Network Layer
Study-Ready Notes

Compiled by Andrew Photinakis

November 17, 2025

Contents

Chapter 5: The Network Layer — Control Plane

1 5.1 Introduction

2 5.2 Routing Algorithms

2.1 Concept Overview

1. Logic of how packets forwarded is done in control plane, implemented by routing algorithms
2. A good path is synonymous with ”least-cost”, such path populates forwarding table in each router

2.2 Graph Abstraction: Modeling the Network

1. Assume graph G = (N , E)
2. N = nodes in graph that represent routers in network, are points where packet-forwarding decisions are made
3. E = edges, represent physical links that connect routers
4. Costs c(x, y) = each edge (x, y) is assigned a numerical cost
   (a) Cost is typically set by network admin, is a piece of policy
   (b) 3 types of costs
      i. Physical distance, a trans-oceanic link has a higher cost than a short link
      ii. Link speed, a 1 Gbps link might have a cost of 10, while a 100 Gbps link has a cost of 1. Often inversely proportional to bandwidth
      iii. Monetary cost, leasing a link from another provider may be more ”costly”

2.3 Taxonomy of Routing Algorithms

1. Centralized
   (a) Algo computes least-cost path using complete, global knowledge of network
   (b) Has complete map of all nodes and all edge costs
   (c) Can be done at a single site (like an SDN controller) or replicated on every router, butkey is that the input is the global network state
   (d) Approach is known as a Link-State algorithm
2. Decentralized
   (a) Calculation distributed among routers

   (b) Each node begins with only local knowldge (cost of its own directly attached links)
   (c) Iterative process of communication only with neighbors, a node gradually learns least-cost paths to other nodes
   (d) Approach know as Distance-Vector algorithm

3. Static vs Dynamic
   (a) Static: routes are changed very slowly, often by human admin manually editing costs or forwarding tables
   (b) Dynamic: routes cahnge automatically as network topology or link costs change.

4. Load-Sensitive vs Load-Insentive
   (a) Load-Sensitive: link costs dynamically change to reflect current level of congestion on link. Idea to route around congestion
   (b) Load-Insensitive: link costs are fixed and do not reflect current load

2.4 5.2.1 Link State Routing Algorithm

1. Centralized approach that uses complete, global knowledge of network and lnk costs to compute least-cost paths

2. Each node in network responsible for a simple task
   (a) Discover neighbors and link costs
   (b) Broadcast this info to all other nodes in network
   (c) Receive link-state packets from all other nodes and build a complete, identical map of entire network
   (d) Compute least-cost paths from itself to all other nodes using map

3. Dijkstras Algorithm
   (a) Iterative algo, finds least-cost paths in increasing order of cost
   (b) After k-th iteration, algo has found least-cost path to kndoes that are closest to source
   (c) Sicne each router computes its own table independently based on received global info, a malfunctioning router can only broadcast an incorrect cost for its attached links, and cannot directly corrupt entire network's routing knowledge
   (d) Variables
      i. N: set of nodes to which least-cost path is definitively known
      ii. D(u): current cost of least-cost path from source (u) to node v. value updated as new, shorter paths found
      iii. p(u): predecessor node neighbor of v along current least-cost path from source. Used to build forwarding table

```
    // Initialization
1.  N' = {u}  // The source node is the only member
2.  for all nodes v
3.      if v is a neighbor of u
4.          then D(v) = c(u,v)
5.          else D(v) = infinity
6.
// Loop
7.  loop
8.      find w not in N' such that D(w) is a minimum
9.      add w to N'
10.     update D(v) for each neighbor v of w and not in N':
11.         // Does the path through w to v offer a shorter
12.         D(v) = min( D(v), D(w) + c(w,v) )
13.         // If D(v) was updated, set p(v) = w
14. until N' = N
```

(e) Pathological Oscillations

   i. Problem arises if link costs are load-sensitive, that is if cost of a link depends on amount of traffic it is carrying
   ii. Cost of link equals traffic it carries
   iii. Solution is to not use load sensitive costs and ensure routers don't run algorithm at same time. Done by randomizing when link state updates are sent or when algorithm is run

2.5 5.2.2 Distance Vector Routing Algorithm

1. Distance vector algorithm is decentralized approach
2. Code is it "tell your neighbors what you know about the world"
3. Algorithm is:
   (a) Distributed, each node communicates only with direct neighbors
   (b) Iterative, process repreats until no more information is exchanged and calcualtions stabilized
   (c) Async, nodes don't need to operate in lock step
   (d) Self-Terminating, algorithm stops when calculations converge
4. In DV, a node knows cost to its neighbors, and receives it "distance vectors" from its neighbors. A distance vector is a list of cost estimates from that neighbor to all other nodes in network

2.5.1 Bellman-Ford Algorithm

1. **Definitions of variables:**
   - $d_x(y)$: Node x's estimate of the least-cost path from x to destination y.
   - $c(x, v)$: The cost of the direct link from node x to neighbor v.
   - $N(x)$: The set of neighbors of node x.
   - $d_v(y)$: Node v's estimate of the least-cost path from v to destination y.

2. **Bellman–Ford update formula:**

   $$d_x(y) = \min_{v \in N(x)} \{c(x, v) + d_v(y)\}$$

   - Node x considers every neighbor $v \in N(x)$.
   - For each neighbor, it computes:

     $$c(x, v) + d_v(y)$$

   - It chooses the neighbor v that yields the smallest total cost.

3. **Intuition:** The cost from node x to destination y is the minimum, over all neighbors v, of the cost to reach v plus the cost (as advertised by v) for v to reach y.

4. **Distance Vector at each node:** Each node x maintains a distance vector

   $$D_x = \left[D_x(y) : y \in \mathcal{N}\right]$$

   representing its current estimates of $d_x(y)$ for all destinations y.

5. **Update behavior:** When node x receives a distance vector D_v from neighbor v, it recomputes its own distance vector using the Bellman–Ford formula. If any entry changes, node x sends its updated distance vector D_x to all neighbors.

6. **Distance Vector Algorithm (Pseudo-Code):**

```
At each node x:

1. Initialization:
2.     for all destinations y in N:
3.         D_x(y) = c(x,y)       // If y not a direct neighbor: c(x,y) = infinity
4.     for each neighbor w:
5.         send distance vector D_x to w

6. loop
7.     wait until a distance vector D_v is received from neighbor v
```

```
8.     for each destination y in N:
9.         // Bellman-Ford update
10.        D_x(y) = min over all neighbors u { c(x,u) + D_u(y) }

11.    if any D_x(y) changed:
12.        send updated distance vector D_x to all neighbors

13. forever
```

7. Pathological Dynamics
   (a) Complexity of DV is when link costs change
   (b) Cases to consider:
       i. Link cost decreases
       ii. Link cost increases at & count to infinity problem
   (c) Solution
       i. Poisoned Reverse
       ii. A simple fix for 2-node loops
           A. if z routes to x through y, z will "lie" to y and advertise its distance to x to inifinity
       iii.

2.5.2 Comparison of Link-State vs Distance-Vector Algorithms

2.5.3 5.2 Section-Wide Summary

Key Terms and Definitions

- **Routing Algorithm**: The control-plane process for finding "good" (least-cost) paths through a network of routers.
- **Graph** ($G = (N, E)$): Abstract model of a network where nodes N are routers and edges E are links with associated costs.
- **Least-Cost Path**: The path between two nodes with the minimum sum of edge costs.
- **Centralized Algorithm**: Requires complete, global network state (a full map) as input.
- **Decentralized Algorithm**: Nodes compute paths iteratively using only local information from neighbors.
- **Link-State (LS) Algorithm**: Canonical centralized approach (e.g., Dijkstra's). Each node floods its local link information to all others.

