Professional Electives IV- Software Defined Networking (5CS 413)

- **Course Objectives :**
  1. To understand SDN/NFV motivation and benefits
  2. To describe how SDN/Openflow work
  3. To describe OpenFlow operation and the OpenStack
- **Course Learning Outcomes:**
  1. Explain and discuss the basic concepts and architecture of SDN in particular benefits brought about by the separation of data and control planes.
  2. analyze and apply implementation of SDN through Open Flow Switches
  3. critically evaluate the pros and cons of applying SDN, API approaches, Hypervisor overlays, and Data Center SDN

References

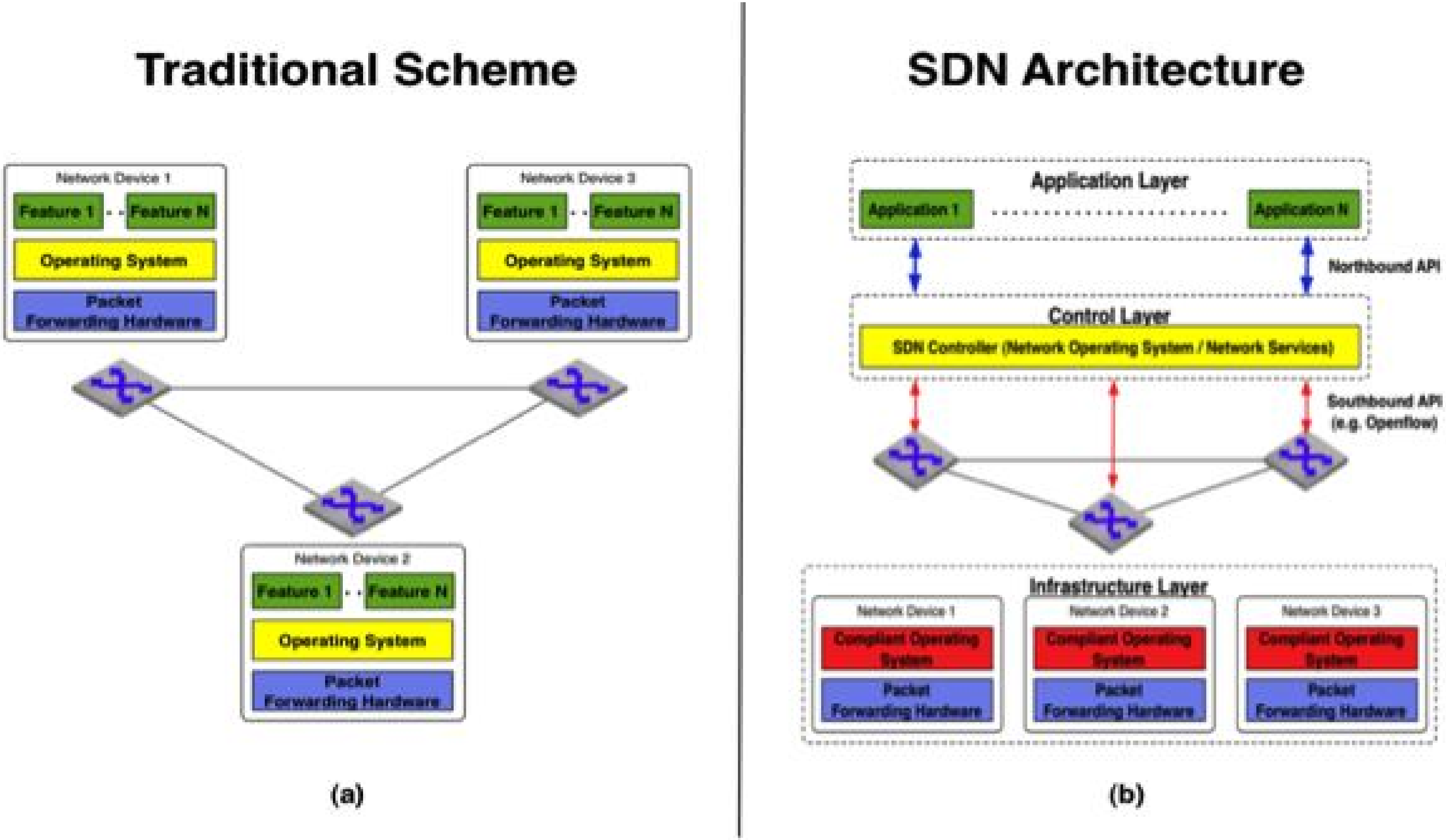
1. SDN:Software Defined Networks, An Authoritative Review of Network Programmability Technologies, By Thomas Nadeau.
2. Software Defined Network: A comprehensive Approach by Paul Goransson.
3. SDN and openflow for beginners by Vivek Tiwari.

Module 1 : History & Evolution of Software Defined Network

- **Basic packet switching terminologies:**
  1. WAN, LAN, MAN, WLAN (50 M)
  2. OSI LAYERS
  3. MAC
  4. PORT (INTERFACE)
  5. FRAME
  6. PACKET
  7. MAC ADDRESS
  8. IP ADDRESS (IP4 & IP6)
  9. CIRCUIT SWITCH ,PACKET SWITCH
  10. CONNECTION ORIENTED & CONNECTIONLESS MODEL
  11. ROUTER
  12. FLOODING, BROADCASTING

Traditional network:

- Network functionality is mainly implemented in a dedicated appliance.
- ‘Dedicated Appliance’ refers to one or multiple switches, routers and/or application delivery controllers.
- Most functionality within this appliance is implemented in dedicated hardware. An Application Specific Integrated Circuit (or: ASIC) is often used for this purpose.
- It consists of control plane, management plane and data plane. This are referred as static kind of networks.
- They work using protocols.


Traditional Scheme
Network Device 1
Feature 1
Feature N
Operating System
Packet Forwarding Hardware
Network Device 3
Feature 1
Feature N
Operating System
Packet Forwarding Hardware
Network Device 2
Feature 1
Feature N
Operating System
Packet Forwarding Hardware
SDN Architecture
Application Layer
Application 1
Application N
Northbound API
Control Layer
SDN Controller (Network Operating System / Network Services)
Southbound API (e.g. Openflow)
Infrastructure Layer
Network Device 1
Compliant Operating System
Packet Forwarding Hardware
Network Device 2
Compliant Operating System
Packet Forwarding Hardware
Network Device 3
Compliant Operating System
Packet Forwarding Hardware

(a)

(b)

Limitations of Existing network:

- **Traditional Configuration Is Time-consuming And Error-prone:**
  - ➢ Many steps are needed when an IT administrator needs to add or remove a single device in a traditional network.
  - ➢ First, he will have to manually configure multiple devices (switches, routers, firewalls) on a device-by-device basis.
  - ➢ The next step is using device-level management tools to update numerous configuration settings.
  - ➢ This configuration approach makes it that much more complex for an administrator to deploy a consistent set of policies

- **Multi-vendor Environments Require A High Level Of Expertise**

➢ The average organization owns a variety of equipment of different vendors.

➢To successfully complete a configuration, an administrator will therefore need extensive knowledge of all present device types.

- **Traditional architectures complicate network segmentation**

➢ In addition to tablets, PCs and smartphones, other devices such as alarm systems and security cameras will soon be linked to the internet.

➢The predicted explosion of smart devices is accompanied by a new challenge for organizations: how to incorporate all these devices of different vendors within their network in a safe and structured manner.

- Many traditional networks place all types of devices in the same ‘zone’.
- But in case of a compromised device, this design risks giving external parties access to the entire network.
- This can be hackers exploiting the internet connection of smart devices or vendors who can remotely log onto ‘their’ devices.

- to overcome these and other traditional networking limitations, the time has come to introduce a new perspective on network management.

History and Evolution of SDN

1. Evolution of Supporting Technologies
2. Control-Data Plane Separation
3. Developing control channels for specific data planes
4. Convergence of control channels and data planes

1. Evolution of Supporting Technologies

I. **Central Network Control:** Dates back at least to AT&T's network control point (1980s):

- In early days control and data plane were together.
- Data and control sent over the same channel
- Resulting network was brittle, insecure, etc.
- **Network Control Point:**
  - Introduced in 1981 and is intended to support wide range of SPC network applications.
  - Implemented in telephone network by separating control signals from voice and data.
  - By introducing signaling at NCP
  - Benefits:
    1. Services on demand
    2. Rapid introduction of new services

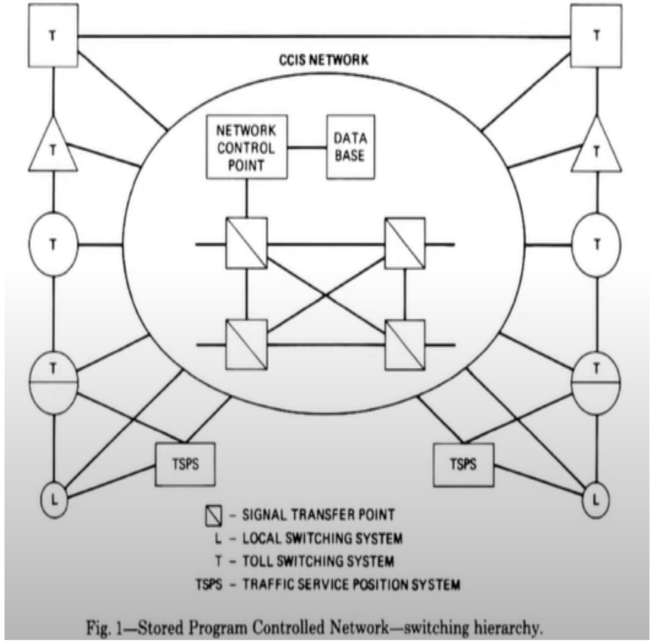
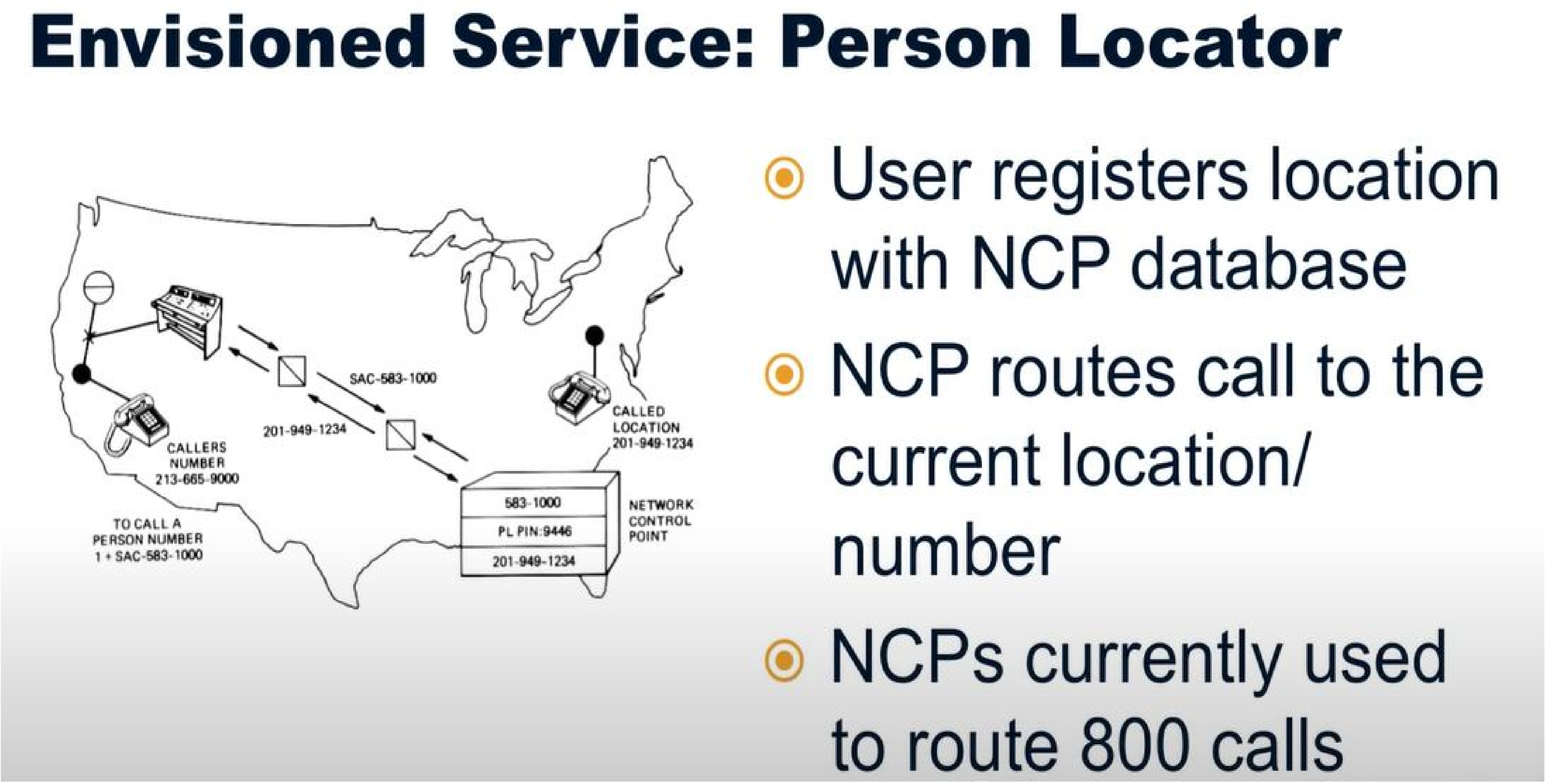

T
T
CCIS NETWORK
NETWORK CONTROL POINT
DATA BASE
T
T
T
T
T
T
TSPS
TSPS
L
L
⧅ - SIGNAL TRANSFER POINT
L - LOCAL SWITCHING SYSTEM
T - TOLL SWITCHING SYSTEM
TSPS - TRAFFIC SERVICE POSITION SYSTEM

Fig. 1—Stored Program Controlled Network—switching hierarchy.

