Chapter 4

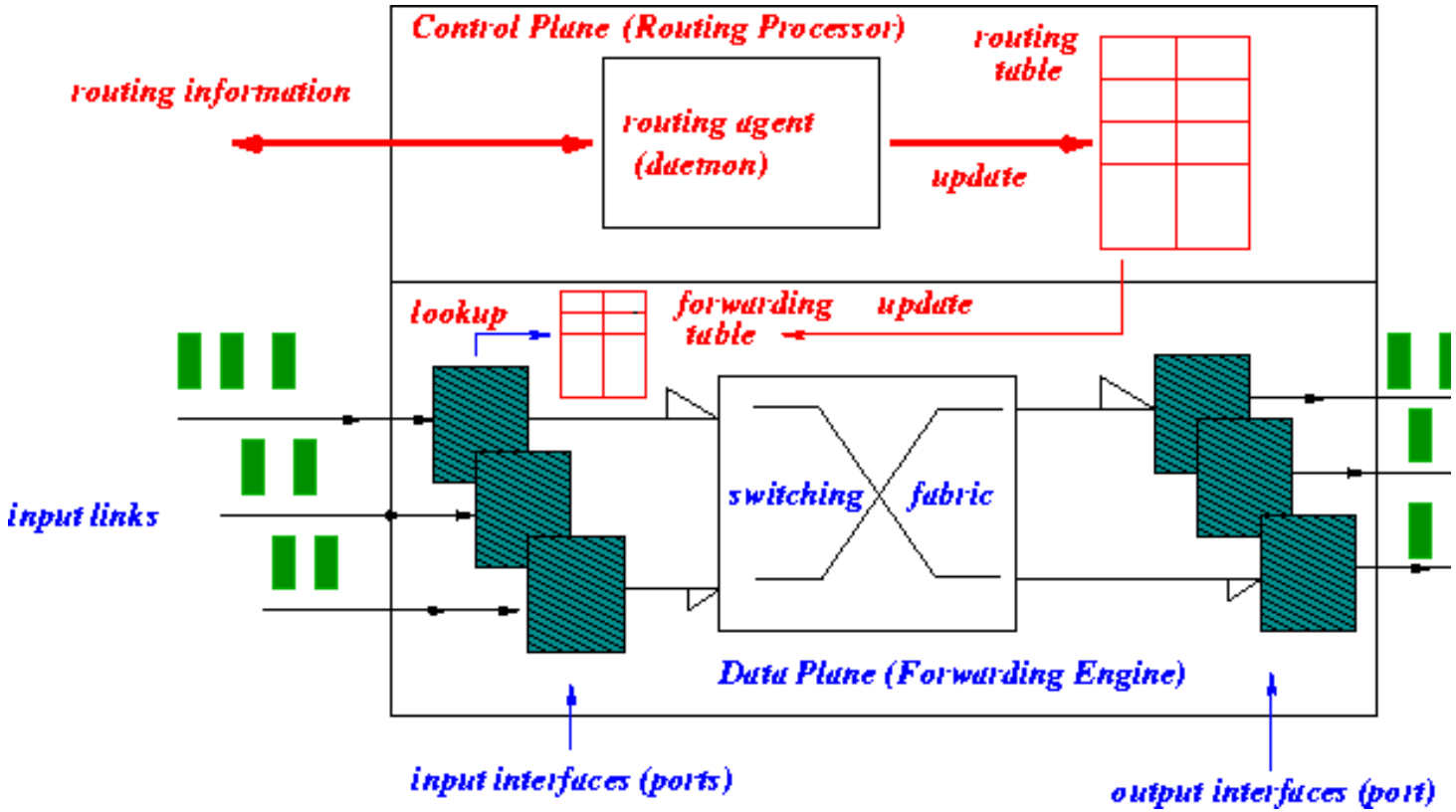
Software Defined Network (SDN)

Rami Tawil

What is Software-Defined Networking?