Feature	**Link-State (LS) Algorithm**	**Distance-Vector (DV) Algorithm**
Information Shared	Each node broadcasts the cost of its directly attached links to all other nodes (flooding).	Each node sends its distance vector (its current cost estimates to all destinations) only to its immediate neighbors.
Knowledge	**Global**: Every node builds a complete map of the network topology.	**Local**: A node knows only its neighbors and the information received from them.
Message Complexity	High. Each link-state update is flooded across the entire network: typically $O(N\,E)$ messages.	Lower. Only neighbor-to-neighbor exchanges; typically $O(N^2)$ over time depending on convergence.
Speed of Convergence	Fast. Once link-state information propagates, each node runs Dijkstra in $O(N^2)$ or $O(N \log N)$. No loops during convergence.	Slower. Can suffer from routing loops and the *count-to-infinity* problem.
Robustness	High. A node computes its own routes independently; a faulty node can only report incorrect local link costs.	Lower. Incorrect distance vectors can propagate from one node to another and corrupt the entire network state.
Used By	OSPF, IS-IS	RIP, BGP (path vector variant)

- **Dijkstra's Algorithm**: Greedy, iterative algorithm used in LS to compute shortest paths from a source to all other nodes.
- **Distance-Vector (DV) Algorithm**: Canonical decentralized approach (e.g., Bellman–Ford). Each node sends its distance vector to its neighbors.
- **Bellman–Ford Equation**:

$$d_x(y) = \min_{v \in N(x)} \{c(x,v) + d_v(y)\}$$

- **Count-to-Infinity**: A pathological DV behavior where a link cost increase causes slow propagation of "bad news," leading to routing loops and repeatedly increasing distance estimates.
- **Poisoned Reverse**: A mitigation for 2-node loops in DV; a node advertises an infinite distance to a destination through the neighbor it routes through.

Core Relationships

- Routing algorithms compute the paths that populate the forwarding tables in the data plane.

- Centralized algorithms correspond to LS (e.g., Dijkstra's).
- Decentralized algorithms correspond to DV (e.g., Bellman–Ford).
- The Bellman–Ford equation defines shortest paths recursively; DV implements this recursively and distributively.
- Count-to-Infinity is the major weakness of DV and results from slow propagation of link-cost increases.

Key Insights and Takeaways

- All routing algorithms model the network as a graph and attempt to solve the least-cost path problem.
- **Fundamental tradeoff:**
  - LS (centralized): robust, fast convergence, but high message overhead (flooding).
  - DV (decentralized): low message overhead (neighbor-only), but slow convergence and vulnerable to loops.
- Pathological behaviors matter:
  - LS can oscillate when link costs depend on load.
  - DV can count to infinity when link costs increase.
- The Internet uses both approaches:
  - OSPF $\rightarrow$ LS
  - BGP $\rightarrow$ DV-like (path vector)
- No single algorithm is universally superior—both excel in different contexts.

3 5.3 Intra-AS Routing in the Internet: OSPF

3.1 Need for Routing Hierarchy: Scale and Autonomy

A "flat" network where every router runs same algorithm doesn't work for global internet for 2 reasons

1. Scale
   (a) Storage and Computation
      i. Internet has hundreds of millions of routers
      ii. A link-state algorithm like Dijkstra's, which requires each router to store a map of the entire network and run an $O(N^2)$ or $O(N \log N)$ computation, is completely unfeasible at this scale.

   (b) COmmunication Overhead
      i. a link-state algo requires link-state ’advertisements’ (LSAs) to be flooded to every other router
      ii. Communication overheard would be overwhelming.
      iii. Distance vector would never converge

2. Administrative Autonomy
   (a) Internet is not one network, it’s composed of thousands of individual ISPs, universities, and corporate networks
   (b) An ISP is its own administrative domain. It wants and needs to run its network as it sees fit. SHould be able to choose its own internal routing algo and manage its network for its own performance and economic goals.
   (c) Furthermore, an ISP has no desire to ”broadcast” details of its internal network topology and link costs to its competitors

3.2 Solution: Autonomous Systems (ASs)

1. Is a group of routers that are all under same admin control, typically a single ISP or a large organization
2. Each AS identified by a globally unique Autonomous System Number (ASN), which is assigned by ICANN regional registries
3. Two-level hierarchy
   (a) Intra-AS Routing (Interior Gateway Protocol - IGP)
      i. Routing algorithm that runs within a single AS
      ii. ALl routes in AS are teammates and run samw intra-AS protocol
      iii. Focus is on performance, finding best path within network
   (b) Inter-AS Routing (Exterior Gateway Protocol - EGP)
      i. routing algorithm that runs between ASs
      ii. Protocol is used to route traffic across different ISPs
      iii. Focus is on policy, enforcing economic and security rules (e.g. ”I will not carry traffic from this competitor for free”)

3.3 OSPF: Canonical Intra-AS Protocol

1. Open Shortest Path First
2. is a link-state protocol that uses Dijkstras algo to compute shortest paths
3. How it works

   (a) Construct map: each router in AS constructs a complete, identical topological map of entire AS
   (b) Flood Information: To build map, each router broadcasts its link-state information (i.e. its list of directly connected neighbors and costs of those links) to all other routers in AS.
   (c) Run Dijkstras: Once router has complete map, it runs Dijkstras alg using itself as root node. Finds least-cost path from itself to every other node (and subnet) in AS
   (d) Install Routes: router uses results of Dijkstras to build forwarding table

4. Practical View: Link Costs
   (a) Cost of a link is not standard, its a configured parameter set by network admin
   (b) OSPF provides mechanism to find least-cost path, but policy of what "cost" means is up to admin

3.4 OSPF Link-State Broadcasts

Detail of OSPF is how it manages its link-state advertisements (LSAs)

1. Transport: messages are not sent over TCP or UDP, sent directly inside IP datagrams, using IP protocol number 89
2. Implication: Doesn't use a transport protocol, must implement its own reliability. Checks that links are operational (using HELLO messages sent to neighbors), and allows routers to get a neighbors link-state database
3. When to broadcast
   (a) On Change: A router broadcasts its link-state information whenever one of its links changes state (e.g. goes down, comes up, or its cost is unchanged)
   (b) Periodically: for robustness, router also re-broadcasts its link state at a regular interval even if there has been no change

3.5 Advanced Features in OSPFv2

is not just a simple implementation of Dijkstras. [RFC 2328] specifies several advances features that are critical in real-world networks

1. Security
   (a) How do you stop a malicious router from joining AS and injecting false link-state advertisements?
   (b) OSPF allows for authenticaion of OSPF messages
      i. Simple Auth: clear-text password, which is weak

        ii. MD5 Auth
            A. All routers configured with a shared secret key
            B. Router computes an MD5 hash of OSPF packet (including key) and sends hash with packet
            C. Receiver does same computation and verifies hash
            D. Provides message integrity and authentication, and sequence numbers are used to prevent replay attacks

2. Multiple Same-Cost Paths
    (a) What if Dijkstra's finds two paths to a destination with exact same cost?
    (b) Instead of just picking one, OSPF allows router to install both paths in its forwarding table
    (c) Traffic destined for that node can then be split across both paths
    (d) Simple and effective form of load balancing
3. Support for Hierarchy (Areas)
    (a) For a very large AS, flooding LSAs to every touer is too much overhead
    (b) Solution: AS can be partitioned into areas
    (c) Inra-Area routing: routers within an area run their own OSPF algorithm. THey flood LSAs only to other routers in same area. Each router in area has a complete map of only its area
    (d) Backbone area (Area 0): One special area is designated as backbone. Primary role of backbone is to route traffic between different areas
    (e) Area Border Routers (ABRs): Routers that sit at edge of an area. They're in backbone and in one or more other areas. Summarize routing information from their area and advertise it to the backbone, and vice-versa
4. Multicast support: OSPF extended by Multicast OSPF (MOSPF), which uses same link-state database and flooding mechanism to compute multicast paths

3.6 Section-Wide Summary

Key Terms and Definitions

- **Autonomous System (AS):** A group of routers under a single administrative control.
- **Intra-AS Routing (IGP):** Routing protocol used within an AS; focus is on performance.
- **Inter-AS Routing (EGP):** Routing protocol used between ASs; focus is on policy.
- **OSPF (Open Shortest Path First):** A widely used, open-standard, link-state, intra-AS routing protocol.

- **Link-State Advertisement (LSA):** A broadcast packet containing the state and cost of a router's links, used by OSPF.
- **Traffic Engineering:** Setting link costs to achieve network-wide performance goals (e.g., minimize congestion).
- **OSPF Area:** A sub-domain within a large AS used to create hierarchy and limit LSA flooding.
- **Backbone Area (Area 0):** The central area in a hierarchical OSPF network connecting all other areas.
- **Area Border Router (ABR):** A router that connects one or more areas to the backbone area.

Core Relationships

- The Internet is divided into ASs to address issues of scale and administrative autonomy.
- This leads to a two-tier routing system: Intra-AS (e.g., OSPF) and Inter-AS (e.g., BGP).
- OSPF is a practical implementation of the Link-State algorithm (using Dijkstra's).
- Routers flood LSAs to build an identical network map, then compute shortest paths independently.
- Large ASs use OSPF hierarchy (Areas + Backbone) to limit LSA flooding and reduce computation.