1. Evolution of Supporting Technologies



Envisioned Service: Person Locator

SAC-583-1000
201-949-1234
CALLERS NUMBER 213-665-9000
TO CALL A PERSON NUMBER 1 + SAC-583-1000
CALLED LOCATION 201-949-1234
583-1000
PL PIN:9446
201-949-1234
NETWORK CONTROL POINT

- User registers location with NCP database
- NCP routes call to the current location/ number
- NCPs currently used to route 800 calls

1. Evolution of Supporting Technologies

I. **Programmability in Networks:** Active Networks (1990s)

- What are active networks?
- Motivation
- Technology behind active networks
- How do active networks relate to SDN
- The legacy of active networks

1. Evolution of Supporting Technologies

I. **Programmability in Networks:** Active Networks (1990s)

- What are active networks?
- Motivation
- Technology behind active networks
- How do active networks relate to SDN
- The legacy of active networks

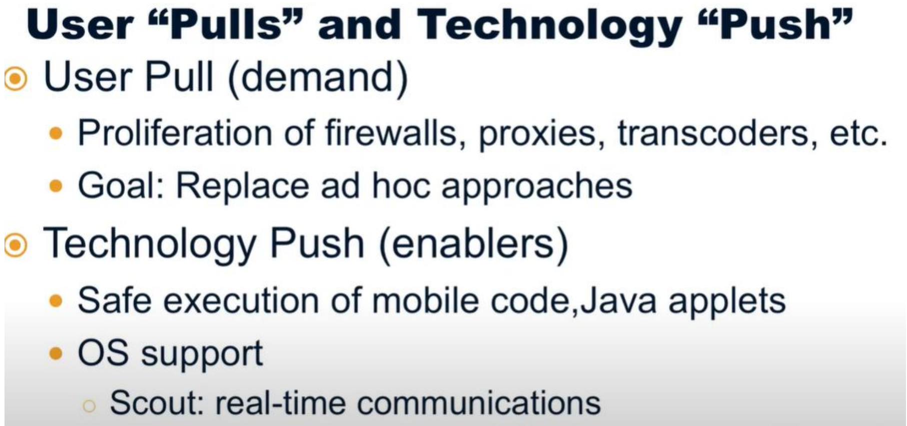
1. Evolution of Supporting Technologies

II. Programmability in Networks: Active Networks (1990s)

- **What are active networks?**
  - Networks where switches perform custom computations on packets.
- Examples:
  - Middlebox (Firewalls, proxies, application services)
  - Trace program running at each router
- **From where active networks came from?**
  - DARPA research community (1994-1995)
  - Identifies some of the problems with today’s networks:
    - Difficulty of integrating new technology
    - Poor performance due to redundant operations at several protocol layers
    - Difficulty accommodating new services

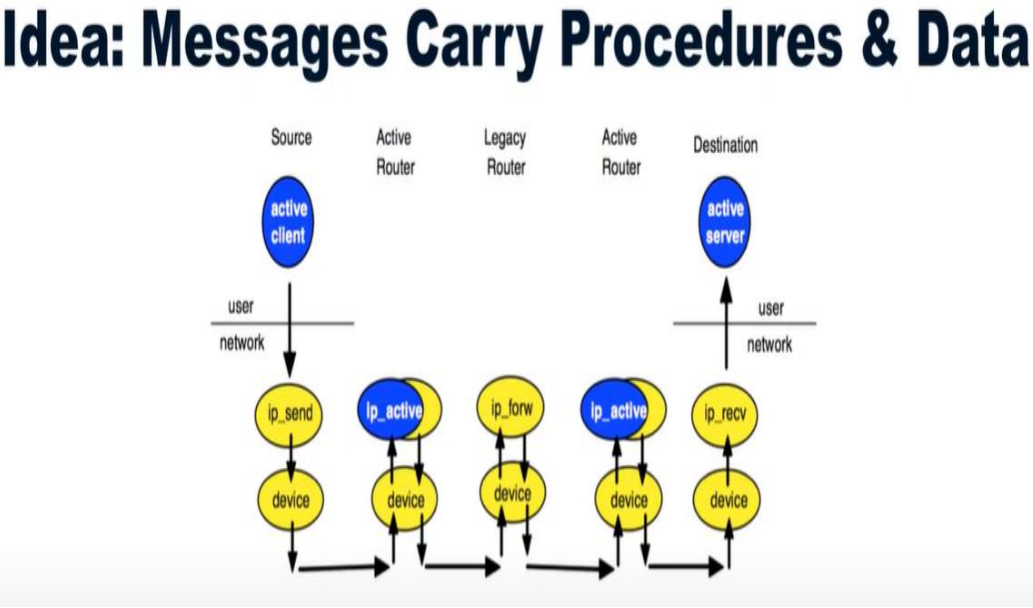
1. Evolution of Supporting Technologies

- **Motivation behind Active networks:**
  - Accelerating innovations
  - Internet innovations relies on consensus.
  - Takes ten years from prototype to deployment (Standardization, procurement, deployment)
  - Active nodes allow routers to download new services into the infrastructure.
  - That allows user driven innovations
  - Active routers coexist with legacy routers
  - Each programmable switch can perform additional processing



User "Pulls" and Technology "Push"

- User Pull (demand)
  - Proliferation of firewalls, proxies, transcoders, etc.
  - Goal: Replace ad hoc approaches
- Technology Push (enablers)
  - Safe execution of mobile code,Java applets
  - OS support
    - Scout: real-time communications



Idea: Messages Carry Procedures & Data

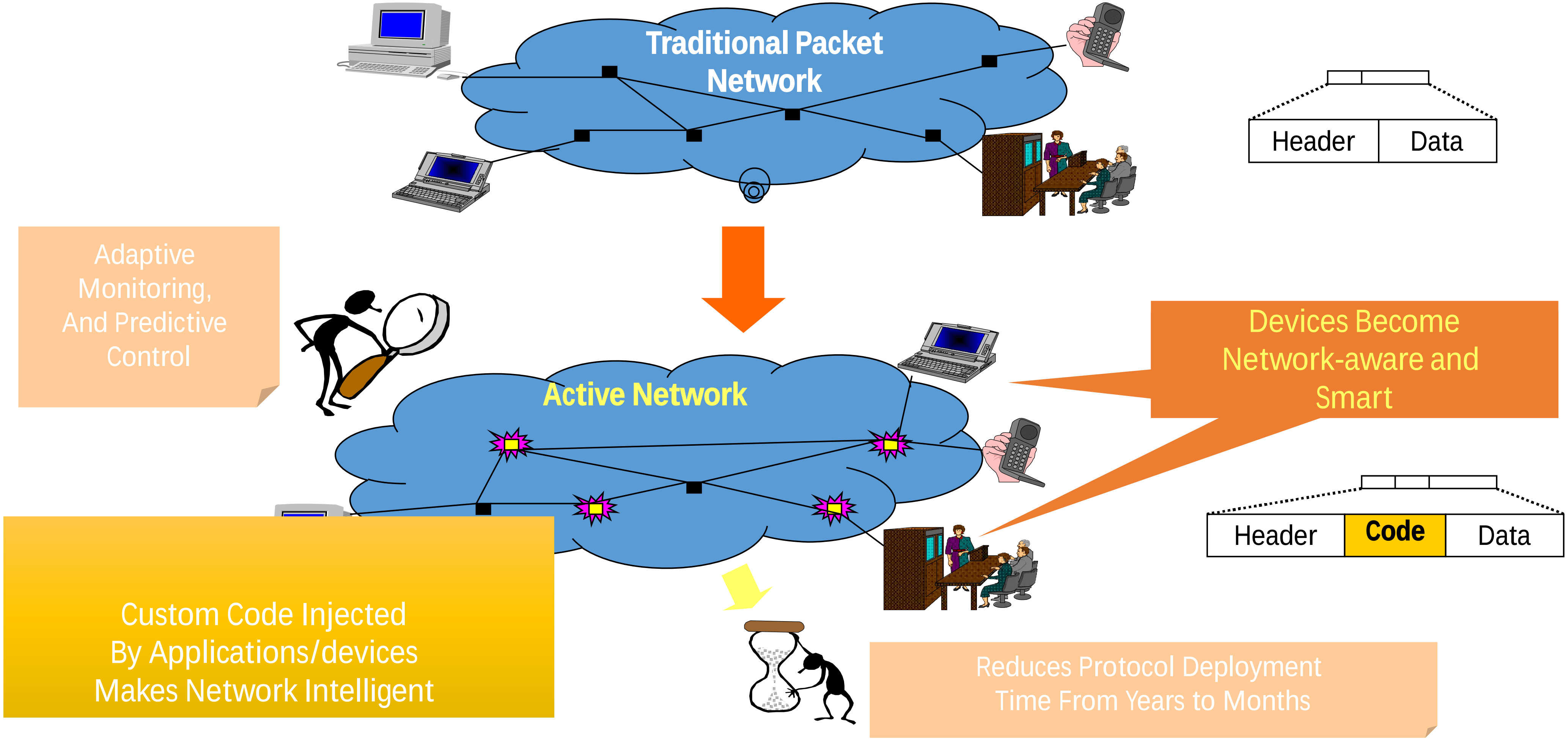
ACTIVE NETWORK

- Faster hardware not fully utilized.
- Enables more flexible network.
- De-couples protocol from transport.
- Minimizes requirement for global agreement.
- Enables on-the-fly experimentation.
- Enables faster deployment of New services.

Active Networking

- An active network is a network in which the **nodes are programmed** to perform custom operations on the messages that pass through the node.
- For example, a node could be programmed or customized to handle packets on an individual user basis or to handle multicast packets differently than other packets.
- Active network approaches are expected to be especially important in networks of mobile users.
- "Smart packets" use a special self-describing language that allows new kinds of information to be carried within a packet and operated on by a node.

Active Network: A Natural Evolution



Traditional Packet Network

Header | Data

Adaptive Monitoring, And Predictive Control

Active Network

Devices Become Network-aware and Smart

Header | **Code** | Data

Custom Code Injected By Applications/devices Makes Network Intelligent

Reduces Protocol Deployment Time From Years to Months

ACTIVE NETWORKS

Traditional Network

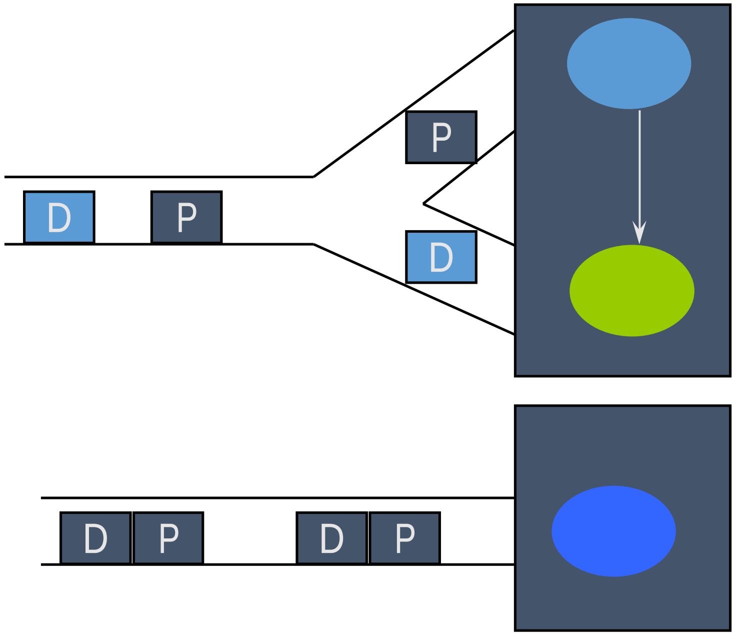
- Fossilized: Resistant to Change
- Layers of Complexity *O(4000)* RFCs
- Inability to Customize Quickly or Efficiently
- Lack of Security Paradigm
- Downward Side of the Innovation Curve

Active Network

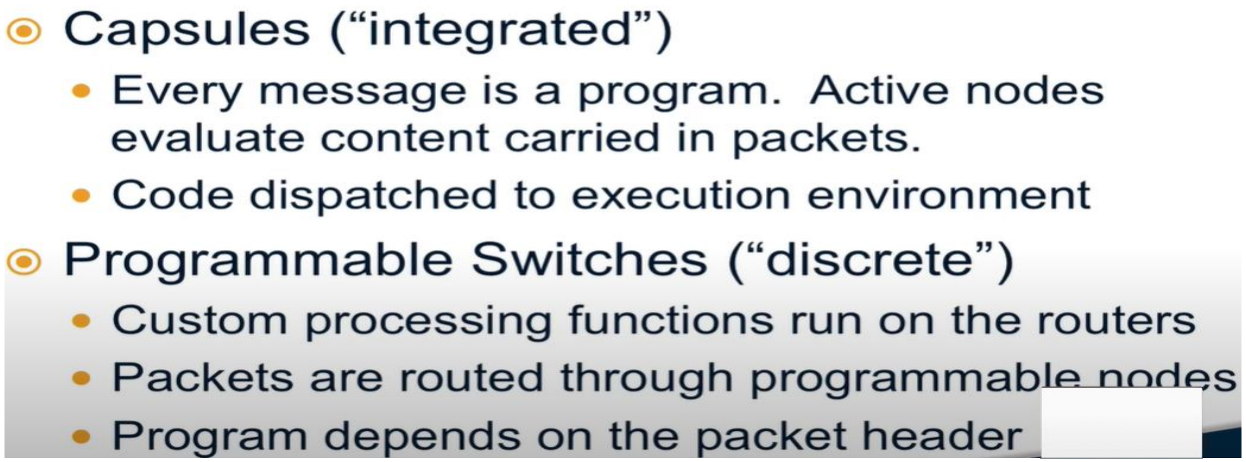
- Built for Change
- Reduced Complexity
- Rapid, Efficient Customization
- Security Paradigm Built-in
- Upward Innovation Path