- “Software defined” becomes a very popular
  - Software defined networking,
  - Software defined storage,
  - Software defined radio, etc.
- What is it?
  - The control of the underlying system is exposed to the upper layer developer through an API.
  - System functionality is implemented over the API as an app (software).
  - This allows *customization* for what users want.
- Another word for “Software defined” is “Programmable”

Today's router: control plane and data plane

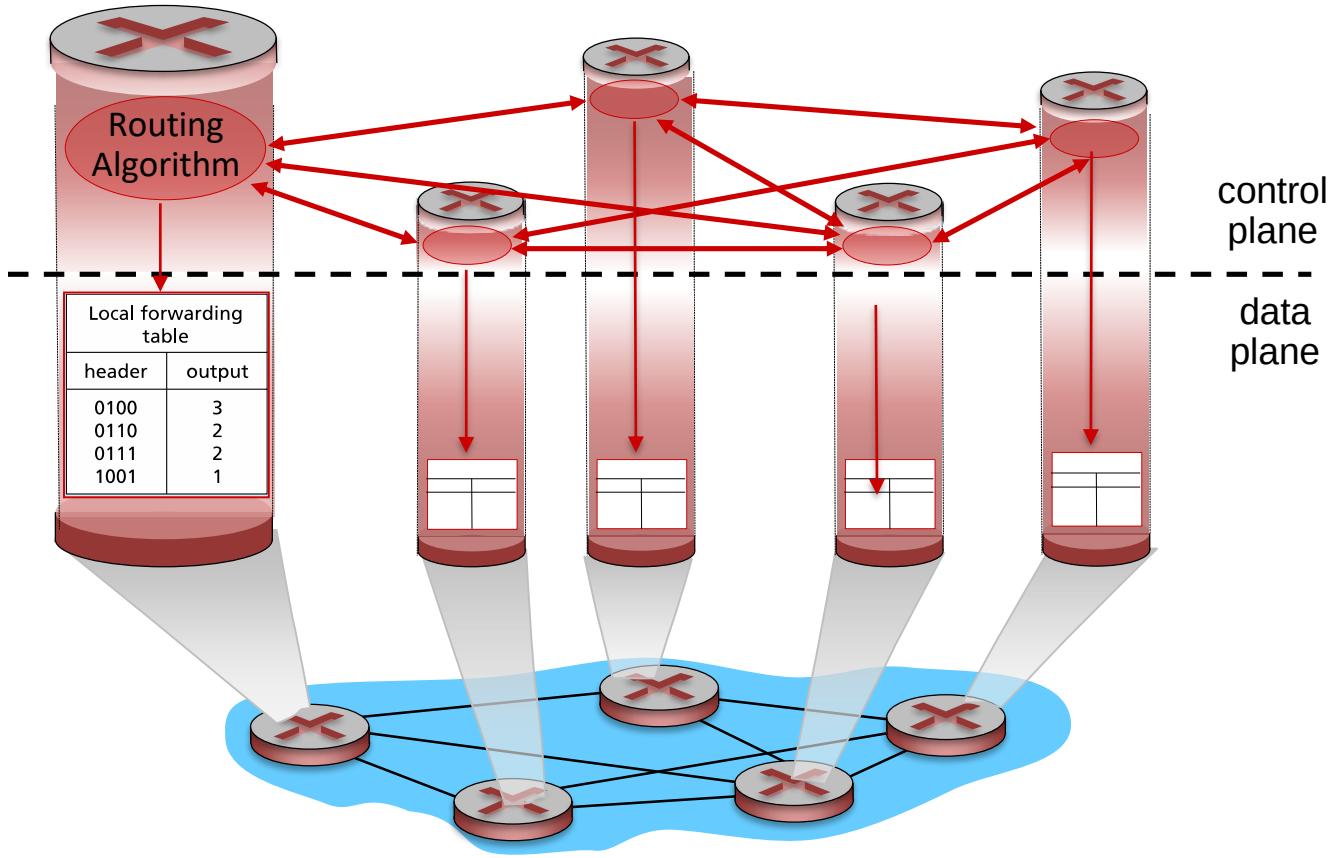


Today's router

- *Tightly coupled data and control plane*
- Hardware vendors also provide proprietary control software
- *monolithic* router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, IS-IS, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)

Today's network

Individual routing algorithm components in each and every router interact with each other in control plane to compute forwarding tables



Today's network

- Equipment vendors provide a set of routing (network control) choices: OSPF, RIP, ISIS, BGP, etc.
- Network control using a distributed, per router approach.
- Network administrator can change network parameters to achieve certain objectives: e.g. changing routes
  - Limited programmability
  - If one wants a new control mechanism (e.g. new routing services for data centers), (s)he is out of luck.
  - This is what SDN tries to overcome: making the network control like an API that user can develop by themselves.
  - SDN is to make network control more programmable, easier to deploy new services.

