Contents

Chapter 5: Network Layer: Control Plane (USE SLIDES WHEN READING)

Routing Protocols

Routing protocols are essential for determining the best paths for data packets to travel across a network. This section covers key concepts, including graph abstractions, algorithm classifications, and examples of routing protocols.

Routing Protocols: Goals

The goal of a routing protocol is to determine a "good" path from the source to the destination through a network of routers.

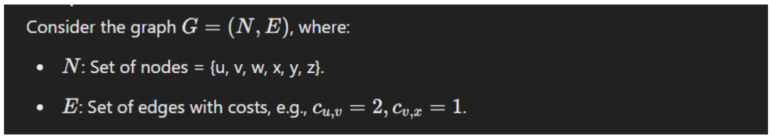
- **Path**: A sequence of routers traversed by a packet.
- **"Good" Path Criteria**:
  - **Least cost**: Minimal expense associated with the route.
  - **Fastest**: Low delay for packet delivery.
  - **Least congested**: Avoids heavily used links.

Graph Abstraction: Link Costs

Networks are modeled as graphs:

- **Nodes**: Represent routers.
- **Edges**: Represent physical or logical links between routers.
- **Cost**: A metric assigned to each link, which may reflect:
  - Bandwidth (e.g., cost inversely proportional to bandwidth).
  - Delay (e.g., cost directly proportional to delay).
  - Congestion (e.g., cost increases with higher utilization).

Example:



Consider the graph $G = (N, E)$, where:

- N: Set of nodes = {u, v, w, x, y, z}.
- E: Set of edges with costs, e.g., $c_{u,v} = 2, c_{v,x} = 1$.

Routing Algorithm Classification

1. Based on Information Scope

- **Global Algorithms**:
  - Every router has complete topology and link cost information.
  - Example: **Link-State Algorithms** (e.g., Dijkstra's algorithm).
- **Decentralized Algorithms**:
  - Routers know only the costs to their immediate neighbors.
  - Gradually share information to compute paths.
  - Example: **Distance-Vector Algorithms** (e.g., Bellman-Ford algorithm).

2. Based on Route Dynamics

- **Static Routing**:
  - Routes change slowly, often manually configured.

    - Suitable for stable, small networks.
- **Dynamic Routing**:
    - Routes change in response to network events (e.g., link failures, congestion).
    - Uses periodic updates or event-triggered recalculations.

Dijkstra's Link-State Algorithm

Dijkstra’s algorithm computes the **least-cost path** from a source node to all other nodes in the network.

- **Input**: Global knowledge of topology and link costs.
- **Output**: A forwarding table for the source node.

Key Notations

- $c_{x,y}$: Cost of the direct link between nodes x and y. If no direct link, $c_{x,y} = \infty$.
- $D(v)$: Current estimate of the least-cost path from the source u to node v.
- $p(v)$: Predecessor of v in the least-cost path.
- N': Set of nodes whose least-cost path is known.

Advantages and Limitations of Dijkstra's Algorithm

- **Advantages**:
    - Guarantees the optimal path.
    - Efficient for smaller networks.
- **Limitations**:
    - Requires global topology knowledge.
    - Computational complexity O(n^2), which can be reduced with priority queues.

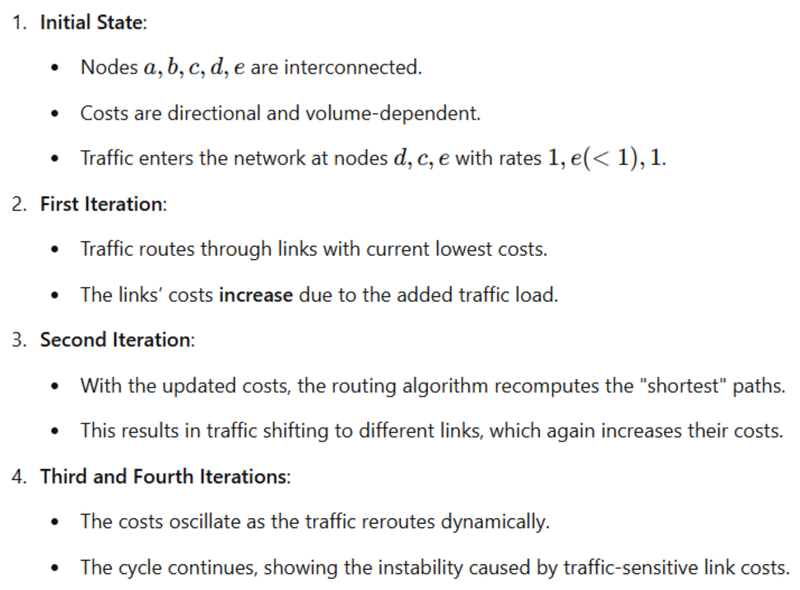
Dijkstra's Algorithm: Oscillations Possible

