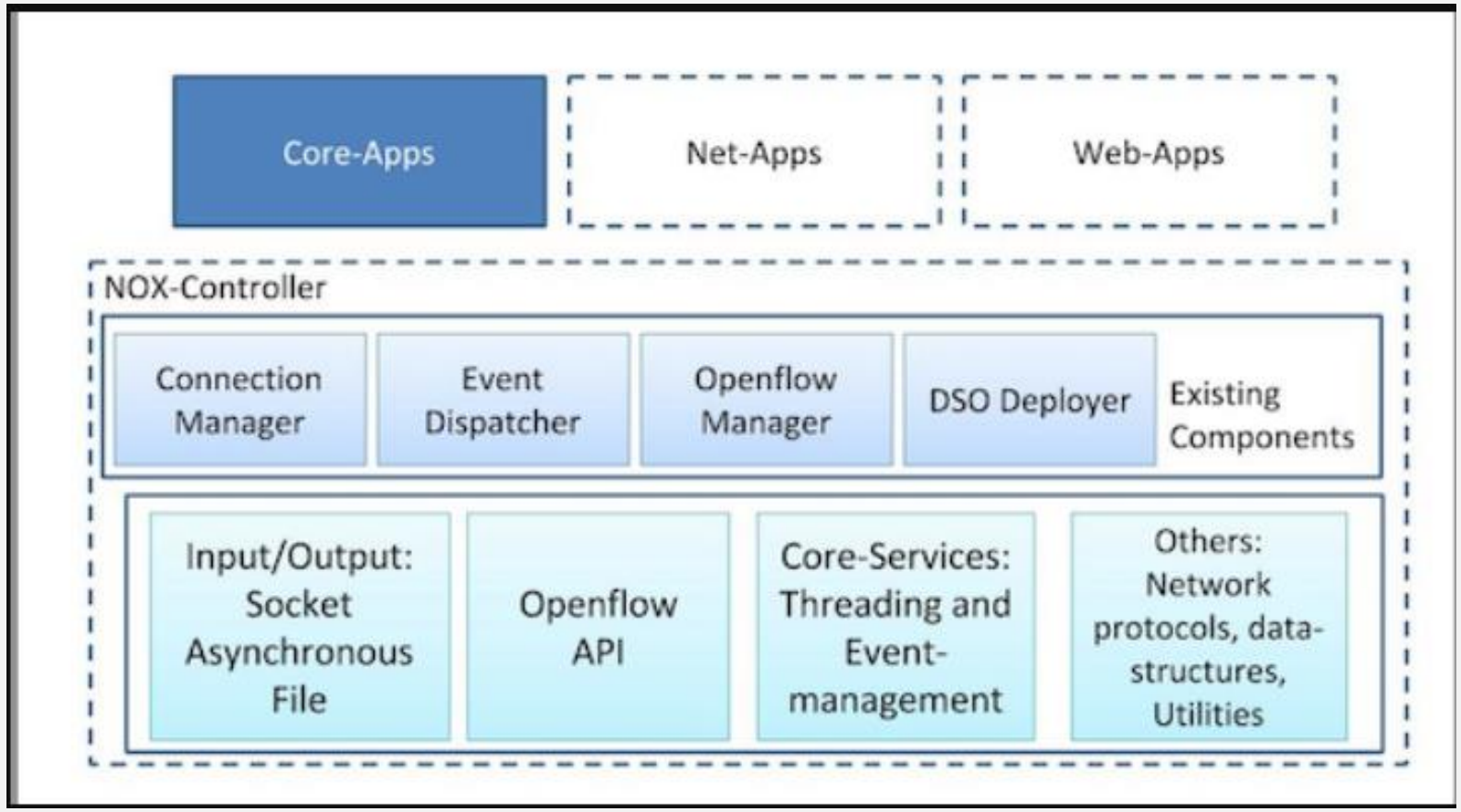
- NOX versions:
  1. NOX classic: This is the version that has been available under the GPL since 2009.
  2. NOX: The “new NOX.” Only contains support for C++ and has lesser applications than the classic; however, this version is faster and has better codebase.
  3. POX: Typically termed as NOX’s sibling. Provides Python support.

The differences between nox classic and nox.

	NOX	NOX classic
Core apps	OpenFlow, Switch	Messenger, SNMP, switch
Network Apps	—	Discovery, Topology, Authenticator, Routing, Monitoring
Web Apps	—	Webservice,Webserver,WebServiceClient
Language Support	C++ Only	C++ and Python
GUI	NO	YES

- NOX aims to provide a platform which allows developers and researchers the ability to innovate within enterprise networks in the form of developing novel applications.
- Applications on NOX typically determine how each flow is routed or not routed in the network.