Key Insights / Takeaways

- Internet routing is fundamentally hierarchical: Intra-AS and Inter-AS levels enable global scalability.
- OSPF is the canonical Intra-AS protocol, relying on the Link-State approach.
- OSPF offers robustness: authentication, multi-path load balancing, and hierarchical scalability.
- OSPF link costs are policy tools—not purely physical measurements—and can be tuned for Traffic Engineering.

4 5.4 Routing Among the ISPs: BGP

With OSPF, a link-state protocol, is designed for scalability and performance within a single administrative domain. All routes within AS are teammates, running the same protocol with the shared goal of finding the best-performing, least-cost paths.

1. BGP
   (a) Fundamentally different from OSPF
   (b) OSPF goal is performance, BGP's goal is policy
   (c) Is a decentralized, async protocol that is similar in many ways to Distance-Vector algo

4.1 5.4.1 Role of BGP

1. Why do we need a separate protocol for inter-AS routing?
   (a) Reasons are scale and autonomy
   (b) OSPF's 'everyone knows the full map' approach won't scale to hundreds of millions of routers on global internet
   (c) Furthermore, an AS doesn't want to share its internal topology with competitors
2. Fundamentals of BGP
   (a) Within an AS, a router uses its intra-AS protocol (OSPF) to find best path to any other router inside of AS1 (let's say router 1a in AS1), but what about a destination outside its AS, say a prefix x located in AS3?
   (b) Router 1a's OSPF built forwarding table is useless for this. It needs mechanisms to:
      i. Discover that prefix x even exists
      ii. Learn a path to prefix x
      iii. Choose best path from potentially many options, based on its AS's policies
3. Routing to Prefixes
   (a) Key distinction with BGP is that it doesn't route individual host IP addresses
   (b) Routes to prefices, which are CIDRized blocks of addresses (e.g. 138.16.18/22) that represent a subnet or a collection of subnets
   (c) Every router forwarding tbale is thus a combination of routes learned from two different sources
      i. Intra-AS routes (from OSPF): typically for destiantions inside the AS
      ii. Inter-AS routes (from BGP): for destinations outside AS
   (d) BGP provides each router with means to:

i. Obtain prefix reachability information. BGP allows an AS to advertise its prefixes to rest of internet and ensures this advertisement propogates to all other ASs
ii. Determine best route: a router might learn multiple paths to same prefix. For example, AS1 might learn it can reach x via AS2, or via direct link to AS3

4.2 5.4.2 Advertising BGP Route Information

1. **Concept Overview: iBGP and eBGP**

   How does a prefix advertisement (e.g., "AS3 has x") get from a router deep inside AS3 to a router deep inside AS1?

   The process involves two "flavors" of BGP:

   - **External BGP (eBGP):** Used to communicate between routers in different ASs.
   - **Internal BGP (iBGP):** Used to communicate between routers within the same AS.

   Before we see how they work, we must define two types of routers:

   - **Gateway Router:** A router at the "edge" of an AS that has a direct physical link to a router in another AS (e.g., router 1c in Figure 5.8).
   - **Internal Router:** A router that is connected only to other hosts and routers within its own AS (e.g., router 1b in Figure 5.8).

2. **Technical Mechanism: BGP Connections**

   Unlike OSPF, BGP does not send its messages raw over IP. BGP runs over TCP on port 179.

   A BGP connection (or BGP session) is the semi-permanent TCP connection between two routers running BGP.

   An eBGP connection is a BGP session that spans two ASs (e.g., between router 1c in AS1 and router 2a in AS2).

   An iBGP connection is a BGP session between routers in the same AS (e.g., between 1c and 1b in AS1).

   This is a critical point: iBGP connections are logical connections, not necessarily physical. As shown in Figure 5.9, the iBGP session between 1c and 1b runs over the AS1 network. The TCP packets that carry the BGP message from 1c to 1b are themselves routed by AS1's intra-AS protocol (OSPF).

   Textual Representation of Figure 5.9: BGP Connections

```
          <--eBGP-->         <--eBGP-->
 (AS1) 1c ========= 2a (AS2) 2c ========= 3a (AS3)
   | \                |          |              |
   |  \ <--iBGP-->    |          | <--iBGP-->   |
   |   \              |          |              |
   |    1b            |          |      3b      |
   .    .             .          .      .       .
   .    .             .          .      .       .
(Full iBGP mesh inside each AS)
```

3. **Practical View: The Advertisement Propagation Chain**

   Let's trace the advertisement for prefix x from AS3 to all routers in AS1 (Figure 5.8/5.9):

   (a) **Start:** Router 3a (gateway) in AS3 advertises x to router 2c (gateway) in AS2 via an eBGP connection. The message is effectively: "I can reach x".

   (b) **Internal Propagation (AS2):** Router 2c receives this advertisement. It must now inform all other routers in its own AS (AS2) about this new prefix. It sends an iBGP message to all its iBGP peers (e.g., 2a, 2b, 2d) saying "I can reach x via 3a".

   (c) **Cross-AS Propagation:** Router 2a (gateway) now knows it can reach x. It advertises this fact to router 1c (gateway) in AS1 via their eBGP connection. The message is now: "I can reach x, and the path is (AS2, AS3)".

   (d) **Internal Propagation (AS1):** Router 1c receives this advertisement. It uses iBGP to inform all its internal peers (1a, 1b, 1d) of this new route: "I can reach x via the path (AS2, AS3)".

   End State: Now, every router in AS1 and AS2 knows about prefix x and also knows the path of ASs required to get there.

4.3 5.4.3 Determining the Best Routes

1. **Concept Overview: BGP Routes and Attributes**

   In the previous example, AS1 learned one path to x. But in the real Internet (as in Figure 5.10), AS1 might learn multiple paths to x. For example, if AS1 also has a direct link to AS3, it might learn two paths:

   Path 1: (AS2, AS3, x)

   Path 2: (AS3, x)

   How does a router in AS1 choose? It uses BGP's route-selection algorithm.

   A route in BGP is more than just a prefix; it's the prefix plus its associated attributes. Two of the most important attributes are AS-PATH and NEXT-HOP.

2. **Technical Mechanism: AS-PATH and NEXT-HOP Attributes**

   AS-PATH:

   - **What it is:** This attribute contains the list of ASs that the advertisement has traversed.
   - **How it's built:** When a prefix is advertised outside an AS, the AS prepends its own ASN to the list.
     - 3a advertises (AS3, x) to 2c.
     - 2c propagates this internally.
     - 2a advertises to 1c, prepending its own AS: (AS2, AS3, x).
   - **Function 1 (Loop Detection):** This is BGP's primary loop-detection mechanism. If a router receives an advertisement where its own AS is already in the AS-PATH, it rejects the route.
   - **Function 2 (Metric):** The length of the AS-PATH (number of ASs) is used as a routing metric. Shorter paths are preferred.

   NEXT-HOP:

   - **What it is:** This is the IP address of the router interface that begins the AS-PATH.
   - **Crucial Role:** This attribute is the critical link that connects the inter-AS (BGP) routing logic to the intra-AS (OSPF) forwarding logic.

   Example (Figure 5.10):

   Router 1c learns the route (AS2, AS3, x) from router 2a. The NEXT-HOP for this route is the IP address of 2a's interface.

   Router 1d learns the route (AS3, x) from 3d. The NEXT-HOP for this route is the IP address of 3d's interface.

   When 1c and 1d advertise these routes internally via iBGP to router 1b, they include the original NEXT-HOP attribute.

   Router 1b's (internal) view:

   Route 1 for x: AS-PATH = (AS2, AS3), NEXT-HOP = <IP of 2a>

   Route 2 for x: AS-PATH = (AS3), NEXT-HOP = <IP of 3d>

3. **Technical Mechanism: Hot Potato Routing**

   Given these two routes, how does 1b choose? The simplest policy is Hot Potato Routing.

   Concept: This is a "selfish" algorithm. The AS wants to get the packet "off its network" (like a hot potato) as quickly and cheaply as possible. It doesn't care about the cost outside its own network.

   Mechanism (Figure 5.11):

- Router 1b looks at the two routes it knows for x.
- It examines their NEXT-HOP attributes: <IP of 2a> and <IP of 3d>.
- 1b then consults its intra-AS (OSPF) routing table to find the internal cost to reach <IP of 2a> and the internal cost to reach <IP of 3d>.
- It selects the route whose NEXT-HOP is "closer" based on its internal OSPF costs.

If 2a is 2 hops away and 3d is 3 hops away, 1b chooses Route 1. It then looks at its OSPF forwarding table to find the local interface I that is on the shortest path to 2a, and it installs (x, I) in its forwarding table.