Integrated Versus Discrete Approaches

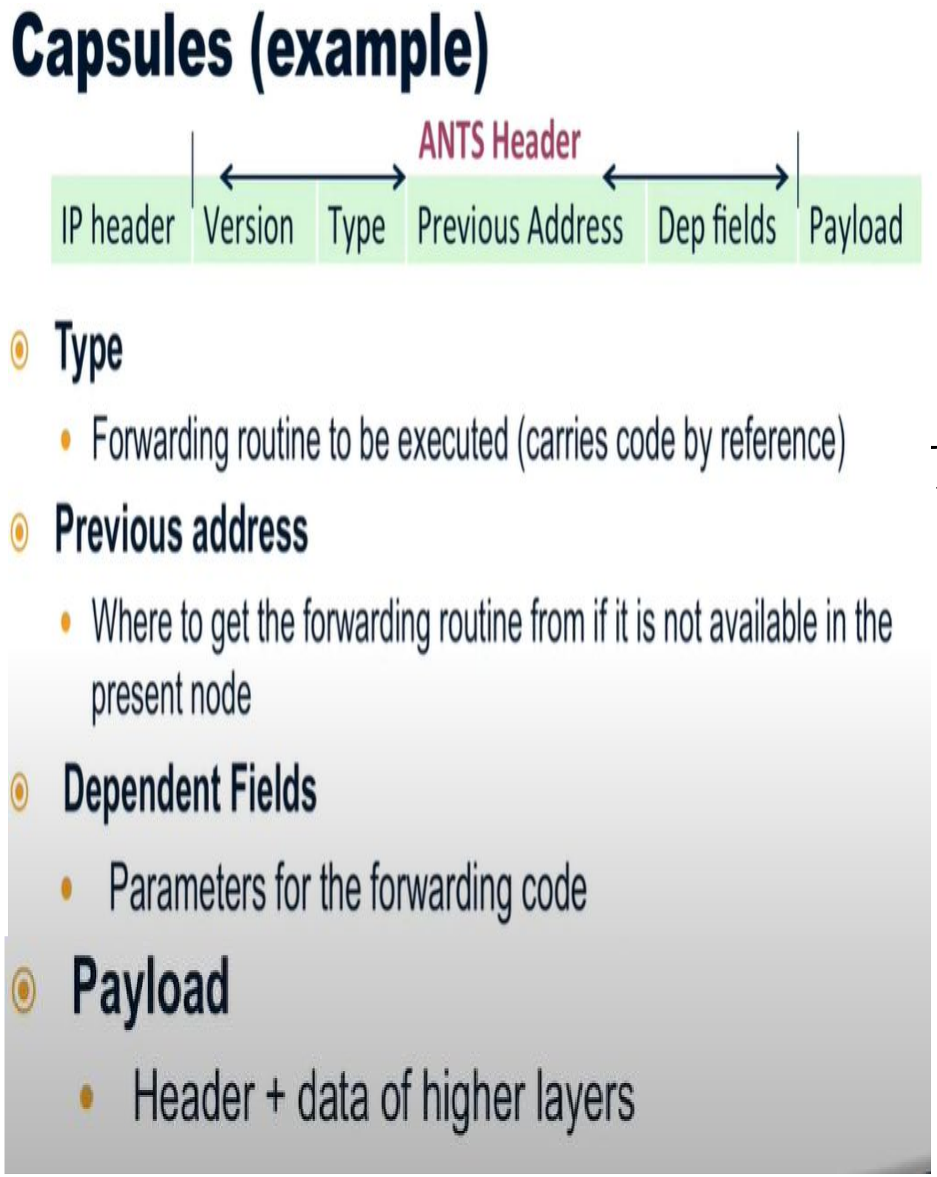
- Discrete Approach (Programmable Switches)
  - Programs (P) Injected Into Active Nodes Separately From Passive Data (D)
- Integrated Approach (Capsules)
  - Programs Integrated Into Every Packet Along With Passive Data



Active
Network Node



- Capsules (“integrated”)
  - Every message is a program. Active nodes evaluate content carried in packets.
  - Code dispatched to execution environment
- Programmable Switches (“discrete”)
  - Custom processing functions run on the routers
  - Packets are routed through programmable nodes
  - Program depends on the packet header

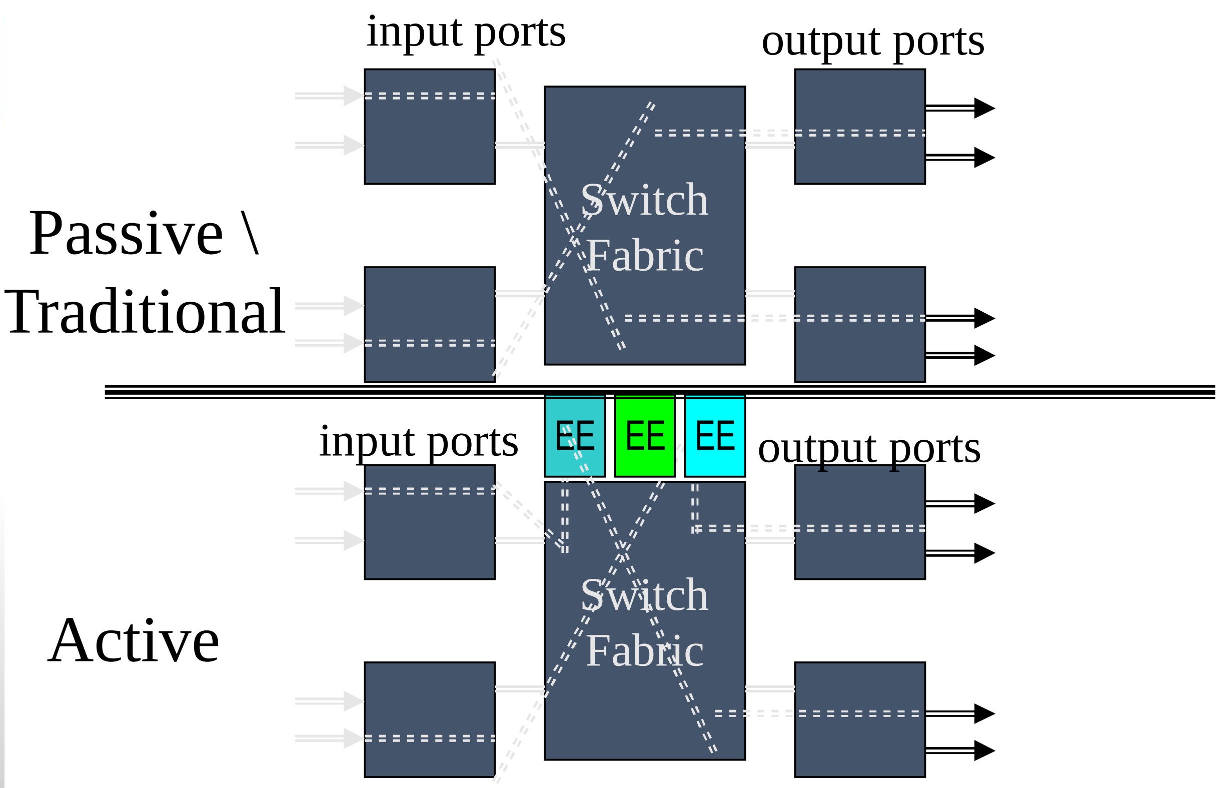


Capsules (example)

	ANTS Header				
IP header	Version	Type	Previous Address	Dep fields	Payload

- **Type**
  - Forwarding routine to be executed (carries code by reference)
- **Previous address**
  - Where to get the forwarding routine from if it is not available in the present node
- **Dependent Fields**
  - Parameters for the forwarding code
- **Payload**
  - Header + data of higher layers

Hardware Reference Model



- Legacy of Active networks for SDN
  - Programmable functions in network to enable innovations
  - Programmable switches
  - Demultiplexing programs on packets headers
  - Planetlab, Flowvisor, GENI, etc. all use this
  - Paying attension to middleboxes and how these functions are composed

1. Evolution of Supporting Technologies

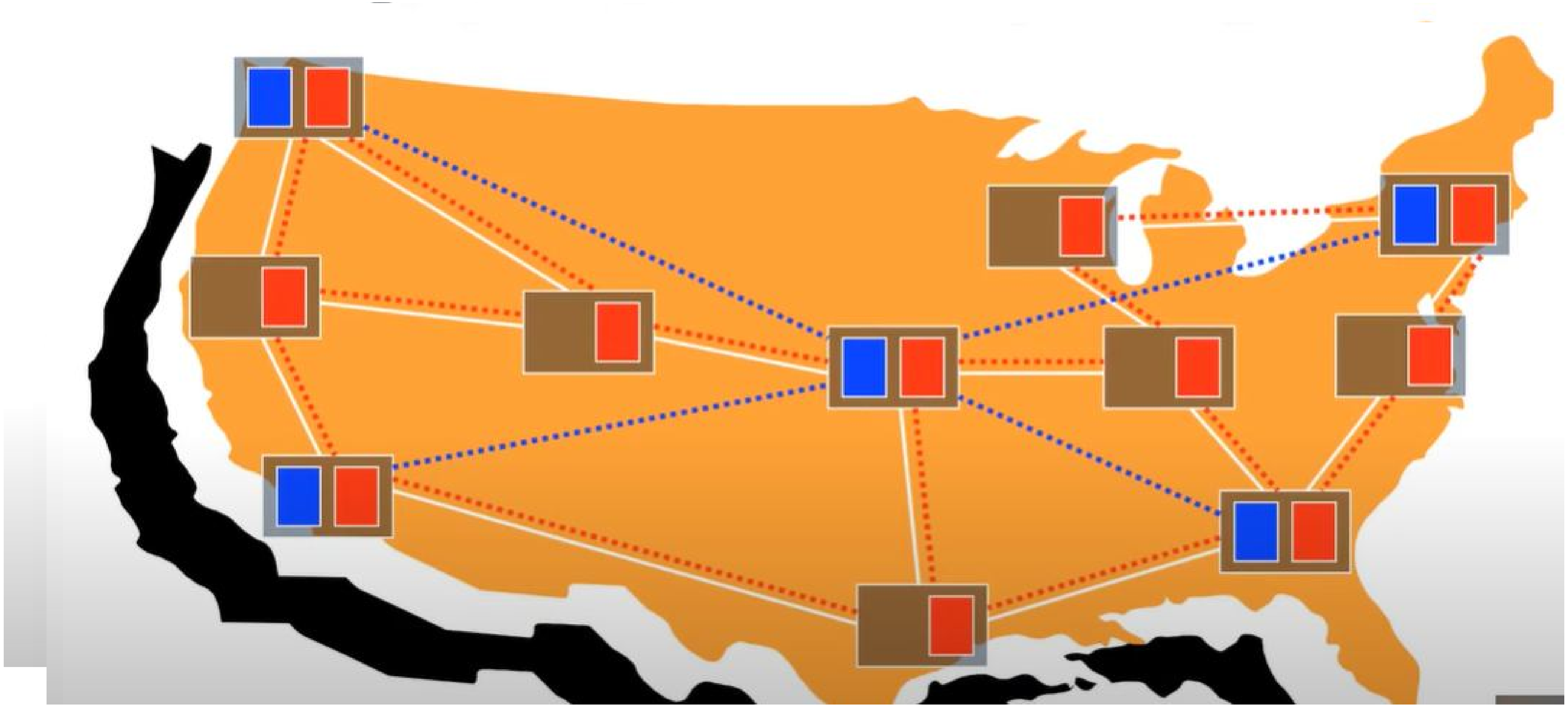
III. **Network Virtualization:** Switchlets, XEN, VINI (1990s to 2000s)

- Representation of one or more logical network topologies on the same infrastructure
- Many different instantiations:
  - Virtual LANs (VLANs-1990s)
  - Various technologies and network testbeds
  - Today: VMWare, Nicira, etc.
- **Benefits:**
  1. **Sharing:**
     - Multiple logical routers on a single platform
     - Resource isolation in CPU, memory, bandwidth, forwarding tables,……

1. Evolution of Supporting Technologies

2. Customizability:

- Customizable routing and forwarding software's
- General purpose CPUs for control plane
- Network processors and FPGAs for data plane



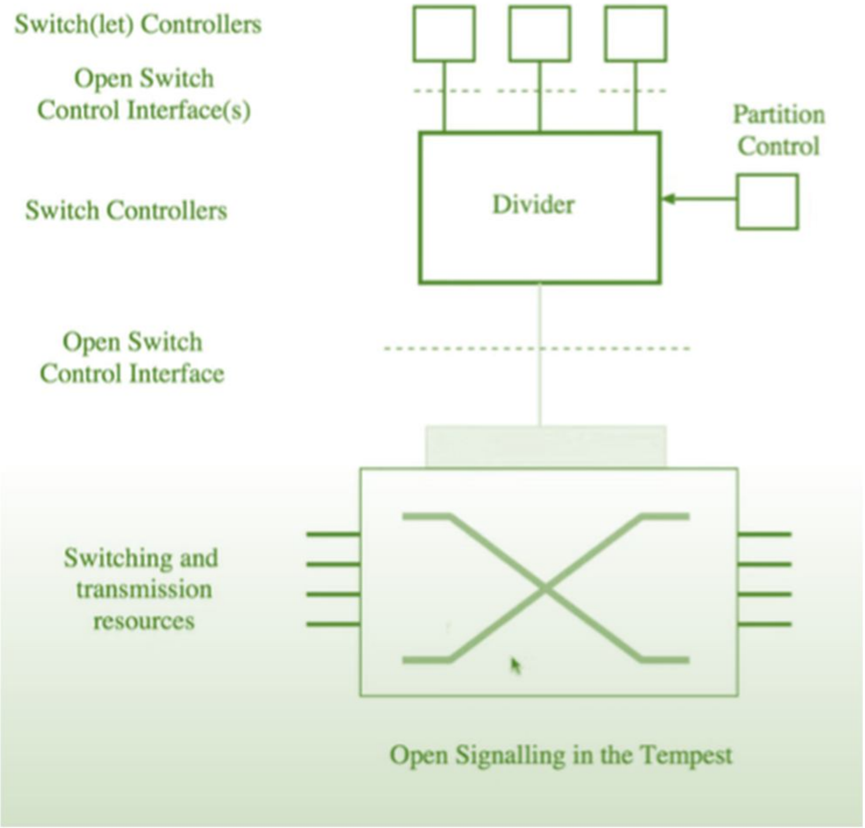
1. Evolution of Supporting Technologies

- Three Examples of Virtual networks:
    1. **Tempest:** Switchlets (1998)
        - Separation of control framework from switches
        - Virtualization of the switch
    2. **VINI:** A Virtual Network Infrastructure (2006)
        - Virtualization of the network infrasrtucture
    3. **Cabo:** Separates infrastructure, services (2007)

1. Evolution of Supporting Technologies

1. **Tempest:** Switchlets (1998)
   - Idea is to operate multiple control architectures above ATM network
   - Separation of switch controller and and fabric via open signaling
   - Partitioning of switch resources across controllers
   - Partitions port space, bandwidth, buffers
   - Different controllers control each switchlet