SDN motivation

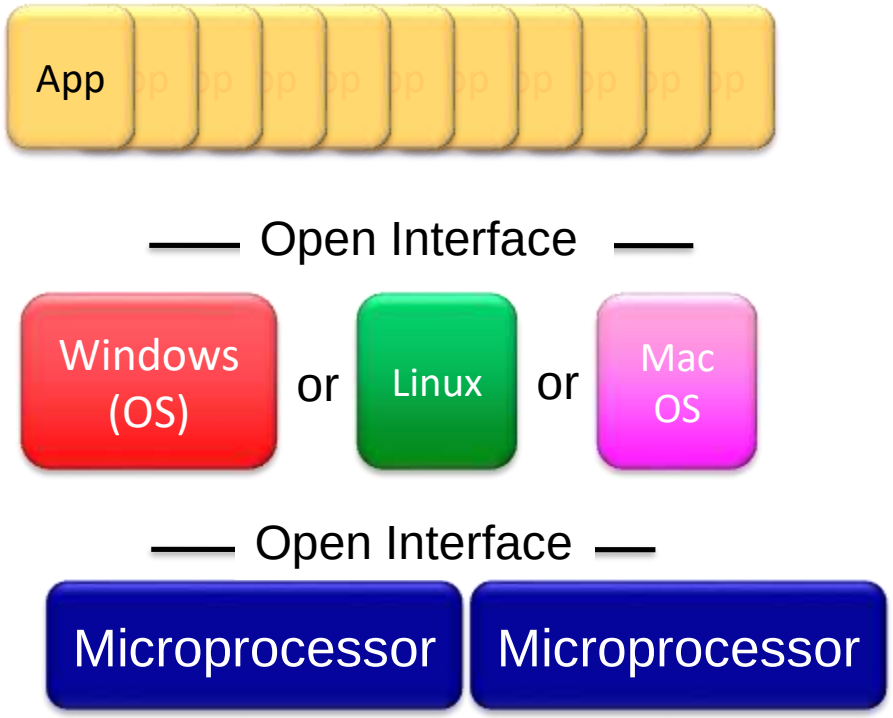
Why do we want to make network control more programmable?

- Short term:
  - Existing network control is no longer sufficient in several important areas, need innovation here!
    - ✓ Data centers, Wireless, network security
  - Existing network control is getting too complicated.
    - ✓ Too many middleboxes with their own control
    - ✓ Would be nice to provide a unified mechanism to deploy and manage these middleboxes
- Long term:
  - Programmability promotes innovation; and innovation is good for the networking industry.

Computing systems once upon a time

- Vertically integrated systems
  - Proprietary hardware
  - Proprietary OS
  - Proprietary applications
  - Highly reliable
- Dominated by a small number of large companies (IBM, HP, etc)
  - Slow software innovation
    - ✓ Proprietary development
  - Small industry

Computing Systems Today

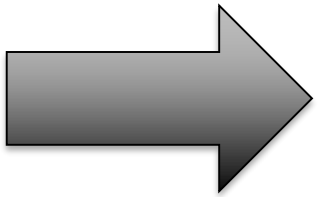


- Open interfaces
- Fast innovation
  - Everyone can participate
- Industry
  - Software is now part of everything.
- To promote innovation, we must make network systems like this!

Traditional vs. SDN Architecture

Traditional Network

- Distributed control plane
- Proprietary interfaces
- Static configuration
- Vendor lock-in
- Complex management