Key Insight: Hot Potato Routing highlights the symbiosis of BGP and OSPF. BGP provides the external path options, and OSPF provides the internal costs to reach those external paths.

4. **Technical Mechanism: The Full BGP Route-Selection Algorithm**

Hot Potato Routing is just one step in the full algorithm. BGP routers are driven by policy. The full algorithm is a sequential list of rules. A router takes all known routes for a prefix and applies these rules in order until only one route is left.

- **Local Preference (POLICY):** Each route is assigned a local_pref value by the router that learns it. This is a policy decision. The router must select the route with the highest local preference value, regardless of any other attribute.
- **Shortest AS-PATH (PERFORMANCE):** From the remaining routes (which are tied for the highest local_pref), select the route with the shortest AS-PATH length.
- **Hot Potato Routing (COST):** From the remaining routes (which are tied on local_pref and AS-PATH length), select the route with the closest NEXT-HOP router (based on intra-AS (OSPF) costs).
- **BGP Identifiers:** If still tied, use other BGP attributes (like router IDs) as a final tie-breaker.

Example Revisited: In Figure 5.10, 1b learns (AS2, AS3, x) and (AS3, x).

Assume local_pref is the same for both (Rule 1 tie).

Rule 2 (Shortest AS-PATH) is applied. (AS3, x) has length 1. (AS2, AS3, x) has length 2.

Router 1b selects the route (AS3, x).

Notice that Hot Potato Routing (Rule 3) is not even used in this case, because the AS-PATH lengths were different. This shows BGP is not a purely selfish protocol; it does prefer shorter AS-level paths, unless policy (Rule 1) dictates otherwise.

4.4 5.4.4 IP-Anycast

BGP's "best path" selection logic enables a powerful service called IP-Anycast, which is heavily used by the DNS system.

Concept: Replicate a service (like a DNS root server) in many different geographical locations, but assign the exact same IP address to all of them.

Mechanism (Figure 5.12):

- The CDN server in AS1 advertises 212.21.21.21 via BGP.
- The CDN server in AS4 also advertises the same prefix 212.21.21.21 via BGP.
- A router in AS3 will receive two different routes for 212.21.21.21: one via AS1 and one via AS4.
- The router in AS3 simply runs its standard BGP route-selection algorithm. It will pick the "best" path (e.g., the one with the shortest AS-PATH) and install that route in its forwarding table.

Result: When a client in AS3 sends a packet to 212.21.21.21, the BGP-built forwarding tables will automatically route the packet to the "closest" server instance. This provides incredible robustness and low latency for critical services like DNS.

4.5 5.4.5 Routing Policy

As we've stressed, policy is the primary driver of BGP. The examples in Figure 5.13 illustrate the two most common policy relationships: provider-customer and peer-peer.

Policy Example 1: Stub Networks (Customer/Provider)

- **Scenario:** X is a customer of B and C. W and Y are customers of A and C, respectively.
- **Goal:** X is a "stub" network. It should originate and terminate traffic, but it should never be a transit AS. That is, B should not send traffic destined for C through X.
- **Implementation (Policy):** X does not advertise routes learned from one provider (C) to its other provider (B).
- **Result:** Since B never learns a path to Y or C via X, it will never route traffic through X to get there. Policy is enforced by selectively choosing what routes to advertise.

Policy Example 2: Peering (No "Free Rides")

- **Scenario:** B and C are peers. A is a customer of B.
- **Goal:** B and C agree to exchange traffic between their respective customers. B does not want to carry traffic from C to the rest of the Internet (e.g., to A) for free.
- **Implementation (Policy):**
  - B learns a route to W via its customer A (path A-W).

  - B will advertise this route (B-A-W) to its customer X (because X pays B for Internet transit).
  - B will *not* advertise this route (B-A-W) to its peer C.

- **Result:** C never learns a path to W via B. C cannot use B as a “free ride” to get to A’s customers.

Rule of Thumb: An ISP only carries traffic if it originates from or is destined to one of its own customers.

Sidebar: Why Different Inter-AS and Intra-AS Protocols? **Policy:** Inter-AS (BGP) is all about policy. It allows an AS to control what traffic it carries and what routes it advertises based on business relationships (customer, peer). Intra-AS (OSPF) has no concept of this; its only goal is to find the best technical path.

Scale: OSPF’s “everyone-knows-the-map” (link-state flooding) approach does not scale to the hundreds of thousands of networks on the Internet. BGP (a DV-like protocol) scales by only advertising paths of ASs, not individual router links.

Performance: Intra-AS (OSPF) can be finely tuned for performance, using link costs to optimize latency or bandwidth. Inter-AS (BGP) often sacrifices performance for policy. A longer, more expensive path that satisfies a business agreement will be chosen over a shorter one that doesn’t.

4.6 5.4.6 Putting the Pieces Together: Obtaining Internet Presence

This section provides a perfect case study integrating everything from Chapter 2, 4, and 5.

Goal: Your new company, Xanadu Inc., wants to connect its network (Web, mail, DNS servers) to the global Internet.

The Protocol Checklist:

- **Get Physical:** Contract with a local ISP. You get a physical link and a block of IP addresses (a CIDR prefix, e.g., /24) from your ISP.
- **Configure Local Network (IP/DHCP):** You assign IP addresses from your block to your gateway router and your servers. You might use DHCP (Chapter 4) to assign addresses to your internal employee hosts.
- **Get a Name (DNS):** You register your domain name (e.g., xanadu.com) with a DNS registrar.
- **Register Your DNS Server (DNS):** You must tell the registrar the name and IP address of your company’s authoritative DNS server. The registrar then inserts an NS record (Name Server record) into the .com TLD servers, “delegating” xanadu.com to your server’s IP address.
- **Populate Your DNS Server (DNS):** In your own DNS server, you create the resource records for your company:

  - An A record mapping `www.xanadu.com` to your Web server's IP address.
  - An MX record mapping `xanadu.com` to your mail server's name.

- **Get Global Reachability (BGP):** This is the final, essential step. Your ISP's gateway router must advertise your prefix (your /24 block) to its upstream providers using eBGP. This advertisement then propagates via BGP across the entire Internet.

The Result (A Day in the Life):

- When a user types `www.xanadu.com`:
  - Their DNS query (Chapter 2) goes to a root server, then a .com TLD server.
  - The TLD server sees the NS record (Step 4) and directs the query to your DNS server.
  - Your DNS server replies with the A record (Step 5), giving the user your Web server's IP.
- When the user's browser sends a TCP SYN (Chapter 3) to that IP:
  - Every router on the Internet uses its forwarding table—built by BGP (Step 6)—to find the AS-PATH toward your ISP, and ultimately, to your network.
  - The packet arrives at your gateway router.
  - Your gateway router uses its internal (e.g., OSPF) forwarding table to send the packet to the final destination: your Web server.

4.7 5.4 Section-Wide Summary

Key Terms and Definitions:

- **BGP (Border Gateway Protocol):** The de facto inter-AS (inter-domain) routing protocol for the global Internet.
- **Inter-AS vs. Intra-AS:** Inter-AS is between autonomous systems (policy-driven); Intra-AS is within an autonomous system (performance-driven).
- **Gateway Router:** An AS-edge router that connects to a router in another AS.
- **Internal Router:** A router that only connects to devices within its own AS.
- **eBGP:** The BGP session between two different ASs. Used to advertise external routes.
- **iBGP:** The BGP session within an AS. Used to propagate external routes to all internal routers.
- **BGP Connection:** A logical session running on TCP port 179.
- **Route:** In BGP, a route is a prefix + attributes.

- **AS-PATH Attribute:** The list of ASs an advertisement has traversed. Used for loop detection and as a metric (shorter is better).
- **NEXT-HOP Attribute:** The IP address of the next router on the external path. This is the crucial link between BGP (inter-AS) and OSPF (intra-AS).
- **Hot Potato Routing:** A "selfish" routing policy where an AS forwards a packet to the internally closest exit point (NEXT-HOP), regardless of the external path cost.
- **Route Selection Algorithm:** The 4-step process BGP uses to select a single "best" route: 1. Highest Local Preference (Policy), 2. Shortest AS-PATH, 3. Closest NEXT-HOP (Hot Potato), 4. Tie-breakers.
- **IP-Anycast:** A service (used by DNS) where a single IP address is assigned to multiple, geographically dispersed servers, and BGP's route selection automatically directs clients to the "closest" one.