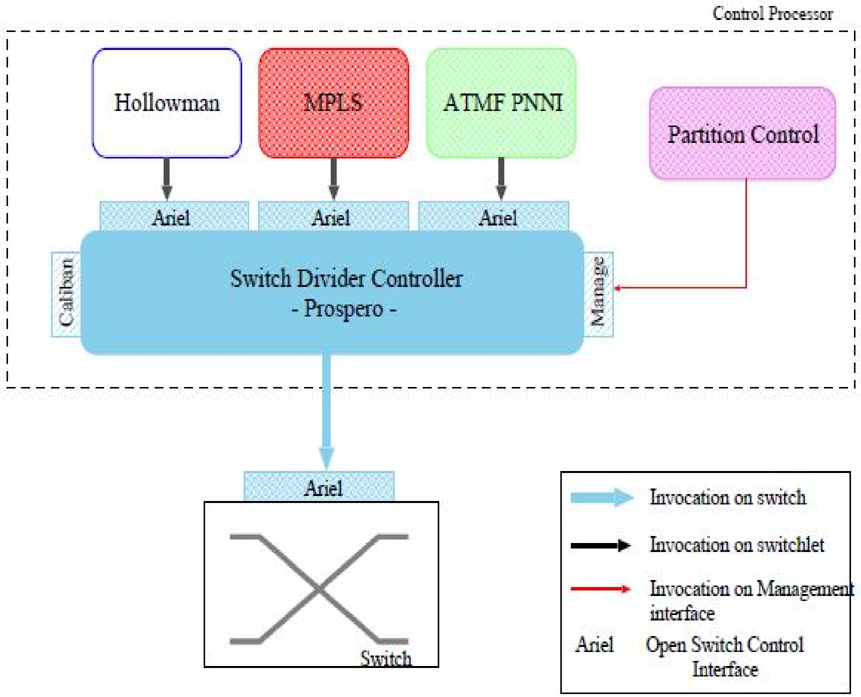
Architecture


Switch(let) Controllers
Open Switch Control Interface(s)
Switch Controllers
Divider
Partition Control
Open Switch Control Interface
Switching and transmission resources
Open Signalling in the Tempest

1. Evolution of Supporting Technologies

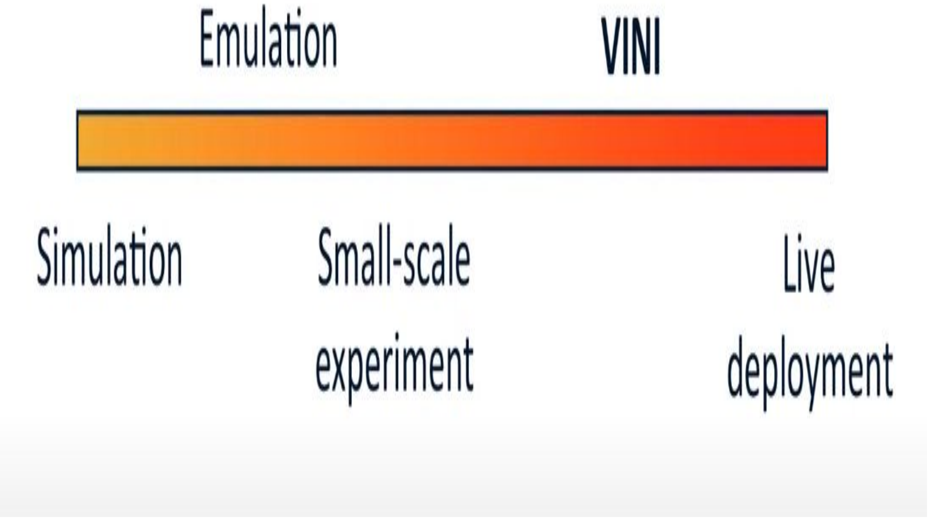
1. **Tempest:** Switchlets (1998)

- Tempest consists of following functional components:
  1. **Prospero Switch Divider:** Allows the resources of the switch to be portioned between control architectures
  2. **Ariel switch independent control interface:** which a control architecture uses to control its part of switch
  3. **Caliban Switch Management Interface:** which a network operator uses to manage its part of switch, e.g. test performance, check for faults
  4. **Bootsrap Virtual Network (MPLS):** which supports the communication needs of Tempest infrastructure itself
  5. **Network Builder (ATMF):** Allows virtual networks to be created
  6. **Hollowman Control Architecture:** which is both a fully functional ATM control and a set of components that third parties can use to build their own control architecture.



1. Evolution of Supporting Technologies

2. **VINI:** A Virtual Network Infrastructure (2006)
   - Bridge the gap between “lab experiments” and live experiments at scale.
   - Runs real routing software
   - Expresses realistic network conditions
   - Gives control over network events
   - Carries traffic on behalf of real users
   - Shared among many experiments
   - Allows network researchers to evaluate their protocols and services in a realistic environment, with a high degree of control over network conditions.



1. Evolution of Supporting Technologies

- To provide researchers flexibility in designing their experiments, VINI supports simultaneous experiments with arbitrary network topologies on a shared physical infrastructure.
- VINI research on network virtualization focuses on two main scenarios:
    1. It considers the role of virtualization in running multiple experiments simultaneously in a shared experimental facility. For example, the NSF GENI initiative focuses on the design and deployment of a shared, wide-area experimental facility to support a wide range of research in networking and distributed systems. The VINI project is a step in that direction, supporting experimentation with new routing, forwarding, and addressing schemes on a shared facility built on top on general-purpose processors.
    2. VINI considers the role of virtualization to support multiple architectures simultaneously as a long-term solution for the future Internet. The Cabo project explores the benefits of running customized architectures, as well as how a virtualized system enables an economic refactoring of a future Internet into infrastructure providers (that own and operate the equipment) and service providers (who lease virtual components and offer end-to-end services to users). These projects grapple with the technical challenges of providing a virtualized, programmable substrate that operates at high speed; the Cabo project must address the additional challenges of building a substrate that can operate without any dependency on the existing Internet.

1. Evolution of Supporting Technologies

1. **Tempest:** Switchlets (1998)
    - Separation of control framework from switches
    - Virtualization of the switch
    1. **VINI:** A Virtual Network Infrastructure (2006)
        - Virtualization of the network infrasrtucture
    2. **Cabo:** Separates infrastructure, services (2007)

Separation of Data Plane & Control Plane

- **Control Plane :**
  - Makes decisions about where traffic is sent.
  - Control plane packets are *destined to* or locally *originated by* the router itself.
  - The control plane functions include the system configuration, management, and exchange of routing table information.
  - The route controller exchanges the topology information with other routers and constructs a routing table based on a routing protocol, for example, RIP, OSPF or BGP.
  - Control plane packets are processed by the router to update the routing table information.
  - It is the *Signaling* of the network.
  - Since the control functions are not performed on each arriving individual packet, they do not have a strict speed constraint and are less time-critical

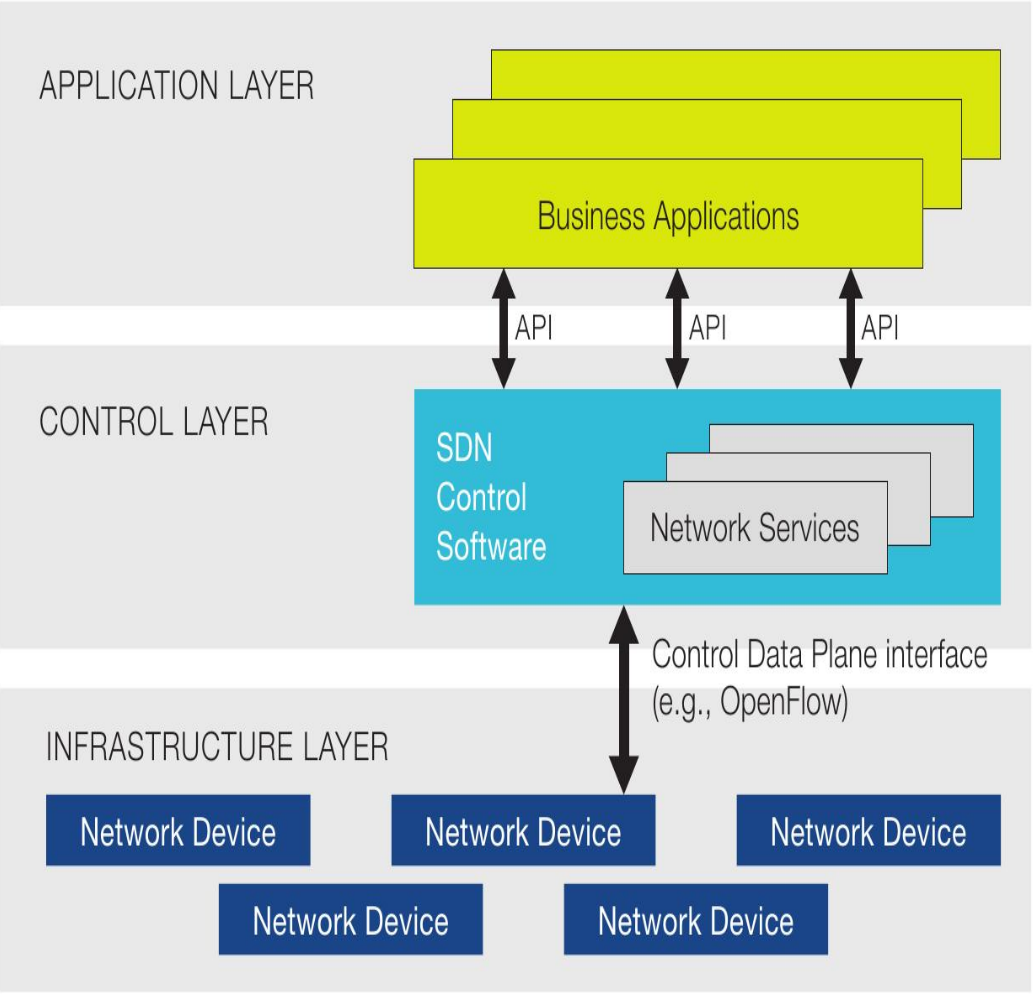
- **Data Plane**
  - Also known as Forwarding Plane
  - Forwards traffic to the next hop along the path to the selected destination network according to control plane logic
  - Data plane packets go *through* the router
  - The routers/switches use what the control plane built to dispose of incoming and outgoing frames and packets
- **Example 1**

The protocol or application itself doesn’t really determine whether the traffic is control, management, or data plane, but more importantly how the router processes it.

Consider a 3 router topology with routers R1, R2 and R3. Lets say a Telnet session is established from R1 to R3. On both of these routers the packets need to be handled by the control/management plane. However from R2′s perspective this is just data plane traffic that is transiting between its links.

Software Defined Network (SDN)

- ‘Decoupling hardware from software’ i.e separation of data plane & control plane
- Decoupling these two planes involves leaving the data plane with network hardware and moving the control plane into a software layer.
- Network Virtualization is one of the application of SDN – SDN can be leveraged as a tool to achieve Network Virtualization.



Advantages of SDN:

- **More configuration accuracy, consistency and flexibility :**
  - As described earlier, traditional networking requires configurations to be executed on a manual, device by- device basis.
  - A key characteristic of the SDN approach, is automating this process, enabling an administrator to manage the entire network as if it were a single device.
- **Data flow optimization :**
  - Instead of having a single path from the source of communication flow to its destination, a SDN controller is able to identify multiple paths per flow.
  - Furthermore, this approach allows the flow’s traffic to be split across multiple nodes.
  - Network performance and scalability is enhanced by optimizing the network path for a particular data flow based on the source and destination nodes1

Traditional Networking	Software Defined Networking
They are Static and inflexible networks. They are not useful for new business ventures. They possess little agility and flexibility	They are programmable networks during deployment time as well as at later stage based on change in the requirements. They help new business ventures through flexibility, agility and virtualization.
They are Hardware appliances.	They are configured using open software.
They have distributed control plane.	They have logically centralized control plane.
They use custom ASICs and FPGAs.	They use merchant silicon.
They work using protocols.	They use APIs to configure as per need.

IETF ForCES: **Separation of Forwarding and Control Planes**

- *Internet Engineering Task Force (IETF)*
- *Forwarding and Control Element Separation* (ForCES)
- The general idea of ForCES was to provide simple hardware-based forwarding entities at the foundation of a network device and software-based control elements above
- The functional components of ForCES are as follows:

1. **Forwarding Element : (FE)**

   ➢ Typically implemented in hardware and located in the network.

   ➢ Responsible for enforcement of the forwarding and filtering rules that it receives from the *controller*.

2. **Control Element. (CE) :**

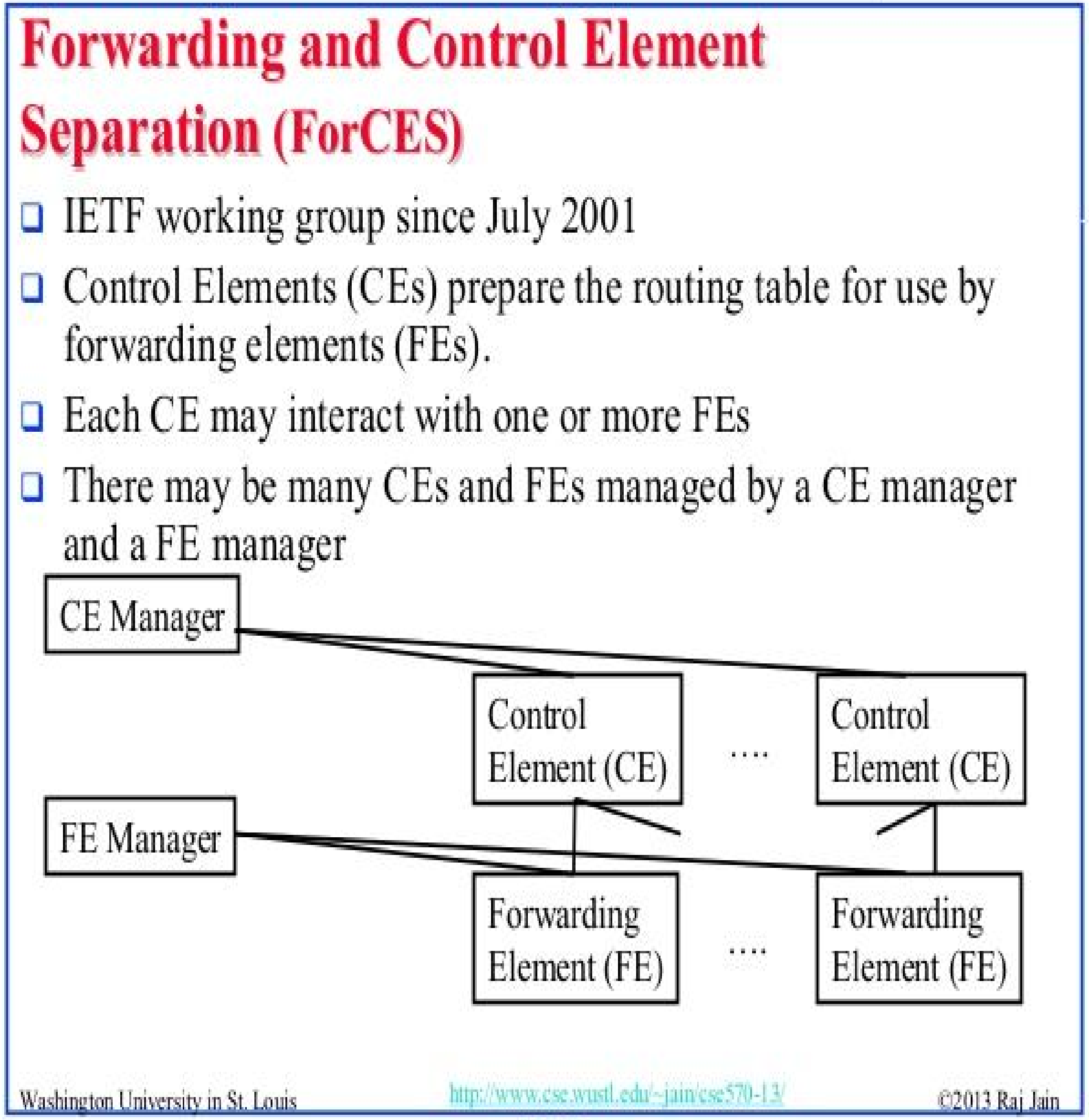
   ➢ Concerned with the coordination between the individual devices in the network and for communication forwarding and routing information to the FE’s below.