SDN Network

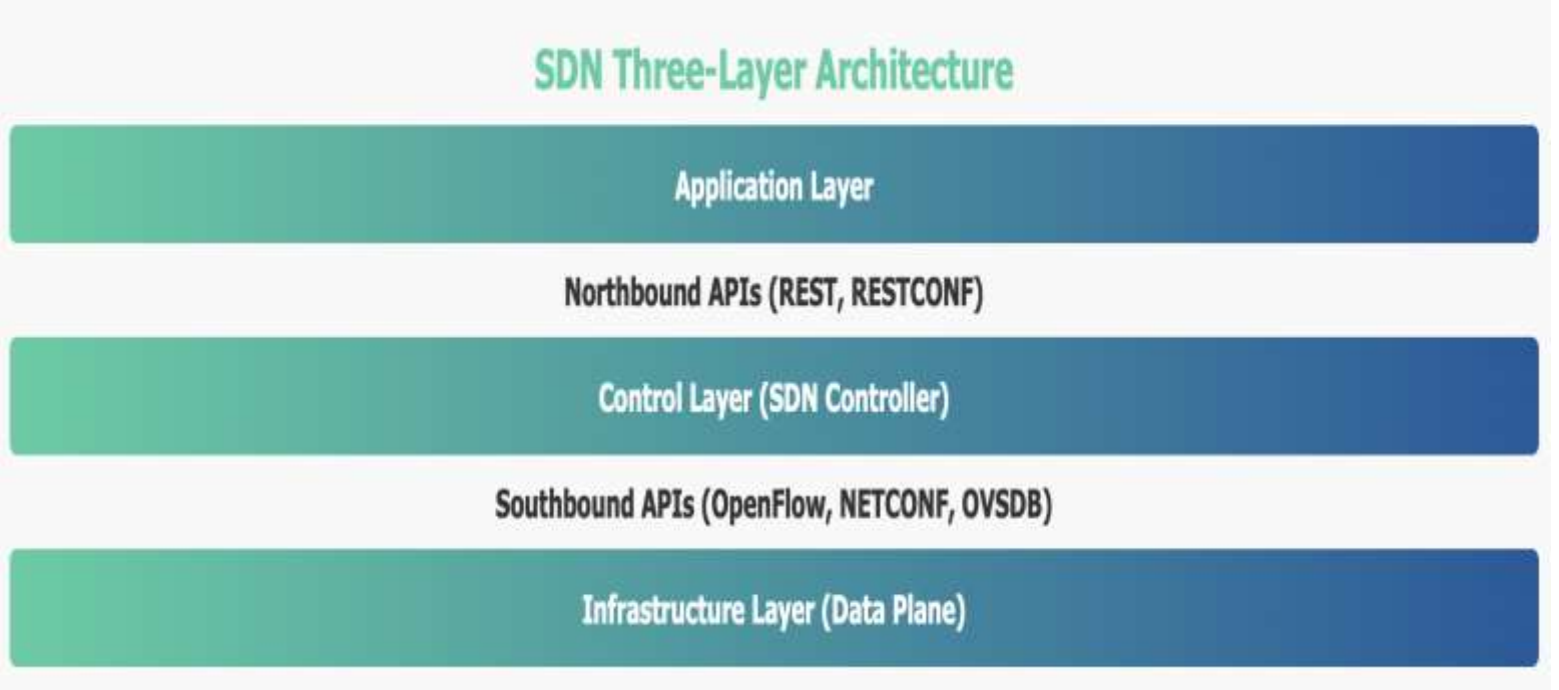
- Centralized control plane
- Open, standard interfaces
- Dynamic programmability
- Vendor neutrality
- Simplified management

SDN Key Principles

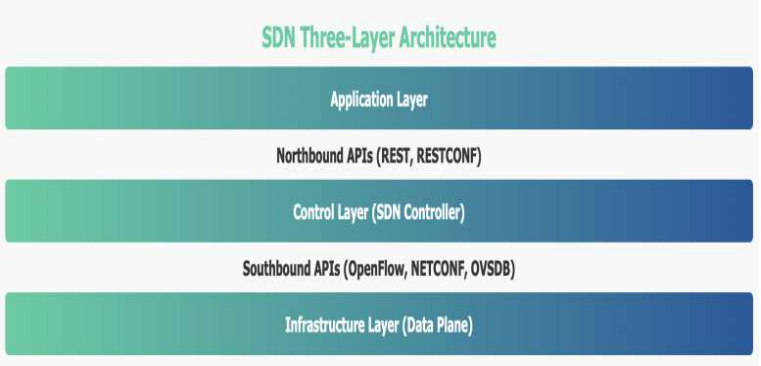
- **Centralized Control:** Global network view and decision making
- **Programmability:** Software-defined network behavior
- **Abstraction:** Applications independent of hardware details
- **Openness:** Vendor-neutral and standards-based

SDN Architecture

- **SDN Architectural Components**
  - Application Layer
  - Control Layer
  - Infrastructure Layer
  - Southbound and Northbound APIs



SDN Architecture



- **Layer Responsibilities**
  - **Application Layer:** Business logic, network services, applications
  - **Control Layer:** Network intelligence, policy enforcement, orchestration
  - **Infrastructure Layer:** Packet forwarding, switching, routing hardware
- **API Functions**
  - **Northbound APIs:** Controller-to-application communication
  - **Southbound APIs:** Controller-to-infrastructure communication
  - **Eastbound/Westbound:** Inter-controller communication

Control Plane vs Data Plane

- **Plane Separation Concept**

  SDN fundamentally separates the ***control*** plane (network intelligence) from the ***data*** plane (packet forwarding), enabling centralized control and programmable network behavior.