Core Relationships:

- BGP is the "glue" of the Internet, connecting all the ASs.
- BGP uses eBGP to talk to other ASs and iBGP to inform its own internal routers.
- The BGP Route Selection Algorithm shows the priority of Policy (Local Preference) over Performance (AS-PATH, Hot Potato).
- A router's final forwarding table is built using both BGP (to find the exit gateway/NEXT-HOP for external prefixes) and OSPF (to find the internal path to that NEXT-HOP).

Key Insights / Takeaways:

- Inter-domain routing (BGP) is not about finding the best path; it's about finding a policy-compliant path that works. Business relationships (customer/provider, peer/peer) dictate routing, not just link speeds.
- BGP is a Path Vector protocol, a "DV-like" protocol. Instead of just advertising a cost (distance), it advertises the entire path (the AS-PATH attribute). This is what allows it to implement policy and detect loops robustly.
- A router in an AS makes its "best" choice based on the information it has. Hot Potato Routing is the classic example of this "selfish" behavior: minimize my cost, and let the next AS worry about its cost.
- The Internet is a "network of networks." To "join" the Internet, you need more than a physical link and an IP address; your AS must participate in BGP so the rest of the world can learn a route to your prefixes.

5 5.5 The SDN Control Plane

- In our previous lectures, we have established a critical architectural principle: the separation of the network layer's data plane (the local, per-router forwarding of packets) from its control plane (the network-wide logic that determines how packets should be forwarded). We have studied the traditional implementation of the control plane, per-router control, where a routing algorithm (like OSPF or BGP) runs in every single router.
- Today, we study the second, and revolutionary, approach: logically centralized control, which is the foundation of Software-Defined Networking (SDN). In this model, the control plane is physically removed from the routers and implemented in a separate, remote controller. This "unbundling" of the network has profound implications for how networks are designed, managed, and programmed.

5.1 5.5.1 The Four Key Characteristics of an SDN Architecture

1. SDN is not a single protocol but an architectural approach defined by four key characteristics:
   - Flow-based Forwarding:
     - This is the data plane mechanism that SDN controls. We studied this in Chapter 4 as "generalized forwarding".
     - Unlike traditional routing, which forwards packets based only on the destination IP address, SDN-controlled switches (which we call "packet switches") use a flow table.
     - This flow table implements the "match-plus-action" paradigm. A "match" can be made on numerous header fields across multiple layers (e.g., L2 MAC address, L3 IP address, L4 TCP/UDP port).
     - The "action" is also general: it can be to forward, drop, modify, or send the packet to the controller. It is the SDN control plane's job to compute, install, and manage these flow table entries.
   - Separation of Data Plane and Control Plane:
     - This is the core philosophy of SDN.
     - The data plane consists of the network's switches. These are now "dumb" (but fast) hardware devices whose only job is to execute the match-plus-action rules in their flow tables.
     - The control plane consists of the remote servers and software (the "brain") that determine and manage these flow tables.
   - Network Control Functions are External to Data-Plane Switches:
     - This highlights that the control plane is physically separate from the data-plane switches.
     - As shown in Figure 5.14, the control plane itself is composed of two parts:

        - The SDN Controller (or Network Operating System): This is the core platform. It maintains a centralized view of the network's state (links, switches, hosts) and provides an API for applications to interact with it.
        - Network-Control Applications: These are the "apps" that run on the controller. They contain the actual logic for network functions like routing, load balancing, or firewalling. They use the controller's API to specify the desired behavior.
    - The controller is "logically centralized," meaning it presents a single, unified view of the network. In practice, it's a physically distributed system (run on multiple servers) for fault tolerance and scalability.
- A Programmable Network:
    - This is the payoff of the first three characteristics. The network becomes programmable via the network-control applications.
    - Analogy (The PC vs. Mainframe): The textbook correctly likens this to the "unbundling" of the PC from the monolithic mainframe.
    - Old Model (Mainframe/Traditional Router): A single vendor (e.g., IBM, Cisco) sold you a "vertically integrated" box containing the hardware, the operating system (e.g., IOS), and the applications (e.g., OSPF, BGP).
    - New Model (PC/SDN): The architecture is "unbundled" and open. You can get (1) data-plane switch hardware from one vendor, (2) an SDN controller OS from another (or open-source), and (3) network-control apps from a third vendor, or write them yourself. This open ecosystem is designed to drive innovation.

5.2 5.5.1 The SDN Control Plane: Controller and Applications

1. Let's now look inside the "brain"—the SDN controller itself. The controller's architecture is typically organized into three layers, as shown in the textbook's Figure 5.15.

2. Textual Representation of Figure 5.15: SDN Controller Architecture

    - +————————————————————-+
    - — Network-Control Applications — (e.g., Routing, Access Control, Load Balancer)
    - +————————————————————-+
    - — NORTHBOUND API (RESTful, Intent) —
    - +===+
    - — SDN Controller (Network OS) —
    - — —
    - — [Network-Wide State Management Layer] —
    - — (Link state, Host info, Switch info, Statistics,—

- — Flow tables, Network graph) —
- — —
- ————————————————————-—
- — Communication Layer —
- +==+
- — SOUTHBOUND API (OpenFlow, SNMP) —
- +——————————————————-+
- — SDN-Controlled Network Devices — (Switches)
- +——————————————————-+

3. We will examine these layers from the bottom-up.
4. The Communication Layer (Southbound Interface)
   - Purpose: This layer provides the communication channel between the SDN controller and the network's data-plane switches. This is the Southbound API.
   - Mechanism: It requires a standardized protocol for two-way communication:
     - Controller-to-Device: To send commands (e.g., "Install this flow rule").
     - Device-to-Controller: To report events (e.g., "A link just failed," "A new host just connected," or "A packet arrived that I don't have a rule for").
   - Example Protocol: The most prominent southbound protocol is OpenFlow, which we will detail next.
5. The Network-Wide State-Management Layer
   - Purpose: This is the controller's "single source of truth." It maintains a real-time, comprehensive view of the entire network's state.
   - Mechanism: This layer is essentially a distributed database that stores all network-state information, including:
     - The state of all hosts, links, and switches.
     - Statistics and counters from the switches' flow tables.
     - A copy of the flow tables installed on all switches.
     - A high-level "network graph" or topology map.
   - Practical View (Logical Centralization): This state must be consistent, reliable, and scalable. In production-grade controllers, this "state management layer" is a sophisticated, physically distributed database that uses distributed systems techniques (like consensus algorithms) to provide a logically centralized but physically distributed and fault-tolerant service.
6. The Interface to Applications (Northbound Interface)

- Purpose: This layer is the Northbound API that the network-control "apps" use to interact with the controller (the network "OS").
- Mechanism: The API allows an application to:
  - Read State: "Give me the current network graph." "Tell me the statistics for flow X."
  - Write State: "Install this set of flow rules in switches S1 and S2."
  - Register for Notifications: "Notify me whenever a link fails" or "Tell me when a new host appears on the network."
- Example APIs: Many controllers expose a RESTful API (using HTTP) for this interface, allowing for easy, language-independent application development.

7. Practical Implementation: ODL and ONOS (from Sidebar)

- The OpenDaylight (ODL) Controller:
  - Key Component: The Service Abstraction Layer (SAL) acts as the central "message bus" or "nerve center" of the controller.
  - Southbound: The SAL provides a uniform interface to plug in various southbound protocols like OpenFlow, SNMP, and NETCONF (which we'll see in 5.7).
  - Northbound: ODL provides two APIs for apps:
    - API-Driven (AD-SAL): A REST-based API for direct command-and-response.
    - Model-Driven (MD-SAL): A more advanced API where the network's configuration and state are modeled using the YANG data-modeling language. Applications then manipulate this data model, and the controller (using NETCONF) ensures the real network's state matches the model.
- The ONOS Controller:
  - Key Component: The ONOS Distributed Core is built from the ground up as a distributed system, running as an identical service on a set of servers to provide scalability and fault tolerance.
  - Southbound: Uses abstractions to mask the specific protocols (like OpenFlow) from the core logic.
  - Northbound (Intent Framework): This is a key innovation. ONOS provides a higher-level API. Instead of an app saying "Install rule X on switch Y," it states its intent:
    - e.g., "I intend for Host A and Host B to be connected."
    - e.g., "I intend for Host A and Host C to never communicate."
  - The ONOS controller itself is then responsible for translating this high-level "intent" into the specific set of low-level flow rules required on all the switches to make it happen.

5.3 5.5.2 OpenFlow Protocol

1. Concept Overview

   - OpenFlow is the most prominent protocol for the southbound API, defining the communication between an SDN controller and an SDN-controlled switch. It runs over TCP (port 6653) and allows the controller to manage the switch's flow table.