3. **Network Element (NE) :**

   The *Network Element* (NE) is the actual network device that consists of one or more FEs and one or more CEs.

4. **ForCES Protocol. :**

   Used to communicate information back and forth between FEs and CEs.



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

CE Manager
Control Element (CE)
....
Control Element (CE)
FE Manager
Forwarding Element (FE)
....
Forwarding Element (FE)

Washington University in St. Louis http://www.cse.wustl.edu/~jain/cse570-13/ ©2013 Raj Jain

16-16

Characteristics of SDN

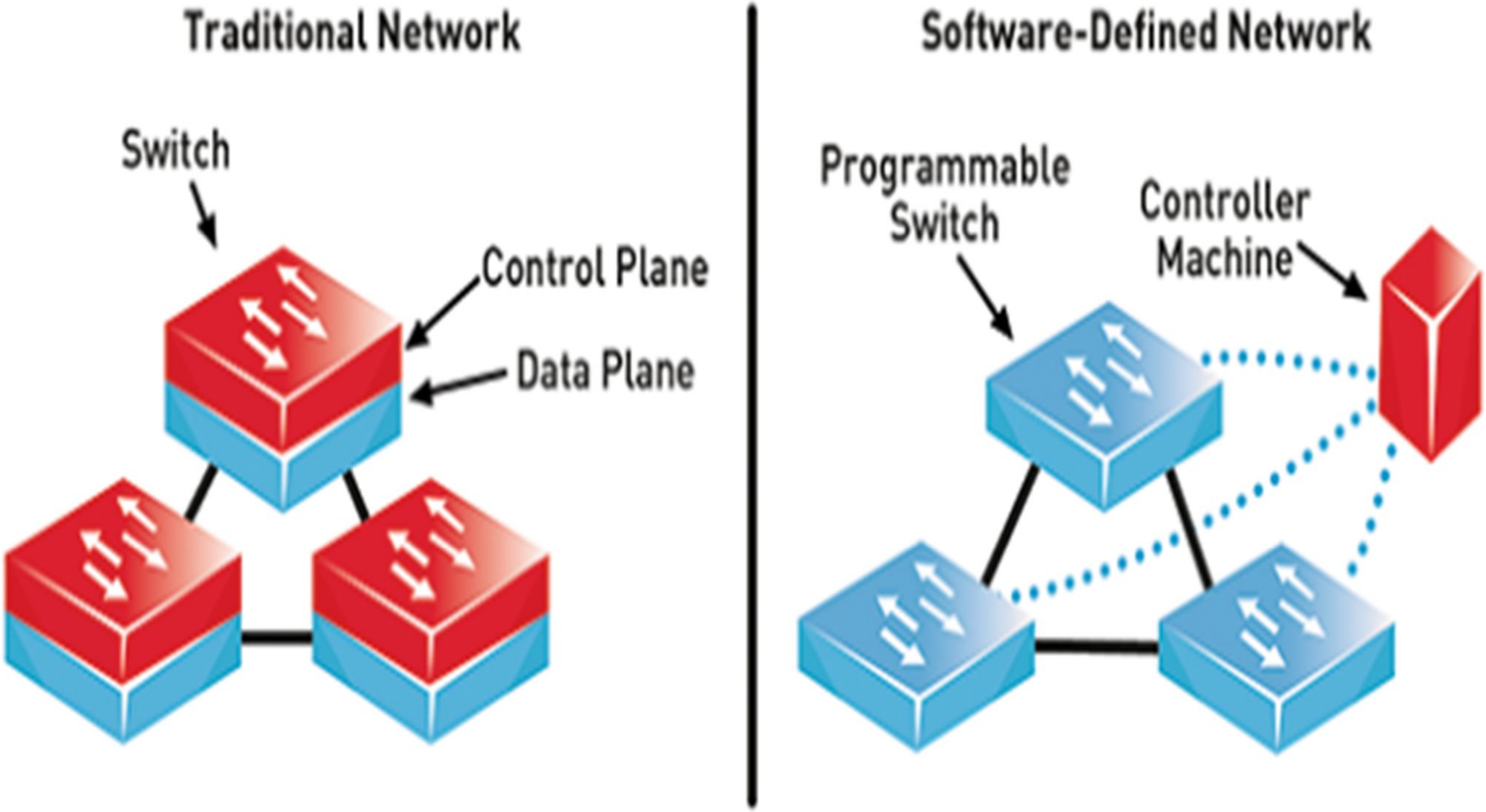
1. Plane Separation
2. A Simplified Device
3. Centralized Control
4. Network Automation And Virtualization,
5. Openness.

1. Plane Separation

- **Forwarding Plane Functionality**:

1. Includes logic and tables for choosing how to deal with incoming packets based on characteristics such as MAC address, IP address, and VLAN ID.
2. How to provide a way to deal with arriving packets. It may *forward*, *drop*, *consume*, or *replicate* an incoming packet.
3. For basic forwarding, the device determines the correct output port by performing a lookup in the address table in the hardware ASIC. (An ASIC (application-specific integrated circuit) is a microchip designed for a special application, such as a particular kind of transmission protocol or a hand-held computer.)
4. A packet may be dropped due to buffer overflow conditions or due to specific *filtering*. for example. Special-case packets that require processing by the control or management planes are consumed and passed to the appropriate plane.

- **Control Plane Functionality :**
  1. It includes the protocols, logic, and algorithms that are used to program the forwarding plane.
  2. Determines how the forwarding tables and logic in the data plane should be programmed or configured.
- In a traditional network each device has its **own control plane**.
- The primary task of that control plane is to **run routing** or **switching protocols** so that all the distributed forwarding tables on the devices throughout the network stay synchronized.
- The most basic outcome of this synchronization is the prevention of loops.
- Co-reside in legacy Internet switches.
- In SDN, the control plane is removed from switching device onto a centralized controller.


Traditional Network
Switch
Control Plane
Data Plane
Software-Defined Network
Programmable Switch
Controller Machine

2. A Simple Device and Centralized Control

- Instead of 100’s of 1000’s of lines of complicated control plane software running on the device (e.g. Switch) and allowing the device to behave autonomously, that software is removed from the device and placed in a centralized controller.
- Software-based controller manages the network using higher-level policies.
- The controller provides primitive instructions to simplified devices when appropriate in order to allow them to make fast decisions about how to deal with incoming packets.

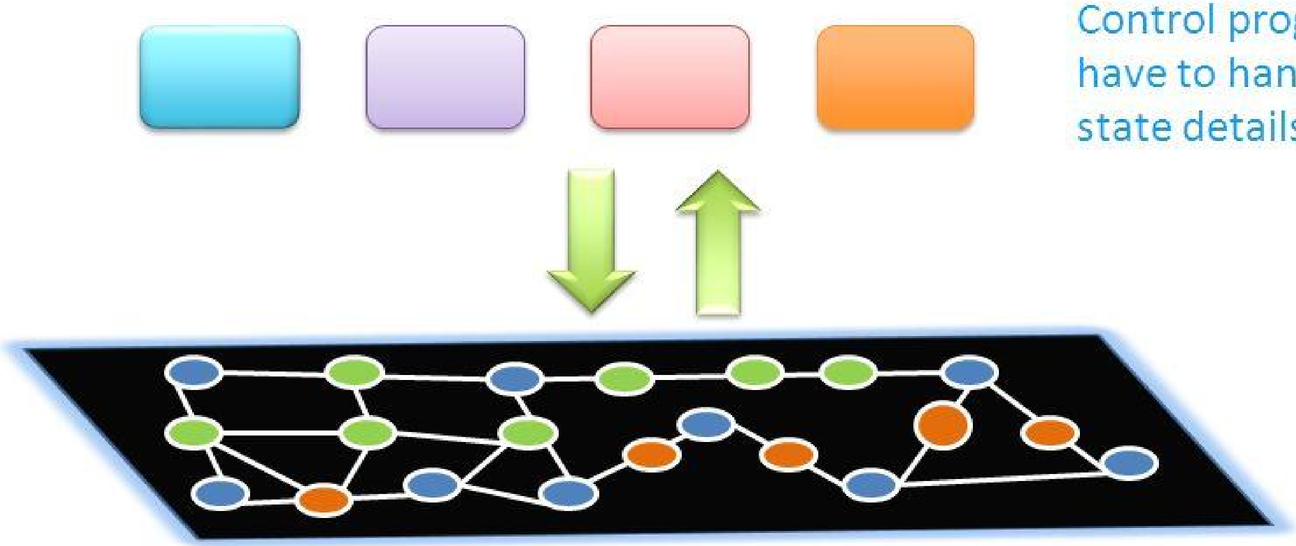
3. Network Automation and Virtualization

- SDN is based on three basic abstractions:
  1. Distributed State
  2. Forwarding
  3. Configuration

1. Distributed State Abstraction:

- Network actually comprised of many machines & each with its own state, collaborating to solve network-wide problems.
- Distributed state abstraction provides the network programmer with global view of network.

State Distribution Abstraction



Control program should not have to handle distributed-state details

Proposed abstraction: global network view

Abstracted away by
Network Operating System

State Collection
Dissemination & Synchronization
Application Isolation

2. Forwarding Abstraction:

- Allows programmer to specify the necessary forwarding behavior without any knowledge of vender specific hardware.

3. Configuration Abstraction:

- Sometimes called the *specification* abstraction.
- Must be able to express the desired goals of the overall network without getting lost in the details of how the physical network will implement those goals.
- Working with the network through this configuration abstraction is really network virtualization at its most basic level.

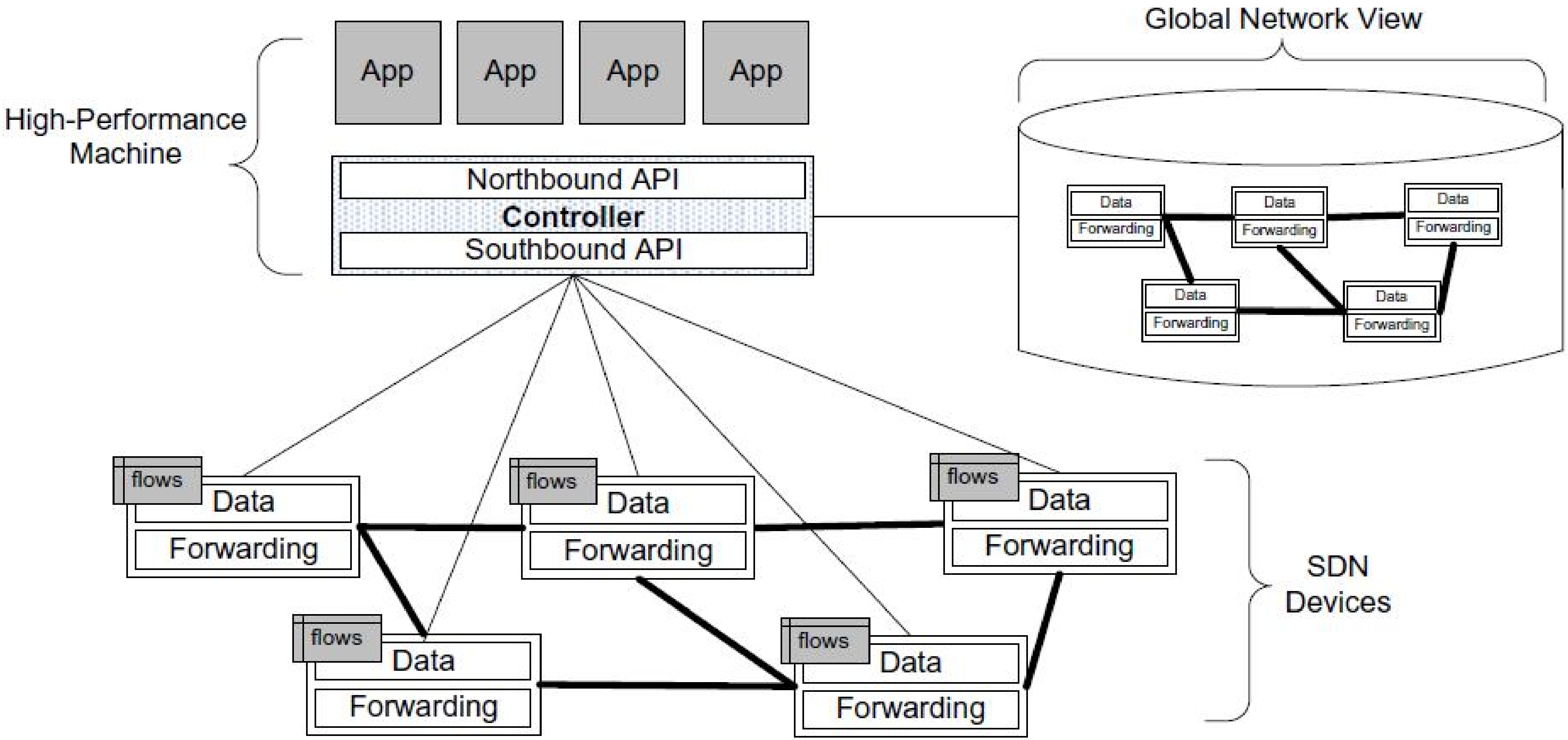
- The centralized software-based controller in SDN provides an open interface on the controller to allow for automated control of the network.
- In the context of Open SDN, the terms *northbound* and *southbound* are often used to distinguish whether the interface is to the applications or to the devices.
- The **southbound API** is the OpenFlow interface that the controller uses to program the **network devices**.
- The controller offers a **northbound API**, allowing **software applications** to be plugged into the controller.
- Software applications then provide the algorithms and protocols that can run the network efficiently.