When link costs in a network depend on the **traffic volume**, the network can experience **route oscillations**. This happens because the link costs change dynamically as routing decisions are made, causing instability.

Why Do Oscillations Occur?

- **Dynamic Link Costs**: Link costs are no longer fixed. Instead, they depend on the traffic load.
- **Feedback Loop**:
    1. Traffic takes the route with the current **lowest cost**.
    2. As traffic increases, the cost of that link rises (due to congestion).
    3. Routing algorithms then choose a **new route** with the now-lower cost.
    4. The cycle repeats, leading to oscillations.

Sample Scenario Explanation (Illustrated in the Slide)



1. **Initial State**:
    - Nodes a, b, c, d, e are interconnected.
    - Costs are directional and volume-dependent.
    - Traffic enters the network at nodes d, c, e with rates $1, e(< 1), 1$.
2. **First Iteration**:
    - Traffic routes through links with current lowest costs.
    - The links' costs **increase** due to the added traffic load.
3. **Second Iteration**:
    - With the updated costs, the routing algorithm recomputes the "shortest" paths.
    - This results in traffic shifting to different links, which again increases their costs.
4. **Third and Fourth Iterations**:
    - The costs oscillate as the traffic reroutes dynamically.
    - The cycle continues, showing the instability caused by traffic-sensitive link costs.

Analogy:

- Imagine two highways leading to the same destination. Initially, drivers choose the less congested highway. However, as more drivers use it, congestion increases, and the other highway becomes the better choice. This behavior keeps repeating as drivers keep switching routes, leading to **traffic oscillations**.

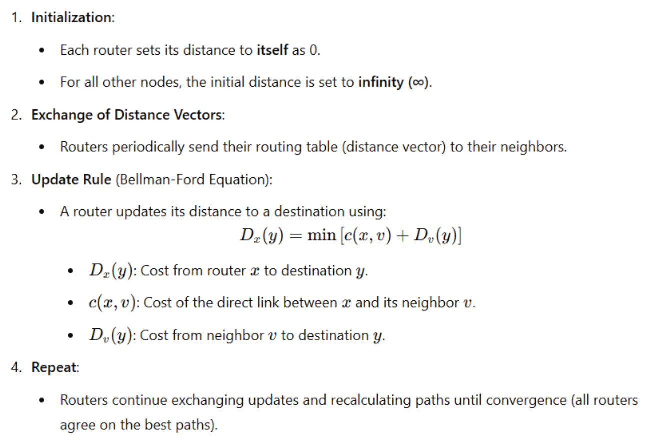
Distance Vector Algorithm

The **Distance Vector (DV) Algorithm** is a dynamic routing algorithm that enables routers to determine the best path to each destination by exchanging information with their **directly connected neighbors**. It is also known as a **decentralized algorithm** because each router operates based on **local knowledge**.

Key Features of the Distance Vector Algorithm

1. **Router Knowledge**:
    - Each router knows only:
        - The cost to its **directly connected neighbors**.
        - Its own forwarding table, which contains:
            - Destination network
            - Distance (cost) to the destination
            - Next-hop router
2. **Exchange of Information**:
    - Routers periodically exchange their **distance vectors** (a list of distances to all destinations) with their neighbors.
    - Each router updates its own routing table based on the information received.
3. **Iterative and Distributed**:
    - The algorithm runs iteratively.
    - Routers only share updates with their **immediate neighbors**.