2. Technical Mechanism: Key Message Types

   - OpenFlow messages flow in both directions:
   - Messages from Controller-to-Switch (Command):
     - Configuration: Used by the controller to query and set configuration parameters on the switch.
     - Modify-State: The "workhorse" message. Used to add, delete, and modify flow entries in the switch's flow table. This is how the controller programs the data plane.
     - Read-State: Used to retrieve statistics and counters from the switch's flow tables and ports (e.g., "How many packets have matched rule 5?").
     - Send-Packet: This message, sent from the controller, actually contains a packet in its payload. It instructs the switch to send this packet out of a specific port. This is useful for injecting control-plane-generated packets (like ARP or probe packets) into the data plane.
   - Messages from Switch-to-Controller (Reporting):
     - Flow-Removed: Informs the controller that a flow table entry has been removed (e.g., because its inactivity timer expired).
     - Port-status: Informs the controller of a change in a port's status (e.g., link up, link down). This is critical for updating the network-wide state.
     - Packet-in: This is the other "workhorse" message. When a packet arrives at a switch and finds no match in the flow table, the switch sends this Packet-in message to the controller. The message contains the unmatched packet's header (or the full packet). This message is the trigger that tells the controller, "A new, unknown flow has arrived. I need instructions."

3. Practical View: Google's B4 Network (from Sidebar)

   - The textbook provides Google's B4 network as a case study of a massive, globally deployed SDN.
   - What it is: A private Wide-Area Network (WAN) that interconnects all of Google's global data centers.
   - Architecture: It is a classic SDN.
     - Data Plane: Custom-built switches running a version of OpenFlow.
     - Control Plane: A logically centralized, physically distributed set of OpenFlow Controllers (OFCs), which are part of a Network Control Server (NCS).

   - Applications: A high-level Traffic Engineering (TE) application runs on top of the controller.
   - Key Benefit: The TE application has a global view of all traffic demands and all link states. It can compute and install flow rules that split large data-copy flows across multiple paths simultaneously. This allows Google to run its WAN links at near 70% utilization, 2-3 times higher than a traditional network, saving billions of dollars.

5.4 5.5.3 Data and Control Plane Interaction: An Example

1. Let's synthesize all these concepts by tracing the complete, step-by-step process of reacting to a link failure in an SDN network.
2. Scenario (Figure 5.16):
   - An SDN controller is managing a network of switches (s1, s2, s3, s4).
   - A "Link-State Routing" application (Dijkstra's) is running on the controller, and its goal is to maintain least-cost paths.
   - The link between switch s1 and switch s2 fails.
3. Sequence of Events:
   - Data Plane Event: The link s1-s2 physically fails. Switch s1 (and s2) detects this failure on its port.
   - Southbound Notification (Switch-to-Controller): s1 sends an OpenFlow Port-status message to the SDN controller, reporting that its port connected to s2 is now "down".
   - Controller State Update: The controller's Communication Layer receives the Port-status message. It passes this event "up" to the State-Management Layer. The "link-state manager" (or topology graph) in the controller's database is updated to reflect that the s1-s2 link is broken.
   - Northbound Notification (Controller-to-Application): The Link-State Routing application had previously registered with the controller, asking to be notified of any link-state changes. The controller now sends a notification up to this application.
   - Application Logic Execution: The Link-State Routing app "wakes up."
     - It performs a Read State action via the Northbound API, requesting the new network graph from the State-Management Layer.
     - It runs its algorithm (e.g., Dijkstra's) on this new graph to re-compute all least-cost paths.
     - It identifies all the flow table entries that must be changed (e.g., on s1, s3, and s4) to reflect these new paths.

- Northbound Command (Application-to-Controller): The application sends the new set of flow rules (e.g., ”packets from s1 to s2 must now go via s4”) down to the controller via the Northbound API, instructing the ”flow table manager” to update the network.
- Southbound Command (Controller-to-Switch): The controller’s Communication Layer translates these rules into OpenFlow Modify-State messages and sends them to the data-plane switches s1, s3, and s4.
- Data Plane Action: The switches receive the Modify-State messages and install the new rules in their flow tables.
- Result: The network has converged to the new topology. The entire process was orchestrated by a single, logically centralized software application. This is profoundly different from the OSPF model, where every router would have had to independently receive a flood of messages and independently re-run Dijkstra’s algorithm.

5.5 5.5.4 SDN: Past and Future

1. Past: The Origins of SDN
   - The textbook notes that the ”unbundling” idea has a long history.
   - Researchers in the 1990s and early 2000s argued for separating the control and data planes (e.g., in ATM networks and in proposals like ForCES [RFC 3746]).
   - The direct ancestor of OpenFlow was the Ethane project at Stanford (2007). It pioneered the core SDN concepts: a centralized controller, simple flow-based switches, and the ”unmatched packets go to the controller” rule.
   - Ethane evolved into OpenFlow, and the SDN movement was born.
2. Future: Network Functions Virtualization (NFV)
   - SDN is a disruptive ”unbundling” of the router. The next logical step is to unbundle all the other ”middleboxes” in the network.
   - NFV (Network Functions Virtualization): This is an approach that aims to replace proprietary, dedicated hardware for middleboxes (like firewalls, load balancers, and caches) with software running on commodity hardware (standard servers, switches, and storage).
   - This is the same architectural principle as SDN, applied to middleboxes. It allows network operators to spin up a new ”virtual” firewall or ”virtual” load balancer as a piece of software on a server, rather than having to buy and install a new physical appliance.
3. Future: Inter-AS SDN
   - A major area of research is extending SDN’s centralized control concepts from the intra-AS setting (where there is one owner) to the inter-AS setting (where competing ISPs must cooperate).

5.6 5.5 Section-Wide Summary

1. Key Terms and Definitions:
   - SDN (Software-Defined Networking): An architectural approach that separates the data plane (hardware switches) from the control plane (software controller).
   - Flow-Based Forwarding: The "match-plus-action" data plane model used by SDN switches.
   - SDN Controller (Network OS): The "brain" of the network. A logically centralized, physically distributed software platform that manages the network's state and programs the switches.
   - Network-Control Application: Software that runs on the controller and implements specific network logic (e.g., routing, firewalling).
   - Southbound API: The interface/protocol between the controller and the switches (e.g., OpenFlow).
   - Northbound API: The interface/protocol between the controller and the network-control applications (e.g., a REST API or Intent Framework).
   - OpenFlow: The most common southbound protocol. Its key messages include Modify-State (controller-to-switch) and Packet-in (switch-to-controller).
   - NFV (Network Functions Virtualization): The concept of unbundling middlebox functions (like firewalls) from proprietary hardware and running them as software on commodity servers.
2. Core Relationships:
   - SDN is the architectural approach that separates the control and data planes.
   - Generalized Forwarding ("match-plus-action") is the data-plane abstraction that SDN controls.
   - OpenFlow is the southbound protocol that allows the SDN Controller to program the flow tables in the data-plane switches.
   - Network-Control Applications use the controller's Northbound API to implement logic (like routing) by reading the network state and writing new flow rules.
   - NFV is a "cousin" of SDN; both apply the principle of "unbundling" (separating software logic from commodity hardware) to network devices.
3. Key Insights / Takeaways:
   - SDN's primary innovation is unbundling the traditional, monolithic router, which is analogous to the shift from mainframes to the open PC ecosystem. This separation is what makes the network "software-defined."
   - The SDN controller is logically centralized (providing a single, unified view of the network) but physically distributed (for reliability and scale). This is a classic, large-scale distributed system.

- The control plane is layered, just like the protocol stack. At the bottom, the Southbound API (OpenFlow) talks to hardware. In the middle, the Controller (ODL, ONOS) manages state. At the top, the Northbound API (REST, Intent) allows applications to program the network's behavior.
- The "intelligence" of the network moves from the protocol (e.g., OSPF) to the application (a routing app running on the controller). This makes network logic easier to change, debug, and innovate.
- The Packet-in (switch-to-controller) and Modify-State (controller-to-switch) messages are the two most fundamental interactions in an SDN, representing the "reactive" loop of network control.