- The northbound API is intended to provide an abstraction of the network devices and topology.
- So, northbound API provides a generalized interface that allows the software above it to operate without knowledge of the individual characteristics and specialty of the network devices themselves.
- In this way, applications can be developed in such a way that they work over a wide array of manufacturers’ equipment that may differ substantially in their implementation details.
- One of the results of this level of abstraction is that it provides the ability to virtualize the network, decoupling the network service from the underlying physical network.

4. Openness

- Keeping open both the northbound and southbound interfaces to the SDN controller will allow for **research** into new and innovative methods of network operation.
- Research institutions as well as entrepreneurs can take advantage of this capability in order to easily experiment with and test new ideas.
- The presence of these open interfaces also encourages SDN-related open source projects.
- In addition to facilitating research and experimentation, open interfaces permit equipment from different vendors to interoperate.
- This normally produces a competitive environment which lowers costs to consumers of network equipment.

How SDN Works ?



- **Basic components of SDN:** The SDN devices, The controller, and The applications

1. The SDN Devices:

- Contain forwarding functionality for deciding what to do with each incoming packet.
- Also contain the data that drives forwarding decisions.
- The **data** itself is actually **represented by** the **flows** defined by the controller, as depicted in the upper-left portion of each device.
- A **flow** describes a set of packets transferred from one network endpoint (or set of endpoints) to another endpoint (or set of endpoints).
- The endpoints may be defined as IP address-TCP/UDP port pairs, VLAN endpoints, layer three tunnel endpoints, and input ports.

OF flow table

	match fields	actions	counters
flow entry →	match fields	actions	counters
	match fields	actions	counters
flow miss entry →		actions	counters

The flow table is populated only by the controller

The incoming packet is matched by comparing to match fields

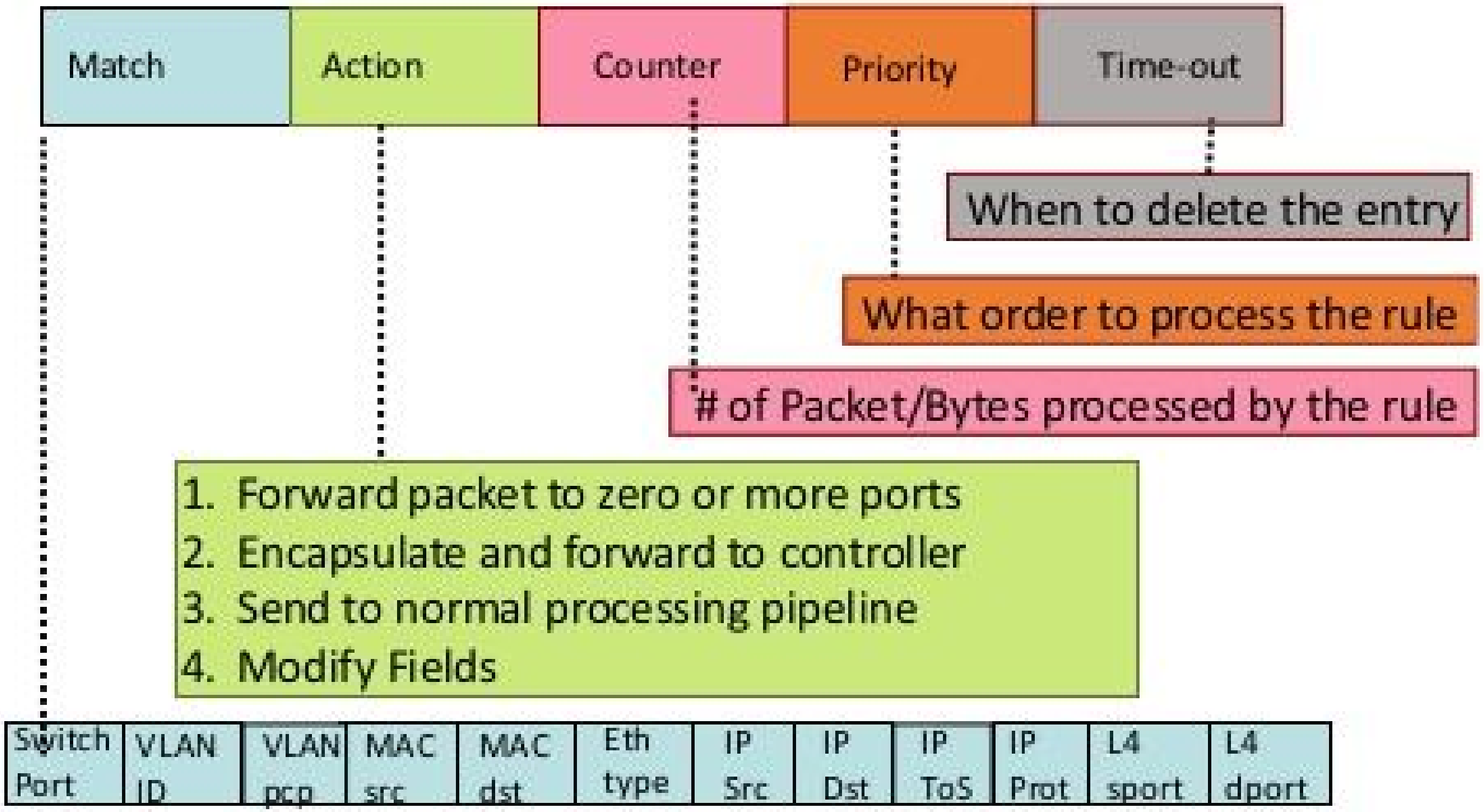
For simplicity, matching is exact match to a static set of fields

If matched, actions are performed and counters are updated

Entries have priorities and the highest priority match succeeds

Actions include editing, metering, and forwarding

OpenFlow: Anatomy of a Flow Table Entry



Match	Action	Counter	Priority	Time-out

When to delete the entry

What order to process the rule

# of Packet/Bytes processed by the rule

1. Forward packet to zero or more ports
2. Encapsulate and forward to controller
3. Send to normal processing pipeline
4. Modify Fields

Switch Port	VLAN ID	VLAN pcp	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP ToS	IP Prot	L4 sport	L4 dport

- **A flow table** present on the network device and consists of a series of flow entries and the actions to perform when a packet matching that flow arrives at the device.
- When the SDN device receives a packet, it consults its flow tables in search of a match.
- If the SDN device **finds a match**, it takes the appropriate configured action, which usually entails **forwarding the packet**.
- If it **does not find a match**, the switch can either **drop** the packet or **pass it to the controller**, depending on the version of OpenFlow and the configuration of the switch.

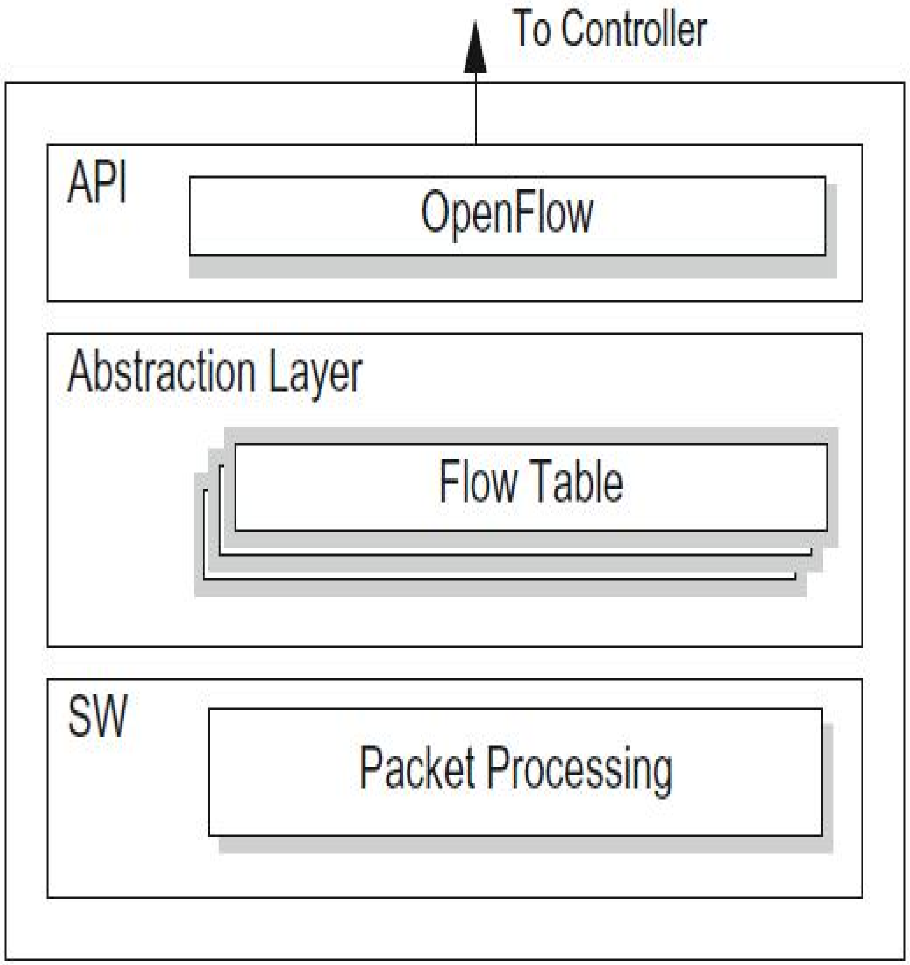
- An SDN device is composed of: an API for communication with the controller, an abstraction layer, and a packet-processing function.
- In the case of a virtual switch, this packet-processing function is packet processing software.
- In the case of a physical switch, the packet-processing function is included in the hardware for packet-processing logic.
- The abstraction layer includes one or more flow tables.

1.1 Flow Tables:

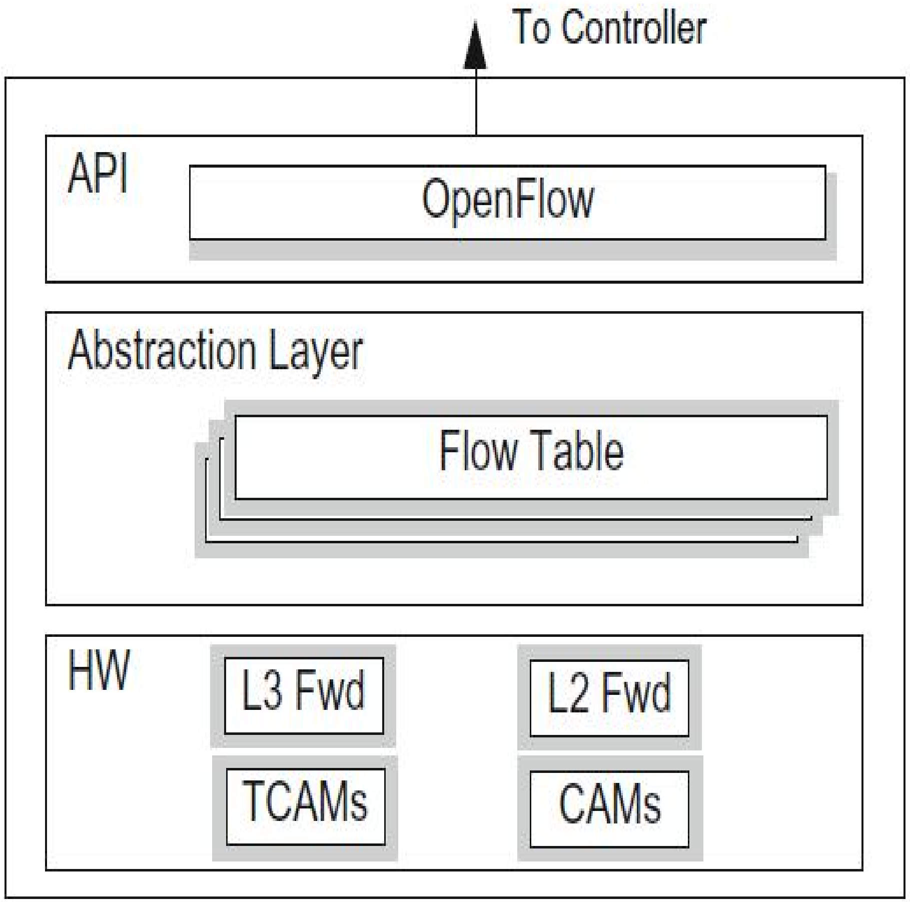
- Fundamental data structures in an SDN device.
- Flow tables consist of a number of prioritized flow entries, each of which typically consists of two components: *match fields* and *actions*.
- Match fields are used to compare against incoming packets.
- Actions are the instructions that the network device should perform if an incoming packet matches the match fields specified for that flow entry.

1.2 SDN Software Switches :

- Simplest means of creating an SDN device, because the flow tables, flow entries, and match fields involved are easily mapped to general software data structures, such as sorted arrays and hash tables.
- Two software SDN devices produced by different development teams will behave consistently.
- For network devices that must run at high speeds, such as 10 Gbps, 40 Gbps, and 100 Gbps, only hardware implementations are feasible.