NOX Architecture



- The NOX core provides helper methods, such as network packet process, threading and event engine, in addition to OpenFlow APIs for interacting with OpenFlow switches, and I/O operations support.

- At the top, we have applications: Core, Net and Web.

- However, with the current NOX version, there are only two core applications: OpenFlow and switch, and both network and web applications are missing.

- The middle layer shows the in-built components of NOX. The connection manager, event dispatcher and OpenFlow manager are self-explanatory, whereas the dynamic shared object (DSO) deployer basically scans the directory structure for any components being implemented as DSOs.

- All the applications can be viewed as components.

- All applications inherit from the component class. Hence, NOX applications are generally composed of cooperating components that provide the required functionality. In short, a component encapsulates specific functionality that is made available to NOX.

- An event represents a low-level or high-level event in the network.
- Typically the event only provides the information, and processing of that information is deferred to handlers.
- Many events roughly correlate to something which happens on the network that may be of interest to a NOX component.
- These components, typically, consists a set of event handlers. In this sense, events drive all execution in NOX.
- NOX events can be broadly classified as core events and application events. The core events map directly to OpenFlow messages received by controlled switches, such as:

OpenFlow-Events	Description
Datapath_join_event	When a new switch is detected.
Datapath_leave_event	When a switch leaves the network.
Packet_in_event	Called for each new packet received.
Flow_mod_event	When a flow has been added or modified.
Flow_removed_event	When a flow in the network expires/removed.
Port_status_event	Indicates a change in port status.
Port_stats_in	When a port statistics message is received.

- In addition to core events, components themselves may define and throw higher level events which may be handled by any other events.

- Though NOX does not contain any such application events, considering it has a minimal set applications, the NOX classic has various events such as **host_event** and **flow_in_event** by authenticator application, and **link_event** by the discovery application.

Running NOX

- NOX must be invoked by the command line within the build/src directory. Generally, the command that starts the controller looks like this:

  ./nox_core [OPTIONS] [APP[=ARG[,ARG]...]] [APP[=ARG[,ARG]...]]...

- For instance, the following will initiate NOX, listening for incoming connections from OpenFlow switches on port 6633 (the Openflow protocol port):

  ./nox_core -v -i ptcp:6633

- At this point, the core of NOX is running; however, while switches can now connect to the controller, no behavior will be imposed on them by NOX.
- NOX is intended to provide the control logic for an entire network, such as handling traffic engineering, routing, authentication, access control, virtual network creation, monitoring and diagnostics.
- However, NOX itself does none of these things. Rather, it provides a programmatic interface to network components which perform the useful functionality.