6 5.6 ICMP: The Internet Control Message Protocol

6.1 5.6.1 Concept Overview: The "Nervous System" of the Network Layer

- In our study of the Internet Protocol (IP) in Chapter 4, we established that IP's service model is "best-effort." This means IP provides no guarantees—it simply tries its best to forward datagrams. But what happens when things go wrong?
- What happens if a router receives a datagram with a destination it can't reach?
- What happens if a datagram's Time-to-Live (TTL) field expires?
- What happens if a host receives a UDP segment for a port that isn't open?
- In a "best-effort" world, the router or host could just silently discard the packet. This, however, would leave the sending host completely in the dark, unable to diagnose the problem.
- To solve this, the network layer needs a "messenger service"—a mechanism for hosts and routers to communicate about the state of the network with each other. This is the Internet Control Message Protocol (ICMP).
- Definition: ICMP is used by hosts and routers to communicate network-layer information to each other.
- Primary Purpose: Its most common use is for error reporting. For example, if you've ever seen a "Destination network unreachable" error, you've seen the result of an ICMP message.
- Secondary Purpose: ICMP is also used for network-layer diagnostics, most famously by the ping and traceroute utilities.

6.2 5.6.2 Technical Mechanism: ICMP's Architectural Placement

- A common point of confusion is where ICMP fits into the protocol stack.
- Analogy: ICMP is to IP as an "out-of-office" reply is to email. It's not a typical user-to-user message (like TCP/UDP), but rather a control/status message that functions at the same level as the protocol it supports.
- Architectural Position:
  1. ICMP is often considered part of IP, but architecturally, it lies just above IP.
  2. An ICMP message is carried inside an IP datagram, just like a TCP or UDP segment.
  3. When an IP datagram arrives at a host or router, the device checks the Protocol field in the IP header (which we saw in Chapter 4).
  4. If Protocol == 6, the payload is passed to TCP.
  5. If Protocol == 17, the payload is passed to UDP.
  6. If Protocol == 1, the payload is passed to the ICMP handler.
- ICMP Message Format:
  1. ICMP messages have a Type field and a Code field. The Type specifies the general class of message (e.g., "Destination Unreachable"), and the Code provides more specific information (e.g., "Host Unreachable").
- Crucial Feature: To provide context, an ICMP error message must include the entire IP header and the first 8 bytes of the original IP datagram that caused the error.
- Why is this payload important? The first 8 bytes of the original datagram's payload contain the TCP or UDP source and destination port numbers. This allows the receiving host's OS to determine which application (e.g., which web browser tab, which SSH session) generated the datagram that failed, so it can report the error to the correct process.

6.3 5.6.3 Practical View 1: Key ICMP Message Types

- ICMP is not just for errors; it's also for queries. Figure 5.19 in the textbook provides a table of common types.
- Textual Representation of Figure 5.19: Common ICMP Message Types
  - Type 0, Code 0: Echo reply (response to a ping)
  - Type 3, Code 0: Destination network unreachable
  - Type 3, Code 1: Destination host unreachable
  - Type 3, Code 2: Destination protocol unreachable

  - Type 3, Code 3: Destination port unreachable
  - Type 4, Code 0: Source quench (a deprecated congestion control message)
  - Type 8, Code 0: Echo request (the ping command)
  - Type 9, Code 0: Router advertisement
  - Type 10, Code 0: Router discovery
  - Type 11, Code 0: TTL expired (used by traceroute)
  - Type 12, Code 0: IP header bad (e.g., invalid field)

- This table provides the ”vocabulary” for IP-level communication.

6.4 5.6.4 Practical View 2: The ping Utility

- The ping program is the most basic network-layer diagnostic tool. Its sole purpose is to answer the question: ”Is that host alive and can I reach it?”
- How it Works:
  1. The ping program on the source host sends an ICMP Type 8, Code 0 message (an ”echo request”) to the destination host's IP address.
  2. The network layer at the destination host receives this ICMP packet. Its IP layer demultiplexes it to its internal ICMP handler.
  3. The ICMP handler sees it's an echo request and immediately constructs and sends an ICMP Type 0, Code 0 message (an ”echo reply”) back to the source host.
- Implementation: The ping client (the program you run) is an application-layer process. However, the ping server (the part that responds) is not a user process. It is almost always implemented as part of the operating system's kernel, directly within the network layer.

6.5 5.6.5 Practical View 3: The traceroute Utility

- The traceroute program is one of the most clever and powerful diagnostic tools. It uses ICMP in an ingenious way to discover the router-by-router path from a source to a destination.
- The Goal: To identify every router on the path and measure the Round-Trip Time (RTT) to each one.
- The Mechanism (a brilliant ”hack” on the IP header):
  1. The traceroute program on the source host constructs a series of IP datagrams, all addressed to the final destination.
  2. These datagrams contain a UDP segment with a port number that is intentionally invalid (i.e., one that is very unlikely to be in use at the destination).

- The Key Trick: The source manipulates the Time-to-Live (TTL) field of these datagrams.
  1. It sends the first datagram with TTL = 1.
  2. It sends the second datagram with TTL = 2.
  3. It sends the third datagram with TTL = 3, and so on.
  4. The source starts a timer for each datagram sent.
- The Chain of Events:
  - Datagram 1 (TTL=1): This packet is sent. It arrives at the first router on the path. The router decrements the TTL from 1 to 0. Per IP protocol rules, the router discards the datagram. It then generates an ICMP "TTL expired" message (Type 11, Code 0) and sends it back to the source.
  - Result: The source receives this ICMP message. It stops the timer for the first datagram, records the RTT, and extracts the router's IP address and name. It now knows the first hop.
  - Datagram 2 (TTL=2): This packet is sent. It arrives at the first router, which decrements the TTL to 1 and forwards it. The packet then arrives at the second router, which decrements the TTL from 1 to 0, discards the datagram, and sends back an ICMP "TTL expired" message.
  - Result: The source receives this second ICMP message, stops the second timer, and records the identity and RTT of the second hop.
  - This process continues, with each successive datagram getting one hop further into the network, "interrogating" each router along the path in sequence.
- How does it stop?
  - Eventually, a datagram will have a high-enough TTL to pass through all routers and reach the final destination host.
  - The host's IP layer receives this datagram. The TTL is not expired, so it's a valid packet. The IP layer inspects the protocol field and sees "UDP".
  - The datagram's payload is passed up to the UDP handler.
  - UDP looks at the destination port number, finds it to be invalid, and sees that no application is listening there.
  - The host's OS then generates an ICMP "port unreachable" (Type 3, Code 3), and sends this back to the source.
- Final Result: When the source traceroute program receives this "port unreachable" message, it knows it has successfully reached the final destination and the trace is complete.

6.6 5.6.6 A Note on ICMPv6

- A new version, ICMPv6, is defined in [RFC 4443] for the IPv6 protocol.
- It serves the same fundamental purposes (error reporting, diagnostics).
- It reorganizes the message types and codes to be more logical.
- It adds new message types that are essential for IPv6's new functionality.
- Example: The "Packet Too Big" message. Recall from Chapter 4 that IPv6 routers do not fragment packets. If a router receives a packet that is too large for the outgoing link, it drops the packet and sends this ICMPv6 message back to the source, telling it to resend using a smaller packet size.

6.7 5.6.7 Section-Wide Summary

- Key Terms and Definitions:
  - ICMP (Internet Control Message Protocol): A network-layer support protocol for hosts and routers to exchange control, error, and diagnostic information.
  - ICMP Message: An IP payload (Protocol 1) that includes a Type, a Code, and the header + first 8 bytes of the IP datagram that triggered it.
  - ping: A diagnostic utility that uses ICMP Type 8 (Echo Request) and Type 0 (Echo Reply) to check host reachability and RTT.
  - traceroute: A diagnostic utility that maps a network path by sending UDP datagrams with increasing TTLs to trigger ICMP Type 11 (TTL Expired) messages from each router, and an ICMP Type 3 (Port Unreachable) from the final destination.
  - ICMPv6: The updated version for IPv6, which adds new functionality like the "Packet Too Big" message.
- Core Relationships:
  - ICMP is not a transport protocol. It is a peer to TCP and UDP within the IP ecosystem, but it serves the network layer itself.
  - ICMP's "payload" (the original datagram header) is a "feedback" mechanism, allowing a host to identify which of its packets and which of its applications caused an error.
  - The ping and traceroute tools are practical, everyday applications built entirely on the ICMP message structure.
- Key Insights / Takeaways:
  - IP's "best-effort" service model would be un-debuggable without a feedback mechanism. ICMP provides this essential feedback.

  - ICMP is the source of many of the common network errors you see, such as "Destination Unreachable."
  - The traceroute program is a beautiful example of protocol "hacking" in the positive sense: it uses the standard, expected behavior of routers (decrementing TTL and sending an ICMP error on expiry) to build a powerful network-mapping tool.
  - ICMP is not just for errors; it is also for querying (e.g., ping).