SDN Software switch anatomy



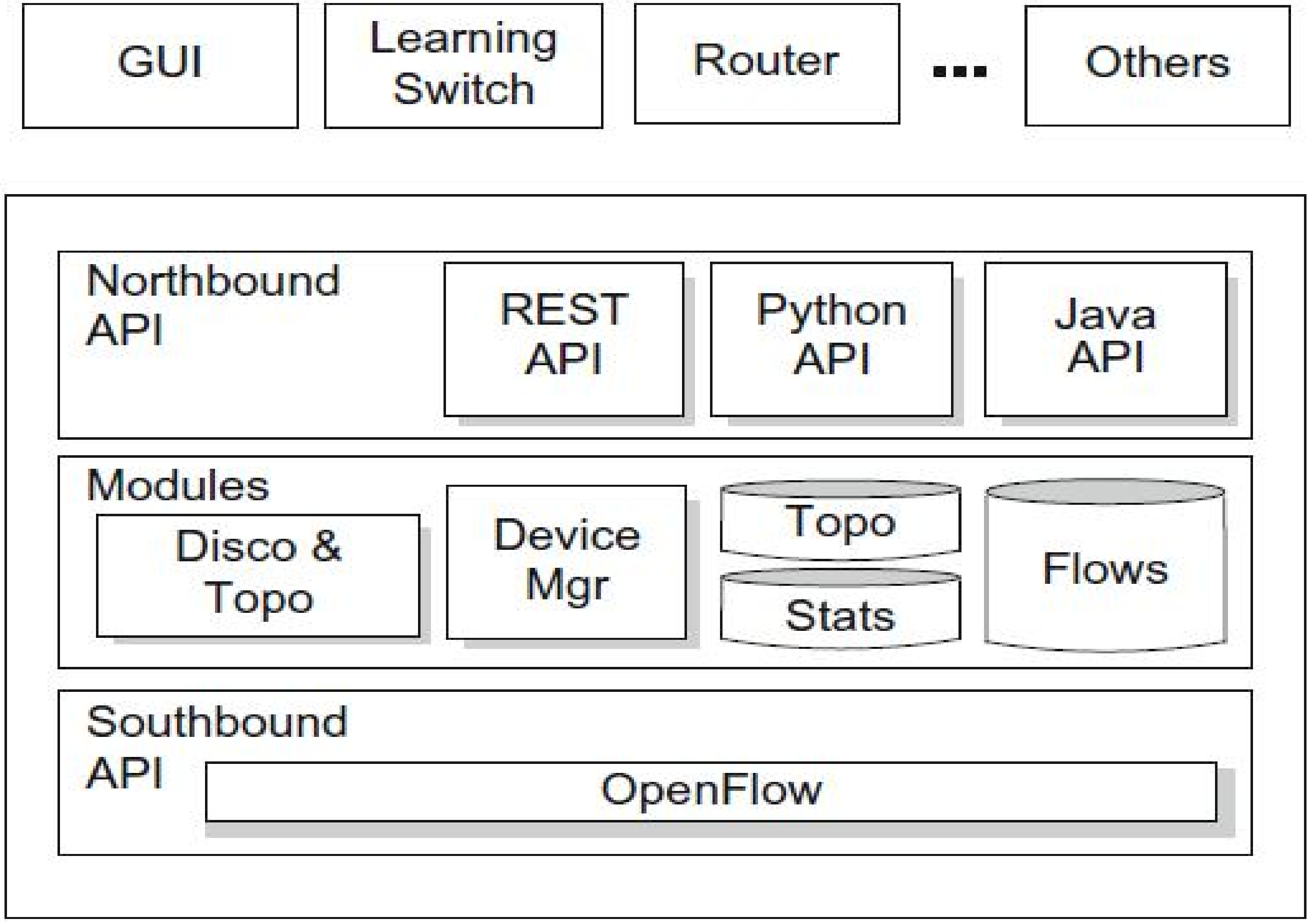
SDN hardware switch anatomy

1.3 Hardware SDN Switch:

- Operates much faster than software counterparts.
- Thus, are more applicable to performance-sensitive environments, such as in data centers and network cores.
- Hardware includes the layer two and layer three forwarding tables, usually implemented using ***content-addressable memories*** (CAMs) and ***ternary content-addressable memories*** (TCAMs).
- Layer 2 switching uses the media access control address (MAC address)
- Layer two forwarding table is used for making MAC-level forwarding decisions. (Switch)
- Layer 3 switching is solely based on (destination) IP address
- Layer three forwarding table is used for making IP-level routing decisions. (Router)

2. SDN Controller:

- Responsible for abstracting the network of SDN devices it controls &
- Presenting an abstraction of these network resources to the SDN applications running above.
- Allows S/W Applications to define flows on devices and to help the application respond to packets that are forwarded to the controller by the SDN devices.
- On right side of the controller it maintains a view of the entire network that it controls.
- This permits it to calculate optimal forwarding solutions for the network in a deterministic, predictable manner
- Maintains a view of the entire network, implements policy decisions, controls all the SDN devices that comprise the network infrastructure, and provides a northbound API for applications.


GUI
Learning Switch
Router
...
Others
Northbound API
REST API
Python API
Java API
Modules
Disco & Topo
Device Mgr
Topo
Stats
Flows
Southbound API
OpenFlow

- Controllers often come with their own set of common application modules, such as a learning switch, a router, a basic firewall, and a simple load balancer.
- These are really SDN applications, but they are often bundled with the controller.
- Southbound API is used to interface with the SDN devices.
- Currently no northbound counterpart to the southbound OpenFlow standard.
- This lack of a standard for the controller-to-application interface is considered a current deficiency in SDN.
- Absence of a standard northbound interfaces have been implemented in a number of disparate forms. For example, the Floodlight controller includes a Java API and a *Representational State Transfer* (RESTful) API.

- **Core Functionalities of Controller:**

1. **End-user device discovery :** Discovery of end-user devices such as laptops, desktops, printers, mobile devices, and so on.
2. **Network device discovery:** Discovery of network devices that comprise the infrastructure of the network, such as switches, routers, and wireless access points.
3. **Network device topology management:** Maintain information about the interconnection details of network devices to each other and to the end-user devices to which they are directly attached.
4. **Flow management:** Maintain a database of the flows being managed by the controller and perform all necessary coordination with the devices to ensure synchronization of the device flow entries with that database.

3. SDN Applications/ SDN Application Plane :

- Built on top of the controller.
- Are part of network layers two and three & should not be confused with application layer of OSI.
- Flows are of 2 types: Proactive & reactive.
- SDN applications Interfaces with the controller, using it to set *proactive* flows on the devices and to receive packets that have been forwarded to the controller.
- **Reactive** mode reacts to traffic, consults the OpenFlow controller and creates a rule in the **flow-**table based on the instruction.
- In Proactive mode-Rather than reacting to a packet, an OpenFlow controller could populate the flow tables ahead of time for all traffic matches that could come into the switch.
- SDN applications are ultimately responsible for managing the flow entries that are programmed on the network devices using the controller’s API to manage flows.

3. SDN Applications/ SDN Application Plane :

- Programs of Application plane connect their requirements and response of network by relating with the SDN controller through the Northbound Interface (NBI)
- Therefore, controller can modify the network resources activities automatically
- The global abstract network view of the network resources are accessed by the programs in the application plane, for their purposes of internal decision-making, given by the controller by using models of data displayed via the Northbound Interface.

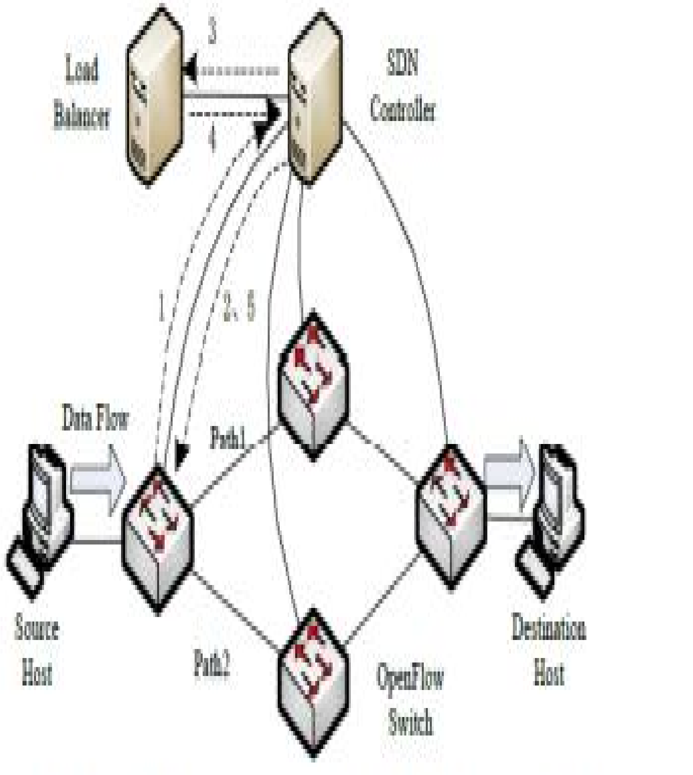
Load Balance System Design:

Building Blocks:

- The switch of SDN, the controller of SDN, interfaces of southbound and northbound and SDN applications are the fundamental building blocks of the SDN architecture.

1. Switch:

- The talented way for data centers and infrastructure of virtualized network is software switches.
- Examples of software based OpenFlow switch implementations consists of Switch Light (Floodlight Controller|Big Switch Networks), ofsoftswitch13, Open vSwitch, Pica8, Pantou, and XorPlus (Open vSwitch).
- The quantity of ports of virtual access is bigger than ports of physical access on data centers.
- Open vSwitch which consisted in software switches are used to move network functions to the edge, as a result allowing network virtualization.



2. Controller:

- The SDN Controller is a rational object that obtains requirements and instructions from the application layer of SDN and transmits them to the components of networking.
- The controller is also in charge of taking out information about the network from the hardware devices and connecting it back to the applications of SDN.
- The flow entries can be added, updated, deleted by the controller while using the southband API.
- The control plane of SDN is represented by the controller of SDN
- The architecture of SDN does not require the internal design of a controller of SDN.
- It could be a set of identical procedures organized to share load or protect one another from failures.
- Components of controller can be executed on computer platforms. They may also execute on distributed resources such as virtual machines in data centers.
- It can say that the controller of SDN is assumed to have global scope. Besides, its components are assumed to share their state and information with the controller.
- Numerous components of controller may have joint write access to resources of network. In this situation, they must be constructed to control disjoint sets of resources or actions and synchronized with each other thus they never issue conflicting commands.

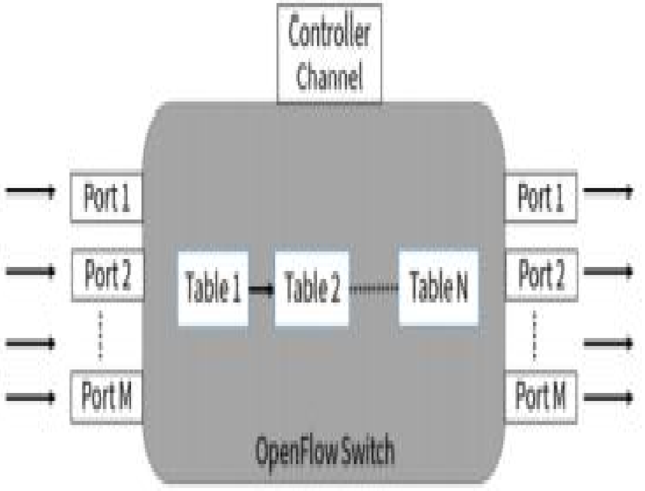
3. Southbound Interface:

- Southbound APIs assist effective control over the network and allow the controller of SDN to make changes based on the real time demands and requirements.
- OpenFlow is the first and perhaps most famous southbound interface. It is an industry standard that describes the way the controller of SDN should work together with the forwarding plane to control the network.
- Extensible Markup Language (XML) is used by NetConf to connect with the routers and switches to set up and construct changes

4. OpenFlow:

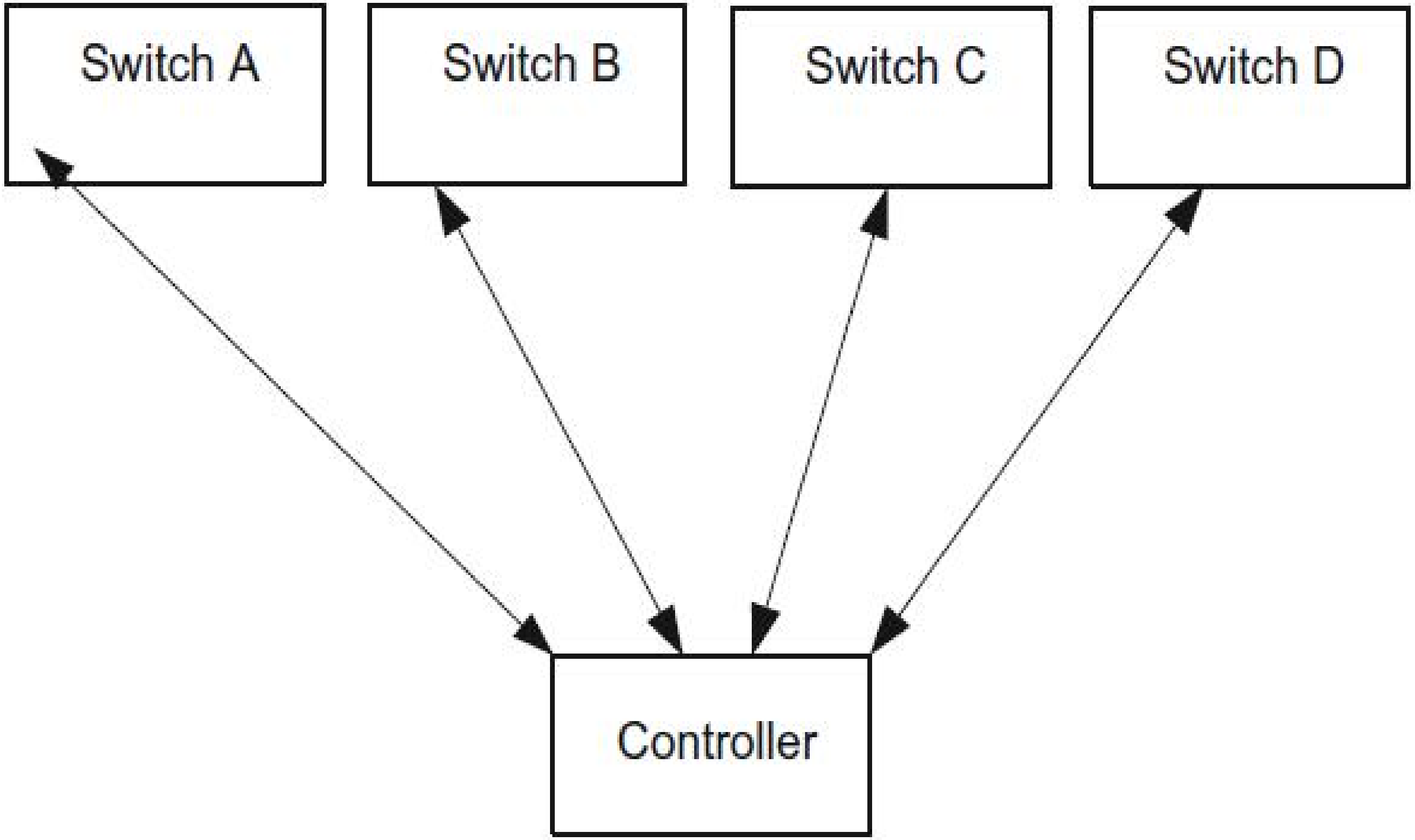
- The protocol of OpenFlow can be noticed as one potential implementation of controller-switch interactions.
- An abstraction for applications of business is provided to use facility which is given by the control layer
- Even though one of the basic concepts of SDN is to avoid vendor locking, dependence on one protocol does not serve this purpose.
- The OpenFlow switch is the elementary forwarding element, which is accessible through the protocol and interface of OpenFlow.