POX

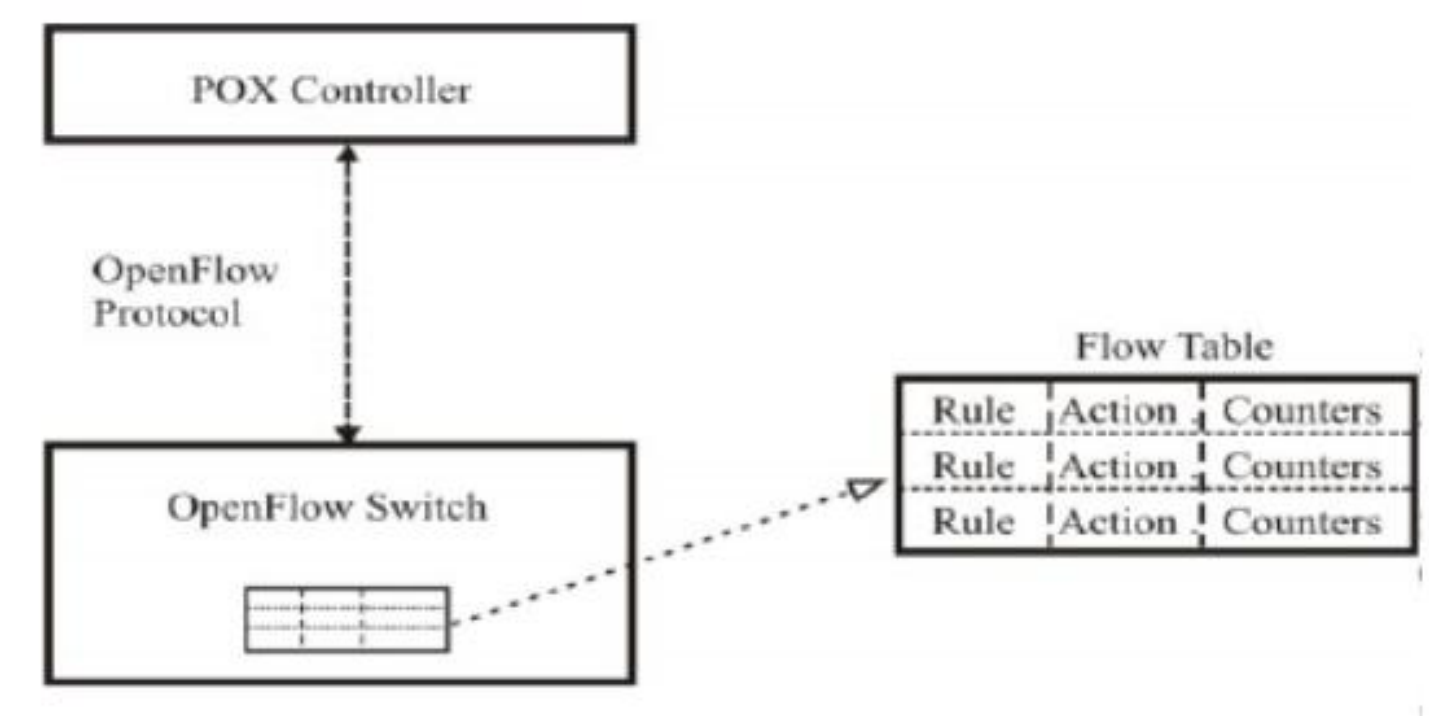
- POX is an open source development platform for Python-based software-defined networking (SDN) control applications, such as OpenFlow SDN controllers.
- POX, which enables rapid development and prototyping, is becoming more commonly used than NOX.
- POX, is to allow users to write their own applications that use the controller as an intermediary — or abstraction layer — between network applications and the network equipment.

Running POX

- Start POX by running the *pox.py* program, and specifying the POX components to use.
- For example, to run POX so it makes the switches it controls emulate the behavior of Ethernet learning switches, run the command:

```
mininet@mininet-vm:~$ sudo ~/pox/pox.py forwarding.l2_learning
```

POX Architecture



- POX controller provides an efficient way to implement the OpenFlow protocol which is the communication protocol between the controllers and the switches.

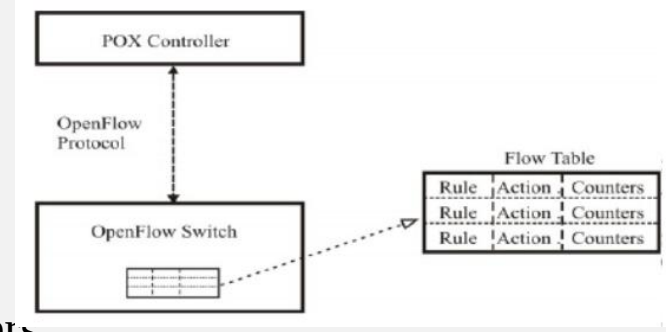
- Using POX controller you can run different applications like hub, switch, load balancer, and firewall.

- Tcpdump packet capture tool can be used to capture and see the packets flowing between POX controller and OpenFlow devices.

- Communication between the controller and the switches is carried by communication protocol. OpenFlow is the most popular standard protocol used in SDN.

- OpenFlow switches behave as dumb forwarding devices. They are unable to perform any actions without programmed by the controller.

- When a switch is powered on, it will immediately connect to an OpenFlow controller.
- Initially, the flow table of the switches is empty.
- When a packet arrives at a switch, it does not know, how this packet is to be handled.
- Then it send packet-in message to the controller.
- To handle the packet, controller inserts a flow entries in flow table of switch.
- Flow entry in flow table contains three parts, rule(match field), action, counters.
- For each packet, that has to pass through a switch, a flow entry will have to be installed so that the switch can forward this traffic without further intervention of the controller .
- Flow modification messages are sent to the switches to install the flow entries in flow table.
- Once these are installed, traffic belonging to this flow will be handled by the switches themselves.



Advantages of POX over NOX

- Has a pythonic OpenFlow interface
- Has reusable sample components for path selection, topology discovery, and so on.
- Runs anywhere and can be bundled for easy development.
- Specifically targets Linux, Max OS, and Windows
- Supports the same GUI and virtualization tool as NOX.
- Performs well compared to NOX applications in Python
- To use POX controller, type the following command in terminal window.