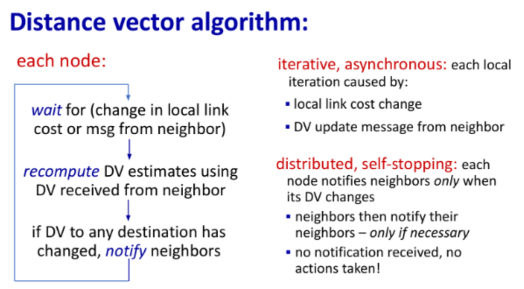
Distance Vector Algorithm Steps



1. **Initialization**:
    - Each router sets its distance to **itself** as 0.
    - For all other nodes, the initial distance is set to **infinity** (∞).
2. **Exchange of Distance Vectors**:
    - Routers periodically send their routing table (distance vector) to their neighbors.
3. **Update Rule** (Bellman-Ford Equation):
    - A router updates its distance to a destination using:

$$D_x(y) = \min\left[c(x,v) + D_v(y)\right]$$

        - $D_x(y)$: Cost from router x to destination y.
        - $c(x,v)$: Cost of the direct link between x and its neighbor v.
        - $D_v(y)$: Cost from neighbor v to destination y.
4. **Repeat**:
    - Routers continue exchanging updates and recalculating paths until convergence (all routers agree on the best paths).



Distance vector algorithm:

each node:

wait for (change in local link cost or msg from neighbor)

recompute DV estimates using DV received from neighbor

if DV to any destination has changed, *notify* neighbors

iterative, asynchronous: each local iteration caused by:
- local link cost change
- DV update message from neighbor

distributed, self-stopping: each node notifies neighbors *only* when its DV changes
- neighbors then notify their neighbors – *only if necessary*
- no notification received, no actions taken!

Comparison: Link State (LS) vs Distance Vector (DV)

Feature	Link State (LS)	Distance Vector (DV)
Knowledge Scope	Global topology knowledge	Only neighbor information
Information Exchanged	Entire network topology	Distance vectors (next-hop costs)
Algorithm Type	Dijkstra's Algorithm	Bellman-Ford Algorithm
Convergence Speed	Faster convergence	Slower; prone to loops (count-to-infinity problem)
Message Overhead	Higher (flooding link-state packets)	Lower (only neighbor updates)

Complexity	Higher computational complexity	Simpler computations
Scalability	Better for large networks	Suitable for smaller networks
Failure Handling	Detects failures quickly	May take longer to recover

---

Making Routing Scalable

Routing must scale efficiently to accommodate the ever-growing size of the Internet. To achieve this, networks use **hierarchical routing**, breaking large networks into manageable pieces.

---

1. Autonomous Systems (ASes)

- The Internet is divided into smaller administrative regions called **Autonomous Systems (ASes)**.
- Each AS is managed independently and has its own **intra-AS** and **inter-AS** routing protocols.

Type	Scope	Examples
Intra-AS Routing	Within an AS	RIP, OSPF, IS-IS, EIGRP
Inter-AS Routing	Between different ASes	BGP (Border Gateway Protocol)

---

2. Intra-AS Routing: Routing Within an AS

Intra-AS routing protocols are used to find paths within a single AS. The most common protocols are:

Routing Information Protocol (RIP):

- **Type**: Distance-vector protocol.
- **Details**:
  - Routers exchange **distance vectors** (routing tables) every 30 seconds.
  - Simple but inefficient for large networks.
- **Status**: RIP is no longer widely used due to scalability issues.

Enhanced Interior Gateway Routing Protocol (EIGRP):

- **Type**: Distance-vector protocol with advanced features.
- **Details**:

        - Developed by Cisco (formerly proprietary, now standardized in RFC 7868).
        - Includes improvements like faster convergence and more flexible metric calculation.

Open Shortest Path First (OSPF):

- **Type**: Link-state routing protocol.
- **Details**:
    - **Open Standard**: Available publicly, unlike EIGRP.
    - **Operation**:
        - Each router floods **link-state advertisements (LSAs)** to all routers in the AS.
        - Routers use Dijkstra's algorithm to compute the least-cost path.
    - **Metrics**: Can account for bandwidth, delay, or other link costs.
    - **Security**: OSPF messages are **authenticated** to prevent malicious routing updates.

Analogy:
Think of OSPF like a map-sharing system where every router (node) shares its local roads with all other routers. Each router then calculates the shortest path to each destination using the full map.

---

3. Hierarchical OSPF

As networks grow, OSPF introduces a **hierarchical structure** to improve scalability and efficiency.