- **Plane Comparison**

Aspect	Control Plane	Data Plane
Function	Network intelligence & decisions	Packet forwarding & processing
Location	Centralized controller	Distributed switches/routers
Processing	Control messages, topology	User data packets
Performance	Scalability focused	Throughput & latency focused
Programmability	Highly programmable	Hardware-optimized

Control Plane vs Data Plane

- **Benefits of Separation**

Control Plane Benefits:

- Global network view
- Centralized policy enforcement
- Simplified network management
- Rapid service deployment
- Vendor independence

Data Plane Benefits:

- Hardware-optimized forwarding
- High throughput performance
- Low latency packet processing
- Simplified switch design
- Cost-effective infrastructure

SDN Controllers

- **Controller Functions**
  - **Network View:** Maintain global topology and state
  - **Policy Enforcement:** Implement network policies
  - **Flow Management:** Install and manage flow rules
  - **Application Interface:** Provide northbound APIs
- **Popular SDN Controllers**

Open Source Controllers

- **OpenDaylight:** Java-based, modular
- **ONOS:** Carrier-grade, high availability
- **Floodlight:** Simple, REST APIs
- **Ryu:** Python-based, component architecture

Commercial Controllers

- **Cisco APIC:** Application Policy Infrastructure
- **VMware NSX:** Network virtualization
- **Juniper Contrail:** Virtual networking
- **Big Switch Big Cloud Fabric:** Data center

SDN Controllers

- **Controller Architecture Patterns**
  - **Centralized:** Single controller instance
  - **Hierarchical:** Master-slave controller relationships
  - **Distributed:** Multiple coordinated controllers
  - **Federated:** Inter-domain controller cooperation

Northbound APIs

- **Northbound API Purpose**

Northbound APIs provide programmable interfaces between SDN controllers and applications, enabling network services and business logic implementation.

- **Common Northbound Interfaces**

REST APIs:
- HTTP-based CRUD operations
- JSON/XML data exchange
- Stateless communication
- Easy integration with web applications

Intent-Based APIs:
- High-level policy specification
- Business logic abstraction
- Declarative network configuration
- Automated policy translation

Network Programming Languages:
- Frenetic: Functional reactive programming
- Pyretic: Python-based DSL
- NetKAT: Network Kleene Algebra with Tests
- P4: Programming Protocol-Independent Packets

Northbound APIs

Intent-Based Networking

- High-level policy expression
- Automated implementation
- Business-driven networking
- Simplified configuration

API Examples

- Network topology discovery
- Flow rule installation
- Traffic engineering
- Security policy enforcement
- Quality of service management

SDN Benefits

Centralized Control

- Global network view and intelligence
- Simplified network management
- Consistent policy enforcement
- Reduced configuration complexity

Agility and Innovation

- Rapid service deployment
- Network programmability
- Faster time-to-market
- Dynamic resource allocation

Cost Reduction

- Commodity hardware utilization
- Reduced operational expenses
- Efficient resource utilization
- Simplified network architecture

Flexibility

- Vendor-independent solutions
- Open standards adoption
- Multi-vendor interoperability
- Future-proof architecture

Business Impact

- **Innovation:** Enables new network services and business models
- **Scalability:** Supports growth and changing requirements
- **Automation:** Reduces manual tasks and human errors

SDN Implementation Challenges

- **Technical Challenges**
  - **Controller Scalability:** Performance bottlenecks and scaling limits
  - **Network Latency:** Controller communication overhead
  - **Reliability:** Single point of failure concerns
  - **Security:** Centralized attack surface
- **Operational Challenges**

Migration Complexity:

- Legacy system integration
- Gradual transition planning
- Skill set transformation
- Process re-engineering
- Tool and platform changes

Performance Considerations:

- Controller placement optimization
- Flow setup latency
- Reactive vs. proactive approaches
- Network-wide consistency
- Fault tolerance mechanisms