- Construction of Flow-based SDN such as OpenFlow might need extra forwarding table entries, buffer space, and statistical counters, although this structure, would appear to simplify the switching hardware at the first glance.
- An OpenFlow switch contains a flow table, which makes packet lookup and forwarding.
- Each flow table in the switch holds a set of flow entries that contain header fields or match fields and counters.



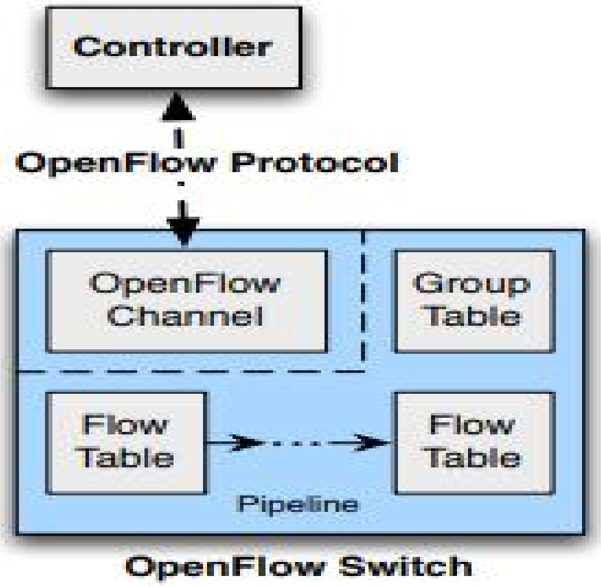
OpenFlow

- OpenFlow defines both the communications protocol between the SDN data plane and the SDN control plane and part of the behavior of the data plane.
- It does not describe the behavior of the controller itself.
- There is always an *OpenFlow controller* that communicates to one or more *OpenFlow switches*.

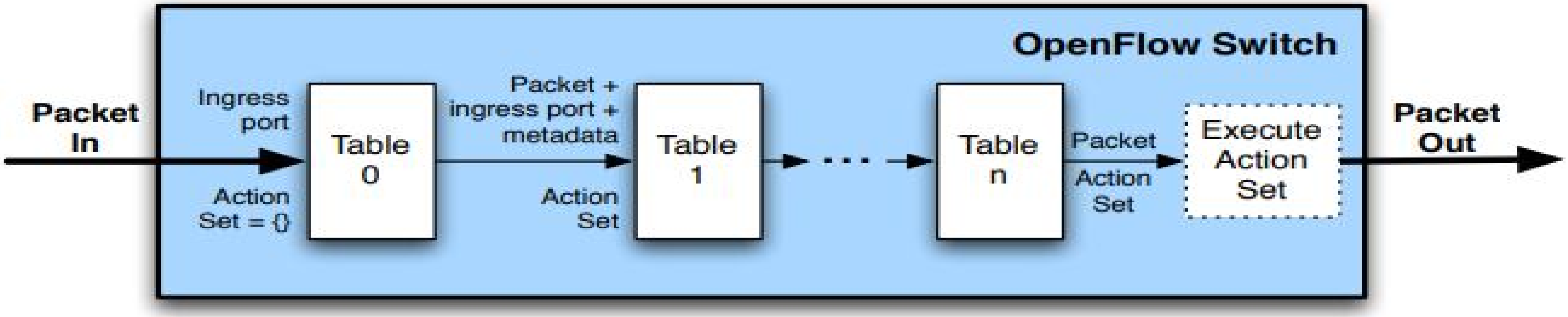

Switch A
Switch B
Switch C
Switch D
Controller

OpenFlow Switch

- The core function is to take packets that arrive on one port (path *X* on port 2 in the figure) and forward it through another port (port *N* in the figure), making any necessary packet modifications along the way.
- A unique aspect of the OpenFlow switch is embodied in the *packet-matching function* shown in Figure. The adjacent table is a *flow table*.
- The wide, gray, double arrow in figure starts in the decision logic, shows a match with a particular entry in that table, and directs the now-matched packet to an *action box* on the right.
- This action box has three fundamental options for the disposition of this arriving packet:
  - ***A*** Forward the packet out a local port, possibly modifying certain header fields first.
  - ***B*** Drop the packet.
  - ***C*** Pass the packet to the controller.



: Main components of an OpenFlow switch.



(a) Packets are matched against multiple tables in the pipeline

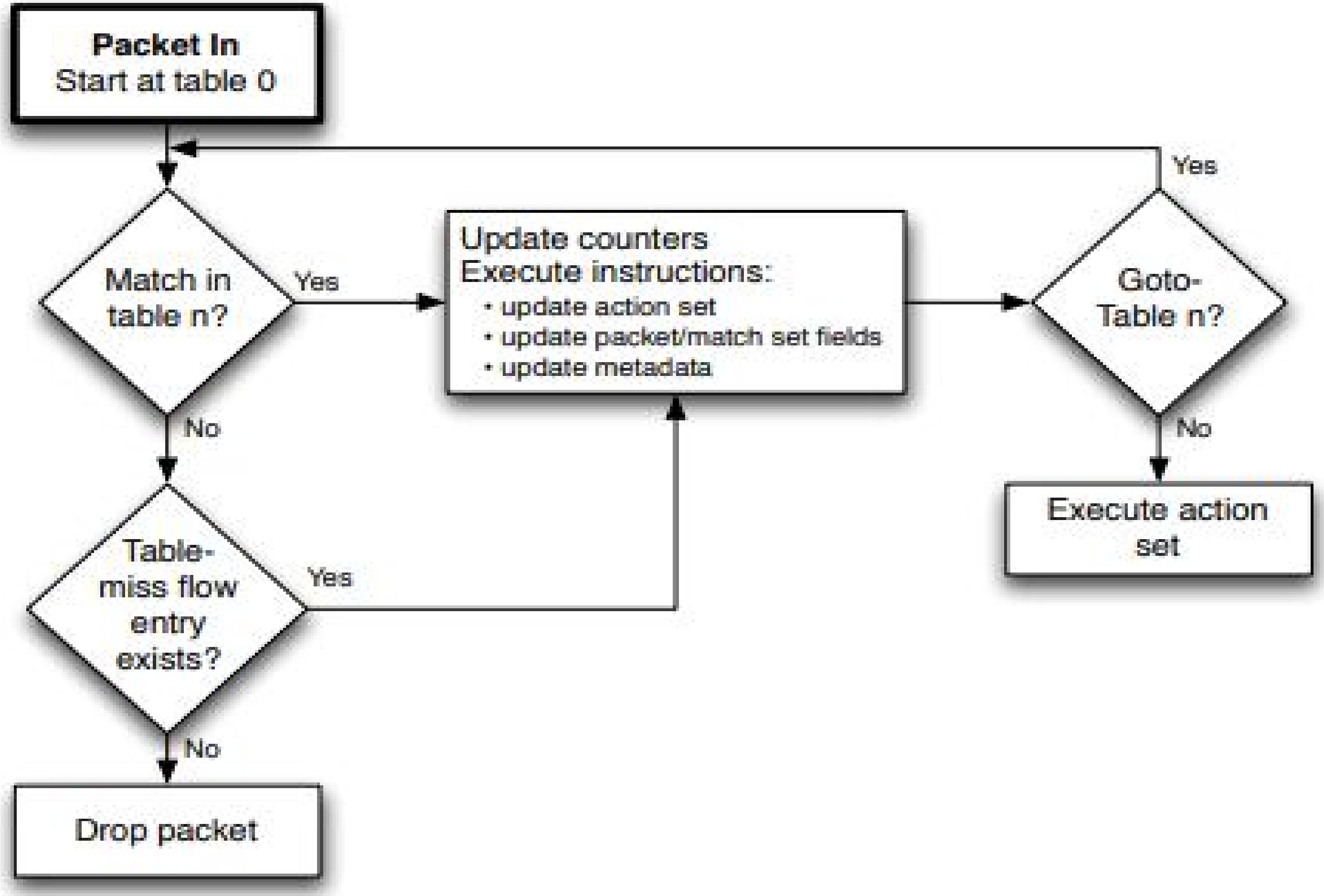
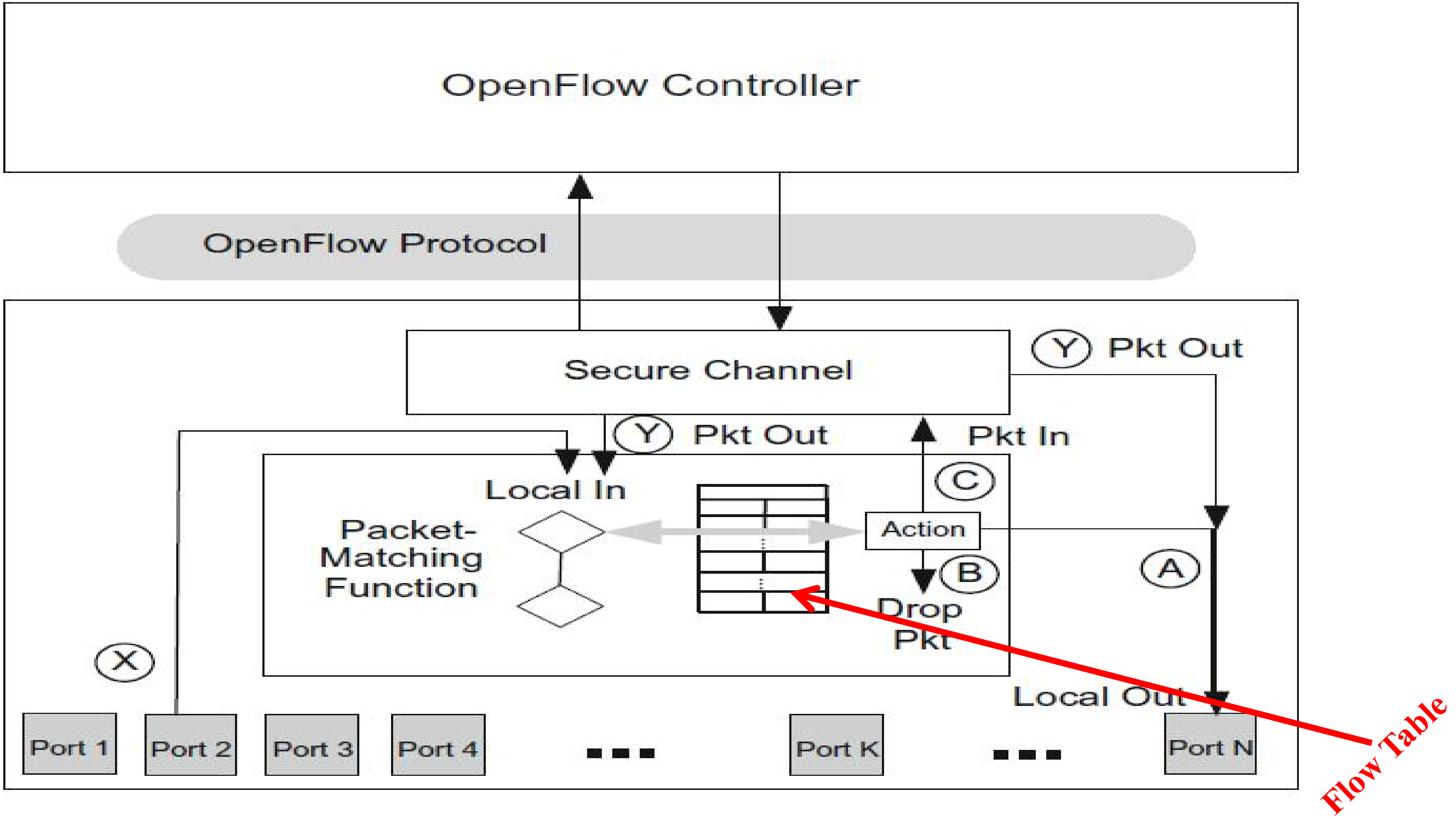


Figure 3: Flowchart detailing packet flow through an OpenFlow switch.


OpenFlow Controller
OpenFlow Protocol
Secure Channel
Y Pkt Out
Y Pkt Out
Pkt In
C
Local In
Packet-
Matching
Function
Action
B
Drop
Pkt
A
X
Local Out
Port 1
Port 2
Port 3
Port 4
■ ■ ■
Port K
■ ■ ■
Port N
Flow Table

- In the case of path *C*, the packet is passed to the controller over the *secure channel*.
- If controller has either a control message or a data packet to give to the switch, the controller uses this same secure channel in the reverse direction.
- When the controller has a data packet to forward out through the switch, it uses the **OpenFlow PACKET_OUT** message.
- data packet coming from the controller may take two different paths through the OpenFlow logic, both denoted *Y*.
- In the rightmost case, the controller directly specifies the output port and the packet is passed to that port *N*.
- In the leftmost path *Y* case, the controller indicates that it wants to defer the forwarding decision to the packet-matching logic.

- http://www.cs.yale.edu/homes/yu-minlan/teach/csci694b-spring14/syllabus.html
- SDN video lectures by nick feamster