Key Features of Hierarchical OSPF:

1. **Two-Level Hierarchy**:
    - **Local Area (Area 1, 2, etc.)**: Routers within an area only exchange local routing information.
    - **Backbone Area (Area 0)**: Connects all local areas together.
2. **Area Border Routers (ABRs)**:
    - Summarize routing information from local areas.
    - Advertise only summarized routes to the backbone, reducing overhead.
3. **Backbone Routers**:
    - Operate within the backbone (Area 0).
    - Use summarized routes to connect to other areas or Autonomous Systems.
4. **Boundary Routers**:
    - Connect the OSPF network to **external networks or other ASes**.

How Hierarchical OSPF Works

1. **Within an Area**:
   - Routers compute routes using **link-state advertisements (LSAs)**.
   - Full routing topology is exchanged only **within the area**.
2. **Between Areas**:
   - Area Border Routers (ABRs) summarize routes to reduce information passed to other areas.
3. **Backbone (Area 0)**:
   - Acts as the central hub that interconnects all areas.
   - ABRs advertise summarized routes from local areas to the backbone.

Example of Hierarchical OSPF

- **Area 1**:
  - Local routers flood LSAs only within Area 1.
  - ABR summarizes routes to the backbone.
- **Backbone**:
  - Routers in Area 0 receive summarized routes from all areas.
  - They advertise these routes to other areas.
- **Area 2**:
  - Routers compute local routes and rely on the backbone for inter-area communication.

Benefits of Hierarchical OSPF:

1. Reduces **routing table size** by summarizing routes.
2. Limits **LSA flooding** to within areas, improving efficiency.
3. Scales better for large networks.

Comparison of RIP and OSPF

Feature	RIP	OSPF
Routing Type	Distance Vector	Link State
Metric	Hop Count	Link cost (bandwidth, delay)
Update Method	Periodic (every 30 sec)	Event-driven (LSAs on change)
Convergence Speed	Slow	Fast
Scalability	Poor (limited to small networks)	Good (supports large networks)
Security	None	Supports authentication

Key Takeaways

1. **Intra-AS Routing**:
   - RIP, OSPF, and EIGRP are common protocols for routing within an AS.
   - OSPF is a **link-state protocol** that scales well for large networks.
2. **Hierarchical OSPF**:
   - Introduces a two-level hierarchy with **local areas** and a central **backbone**.
   - Reduces overhead by summarizing routes and limiting flooding.
3. **Comparison**:
   - OSPF is faster, more scalable, and secure compared to RIP.

Route Aggregation

Route aggregation (also called **route summarization**) reduces the number of routing table entries by combining multiple smaller subnets into a single, larger route.

How Route Aggregation Works

- A group of networks with contiguous IP addresses can be represented by a **single prefix**.
- Routers advertise the single **aggregated prefix** instead of multiple individual routes.

Example of Route Aggregation

Without aggregation:

- Network 1: 200.1.1.0/24200.1.1.0/24200.1.1.0/24
- Network 2: 200.1.2.0/24200.1.2.0/24200.1.2.0/24
- Network 3: 200.1.3.0/24200.1.3.0/24200.1.3.0/24

With aggregation:

- All three networks can be represented by the aggregated prefix **200.1.0.0/22200.1.0.0/22200.1.0.0/22**.

Benefit:

- Reduces routing table size.
- Simplifies routing updates.

Advantages of Route Aggregation

1. **Reduces Routing Table Entries**:
   - Helps routers store fewer entries, saving memory and processing power.
2. **Improves Network Performance**:
   - Smaller routing tables speed up lookup processes.
3. **Simplifies Routing Updates**:
   - Aggregated routes mean fewer updates are sent when changes occur.

Comparison: Flat vs Hierarchical Routing

Feature	Flat Routing	Hierarchical Routing
Routing Table Size	Very large for large networks	Smaller due to route aggregation
Complexity	High (routers need full knowledge)	Lower (routers focus on local areas)
Scalability	Not scalable for large networks	Highly scalable
Example	RIP in a small LAN	OSPF for Intra-AS, BGP for Inter-AS

Why Routing Needs to Scale

1. **Growth of the Internet**:
   - With billions of connected devices, routing tables cannot store individual entries for all networks.
2. **Reduced Overhead**:
   - Smaller routing tables and fewer updates improve router performance and reduce delays.
3. **Efficient Management**:
   - Hierarchical routing allows organizations and ISPs to manage their own networks independently while connecting to the global Internet.

---

eBGP and iBGP Connections

1. eBGP (External BGP):

- **Purpose**: Exchange routing information **between different ASes** (Autonomous Systems).
- **Connectivity**: Shown in the slide as **red dashed lines**.

2. iBGP (Internal BGP):

- **Purpose**: Exchange routing information **within the same AS**.
- **Connectivity**: Shown in the slide as **blue lines**.

Key Point:

- Gateway routers run both **eBGP** (to communicate externally) and **iBGP** (to distribute routes internally).

BGP Basics

- **BGP Session**: Two BGP routers (peers) establish a **TCP connection** and exchange routing information.
- **BGP Protocol**: Works as a **path-vector protocol**, where routers advertise **paths** to reach destination prefixes.

Example:

- AS3 gateway **3a** advertises path **AS3, X** to AS2 gateway **2c**.
  - AS3 promises to **forward traffic** towards destination XXX.

Key Points:

- BGP exchanges **paths** (not just distances like RIP or OSPF).
- BGP ensures inter-AS routing follows policies and avoids routing loops.

BGP Protocol Messages

BGP messages are exchanged over a **TCP connection** between peers.

BGP Message Types:

1. **OPEN**:
   - Establishes a TCP connection and authenticates the peer.
2. **UPDATE**:
   - Advertises new routes (or withdraws old routes).
3. **KEEPALIVE**:
   - Keeps the connection active in the absence of updates.
4. **NOTIFICATION**:
   - Reports errors in the connection; also used to **close** connections.

Path Attributes and BGP Routes

- **BGP Advertised Route**: A route includes:
    1. **Prefix**: The destination network being advertised.
    2. **Attributes**: Additional information about the route.

Important Attributes:

1. **AS-PATH**:
    - Lists all the ASes through which the route advertisement has passed.
2. **NEXT-HOP**:
    - Indicates the IP address of the **next-hop router** to reach the destination.

Policy-Based Routing

- **Import Policies**: Gateways may **accept or decline** routes based on policies.
    - Example: Avoid routing traffic through a specific AS for security or economic reasons.
- **Export Policies**: Gateways decide **whether to advertise** certain routes to neighbors.

Key Point:

- BGP allows network administrators to enforce policies, such as avoiding transit traffic through specific ASes.

BGP Path Advertisement

Process:

1. AS2 gateway router **2c** receives path **AS3, X** (via eBGP) from AS3 gateway **3a**.
2. AS2 router **2c**:
    - Accepts the path **AS3, X**.
    - Propagates the path within AS2 using **iBGP**.
3. AS2 router **2a** advertises path **AS2, AS3, X** to AS1 gateway **1c** via **eBGP**.

BGP Path Advertisement: Multiple Paths

- Gateway routers may learn **multiple paths** to the same destination.

Example:

1. AS1 gateway router **1c** learns paths:
   - **AS2, AS3, X** from **2a**.
   - **AS3, X** directly from **3a**.
2. Based on policies, router **1c** chooses the best path (e.g., shortest AS-PATH).

Key Concept:

- **Multiple Paths**: Routers select the path based on attributes like **AS-PATH length**, NEXT-HOP costs, or policies.

BGP: Populating Forwarding Tables

Process:

1. Routers learn BGP paths using **iBGP** and **eBGP**.
2. For a destination XXX:
   - Path XXX is advertised internally via iBGP.
   - OSPF or another intra-AS protocol determines the **local route** to the next hop.

Example:

- At router **1d**:
  - To reach destination XXX through **1c**:
    - Use **interface 1** (OSPF routing).

Hot Potato Routing

- **Definition**: Hot potato routing chooses the **closest local gateway** to send traffic out of the AS.

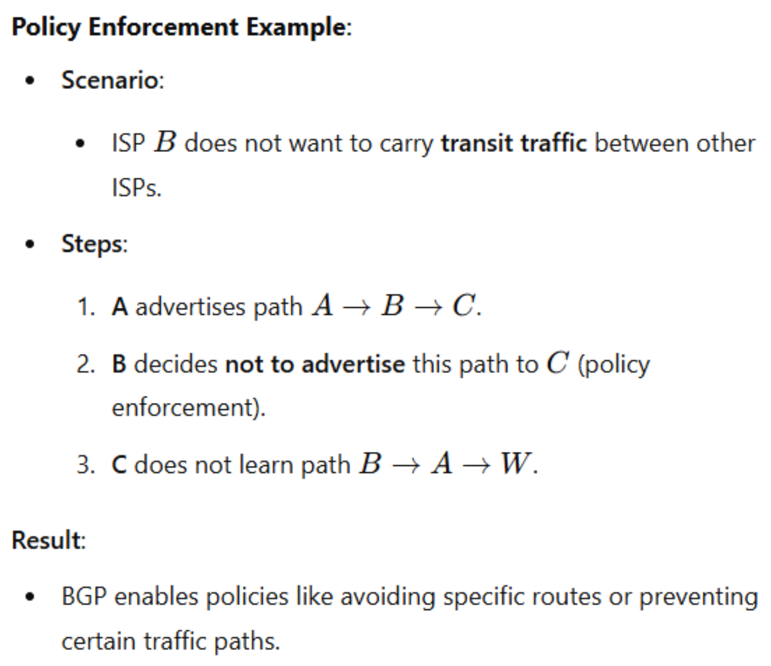
Process:

- **Router 2d** (AS2) has two options to reach destination XXX:
  1. Via gateway **2a**.
  2. Via gateway **2c**.
- Hot potato routing chooses the **gateway with the least intra-AS cost**.

Key Point:

- **Inter-domain costs** (e.g., cost across ASes) are **not considered** in hot potato routing.

BGP: Achieving Policy via Advertisements



Policy Enforcement Example:

- **Scenario**:
  - ISP B does not want to carry **transit traffic** between other ISPs.
- **Steps**:
  1. **A** advertises path $A \rightarrow B \rightarrow C$.
  2. **B** decides **not to advertise** this path to C (policy enforcement).
  3. **C** does not learn path $B \rightarrow A \rightarrow W$.

Result:

- BGP enables policies like avoiding specific routes or preventing certain traffic paths.

- .

BGP Route Selection

When multiple routes to the same destination exist, BGP selects the best route based on the following criteria:

1. **Local Preference**: Policy-based decision set by the network administrator.
2. **Shortest AS-PATH**: Path with the fewest AS hops.
3. **Closest NEXT-HOP Router**: Hot potato routing.
4. **Additional Criteria**: Other attributes like route age or tie-breaking rules.

Why Different Intra-AS and Inter-AS Routing?

1. **Policy**:
   - **Inter-AS**: Administrators want control over how traffic is routed through their network.
   - **Intra-AS**: Less concern about policy since the AS has a single admin.
2. **Scale**:
   - **Inter-AS**: Hierarchical routing reduces table size and update traffic.
   - **Intra-AS**: Routing can remain flat as it is manageable within an AS.
3. **Performance**:

    - **Intra-AS**: Focus on performance and fast routing.
    - **Inter-AS**: Policy often dominates over performance.

Software Defined Networking (SDN)

---

1. Traditional Per-Router Control Plane

- In traditional networks:
    - Each router has its own **control plane** that computes **forwarding tables** individually.
    - Control logic runs in a **distributed manner**, making it harder to manage the network centrally.
- **Key Challenge**:
    - Misconfigurations are common.
    - Difficult to introduce new routing policies.

Example:

- Routers exchange routing updates with their immediate neighbors to build the forwarding table.

---

2. Introduction to Software-Defined Networking (SDN)

- **What is SDN?**
  SDN separates the **control plane** (decision-making) from the **data plane** (packet forwarding) in network devices.
- **Centralized Control**:
  A single, logically centralized **SDN controller** computes and installs forwarding rules for all routers.

---

3. Why a Logically Centralized Control Plane?

- **Easier Management**:
    - Avoid router misconfigurations.
    - Greater flexibility in controlling traffic flows.
- **Programming Routers**:
    - SDN uses a **table-based forwarding approach** (e.g., OpenFlow API) to program routers dynamically.
- **Centralized Programming**:
    - The controller computes tables and distributes them to all routers.

    - This avoids complexity compared to distributed control protocols.

Key Advantage:

- Faster innovation as SDN is **open** (non-proprietary).

---

4. SDN Analogy: Mainframe to PC Revolution

Traditional Networks	Software-Defined Networking
Vertically Integrated:	**Horizontal Open Interfaces**:
Closed, proprietary systems	Open interfaces allow flexibility
Slow innovation, small market	Rapid innovation and growth of industry

Analogy:

- Similar to the shift from closed mainframe computers to open, programmable PCs.

---

5. Traffic Engineering Challenges with Traditional Routing

Scenario 1:

- **Problem**: If the network operator wants to split **u-to-z** traffic across paths uvwzuvwzuvwz and uxyzuxyzuxyz for load balancing:
    - Traditional routing cannot support this as it relies on **destination-based forwarding**.

Scenario 2:

- **Problem**: Routing **blue** and **red traffic** differently from w→zw \to zw→z.
    - Cannot be done with traditional **LS (Link State)** or **DV (Distance Vector)** routing.

Solution:

- **Generalized Forwarding and SDN** (learned in Chapter 4) allow **any routing** to be implemented dynamically.

---

6. SDN: Separation of Control Plane and Data Plane

1. **Data Plane**:

    - Routers forward packets based on tables installed by the controller.
2. **Control Plane**:
    - The centralized **SDN controller** computes the routing decisions.

Key Features:

- SDN uses **OpenFlow** as a protocol to manage forwarding tables on routers.

---

7. SDN Controller

- The **SDN Controller** acts as the **network operating system**:
    - Maintains the **global state** of the network.
    - Communicates with:
        - **Northbound API**: For network control applications (e.g., routing, load balancing).
        - **Southbound API**: To control switches and routers (e.g., OpenFlow).

Advantages:

- Provides scalability, fault-tolerance, and easy interaction between control applications and devices.

---

8. Network-Control Applications

- **Network-Control Apps**:
    - Act as the "brains" of the network.
    - Implement control functions like:
        - Routing
        - Access control
        - Load balancing
- **Third-Party Flexibility**:
    - Applications can be provided by vendors or independent developers.

---

9. Components of SDN Controller

1. **Interface Layer**:
    - Abstract API to network control apps.
2. **State Management**:
    - Distributed database storing:
        - Link states

        - Switch states
        - Network topology
3. **Communication**:
    - Allows communication between SDN controller and switches.

---

10. OpenFlow Protocol

- **OpenFlow** operates between:
    - **Controller**: Centralized brain.
    - **Switches**: Forwarding devices.

Key Features:

- Uses **TCP** to exchange messages.
- **Three Classes of Messages**:
    1. **Controller-to-Switch**: Configures forwarding tables.
    2. **Asynchronous**: Updates from switches to the controller.
    3. **Symmetric**: Miscellaneous messages.

Purpose:

- Allows the SDN controller to specify **generalized forwarding actions**.

---

11. OpenFlow: Controller-to-Switch Messages

1. **Features**: Query switch capabilities.
2. **Configure**: Controller sets switch configurations.
3. **Modify-State**: Add, delete, or modify flow table entries.
4. **Packet-Out**: Controller sends a packet out of a specific port.

---

12. OpenFlow: Switch-to-Controller Messages

1. **Packet-In**: Transfers a packet and its control information to the controller.
2. **Flow-Removed**: Notifies the controller when a flow table entry is deleted.
3. **Port Status**: Reports changes on a port.

---

13. SDN Control/Data Plane Interaction Example

Scenario:

1. A link failure occurs on switch S1S1S1.
2. The switch sends an **OpenFlow message** to the controller.
3. The SDN controller updates the network state and computes new routes.
4. The routing application updates flow tables for affected switches.
5. The controller installs the new flow tables using OpenFlow.

Key Takeaways

1. **SDN** separates the **control plane** from the **data plane**, enabling centralized management.
2. **SDN Controller** acts as a **network OS**, managing the global network state.
3. **OpenFlow** provides a standardized protocol for communication between the controller and switches.
4. **Network-Control Applications** offer programmable, dynamic control of the network.
5. **Traffic Engineering** challenges in traditional routing (e.g., load balancing, customized routes) are solved by SDN's flexibility.