7 5.7 Network Management and SNMP, NETCONF/YANG

7.1 5.7.1 The Network Management Framework

1. In our previous lectures, we have focused almost exclusively on one aspect of the control plane: routing. We've studied how protocols like OSPF and BGP build the forwarding tables that get packets from source to destination.

2. However, a network operator's job involves far more than just routing. A network is a complex, dynamic system with thousands of interacting hardware and software components. Keeping this system "up and running" involves a broad set of activities. This is the domain of network management.

3. The textbook offers an excellent, comprehensive definition:

   - "Network management includes the deployment, integration, and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost."

4. Our focus today is on the architecture, protocols, and data that network administrators use to perform these tasks. We will explore the traditional framework (SNMP) and the modern, SDN-era framework (NETCONF/YANG).

5. Managing Server (The "Manager"):

   - What it is: This is the application, typically with a human network manager in the loop, that runs in the Network Operations Center (NOC).
   - Its Role: It's the central "command and control" station. It initiates actions to configure, monitor, and control the network. It collects, processes, and analyzes all the data from the network devices. In an SDN world, this is the controller.

6. Managed Device (The "Employee"):

   - What it is: Any piece of network equipment (or its software) that resides on the managed network. This includes routers, switches, hosts, firewalls, modems, and even network-connected thermometers.
   - Its Role: It is the device that "does the work" and needs to be managed.

7. Data (The "Information"):
    - What it is: This is the "state" associated with each managed device, which the managing server needs to read and write. This data falls into three categories:
    - Configuration Data: Information set by the manager (e.g., setting a router interface's IP address, configuring OSPF link weights).
    - Operational Data: Information the device acquires as it runs (e.g., the list of OSPF neighbors it has discovered, the current forwarding table).
    - Device Statistics: Status indicators and counters (e.g., number of dropped packets on an interface, fan speed, CPU load).
8. Network Management Agent (The "Local Representative"):
    - What it is: A software process running inside the managed device.
    - Its Role: It's the server's representative on the ground. It communicates with the managing server, receives commands, and takes local action on the device. It also gathers data from the device and sends it back to the server. This is analogous to the Control Agent (CA) in an SDN router.
9. Network Management Protocol (The "Language"):
    - What it is: This is the protocol that runs between the managing server and the agent on the managed device.
    - Its Role: It defines the syntax and semantics of the messages used by the server to query the device's status and control the device. It also allows the agent to inform the server of exceptional events (e.g., a link failure).
    - Key Distinction: The protocol itself does not manage the network. It is just the tool that enables the manager to do so.
10. Textual Representation of Figure 5.20: Network Management Framework
    - +—————————————+
    - — Managing Server / Controller —
    - — (Manages data store, talks to —
    - — human network managers) —
    - +—————————————+
    - — — —
    - — (Network Management Protocol)
    - — — —
    - +———-+ +———-+ +———-+
    - — Agent — — Agent — — Agent —
    - ————-— ————-— ————-—

- — Device — — Device — — Device —
- — Data — — Data — — Data —
- +———-+ +———-+ +———-+
- Managed Managed Managed
- Device Device Device

11. Three Approaches to Management in Practice

    - CLI (Command Line Interface): The operator connects directly to the device (via console, Telnet, or SSH) and types vendor-specific commands (e.g., Cisco IOS).
    - Pros: Powerful, direct control.
    - Cons: Manual, error-prone, vendor-specific, difficult to automate, and does not scale to thousands of devices.
    - SNMP/MIB (The "Classic" Protocol Approach):
    - The operator uses the Simple Network Management Protocol (SNMP) to query or set standardized data objects (defined in a Management Information Base, or MIB) on a device.
    - Primary Use: This is the workhorse for monitoring—querying operational data and device statistics.
    - Limitation: It's less effective for configuration. It manages devices individually, and its "set" operations are often not robust enough for complex configurations.
    - NETCONF/YANG (The "Modern" SDN-era Approach):
    - This is a more holistic, network-wide approach that is "data-model-driven" and focuses on robust configuration management.
    - YANG is a data-modeling language used to define all the configuration and operational data on a device in a clear, standardized way.
    - NETCONF is a protocol that allows managers to manipulate these YANG data models, including performing atomic, multi-device transactions. This is the approach we saw in the ODL-controller's MD-SAL.

12. We will now examine the SNMP/MIB and NETCONF/YANG approaches in detail.

7.2 5.7.2 The Simple Network Management Protocol (SNMP) and the Management Information Base (MIB)

1. SNMPv3 is the current standard, defined in [RFC 3410].

2. SNMP (The Protocol)

    - Concept: SNMP is an application-layer protocol used to convey management information (control commands and data) between a managing server and an agent on a managed device.

- Operational Modes:
- Request-Response Mode: The managing server sends a request to an agent. The agent performs an action and sends a reply.
- Trap Mode: The agent sends an unsolicited trap message to the managing server.
- SNMP PDU Types (Table 5.2):
- GetRequest: Fetches the value of one or more MIB objects.
- GetNextRequest: Fetches the next MIB object.
- GetBulkRequest: Retrieves a large block of data.
- SetRequest: Sets the value of a MIB object.
- Response: The agent reply.
- SNMPv2-Trap: An unsolicited event message.
- Transport and Reliability:
- SNMP runs over UDP.
- Problem: UDP is unreliable.
- Solution: SNMP uses a Request ID so the manager can match responses. The manager must retransmit if necessary.
- Security:
- Early SNMP versions had weak security.
- SNMPv3 added authentication and encryption.

3. MIB (The Data)

   - Concept: The MIB is the collection of managed objects representing device state.
   - SMI: The data description language for MIB objects.
   - MIB Modules: Groups of related MIB objects.
   - Example MIB Object:
   - ipSystemStatsInDelivers OBJECT-TYPE
   - SYNTAX Counter32
   - MAX-ACCESS read-only
   - STATUS current
   - DESCRIPTION "The total number of datagrams successfully delivered ..."
   - ::= ipSystemStatsEntry 18
   - What this tells us: A 32-bit counter representing datagrams delivered to upper-layer protocols.

7.3 5.7.3 The Network Configuration Protocol (NETCONF) and YANG

1. The SNMP/MIB/SMI framework was excellent for monitoring but too inflexible for configuration. The IETF created NETCONF and YANG.

2. NETCONF (The Protocol)

   - Concept: NETCONF [RFC 6241] is an application-layer protocol for managing device configurations.
   - Mechanism:
   - Transport: Uses RPC paradigm.
   - Encoding: XML.
   - Security: Runs over secure, connection-oriented sessions.
   - NETCONF Session:
   - Exchange ¡hello¿ messages.
   - Server sends ¡rpc¿.
   - Device replies with ¡rpc-reply¿.
   - Device may send ¡notification¿.
   - Session ends with ¡close-session¿.
   - Key Operations (Table 5.3):
   - ¡get-config¿: Retrieve configuration.
   - ¡get¿: Retrieve configuration + operational state.
   - ¡edit-config¿: Transactional configuration.
   - ¡lock¿ and ¡unlock¿: Prevent conflicts.
   - XML Message Example (set interface MTU):
   - The full XML configuration message is included as given in the notes.

3. YANG (The Data Model)

   - Concept: YANG [RFC 6020] defines data structure and semantics.
   - Role: YANG specifies all configuration and operational data.
   - Key Features:
   - Structure: Defines full data hierarchy.
   - Data Types: Small built-in set.
   - Constraints: Enforces validity (e.g., MTU ranges, required dependencies).

7.4 5.7.4 5.7 Section-Wide Summary

1. Key Terms and Definitions:
   - Network Management: The activities used to deploy, monitor, configure, and control a network's resources.
   - Managing Server: The centralized management application.
   - Managed Device: A network component that is managed.
   - Management Agent: The software process communicating with the server.
   - SNMP: A UDP-based protocol for monitoring.
   - MIB: The database of managed objects.
   - SMI: The language defining MIB objects.
   - NETCONF: A modern XML-based, RPC-style, secure protocol for configuration.
   - YANG: The data-modeling language used by NETCONF.
2. Core Relationships:
   - The Management Framework defines Manager, Device, Agent, Data, Protocol.
   - SNMP uses SMI to define MIB objects.
   - NETCONF uses YANG to define data models.
   - Evolution: From device-by-device monitoring to full-network, model-driven configuration.
3. Key Insights / Takeaways:
   - Network Management is crucial and much broader than routing.
   - SNMP's simplicity was both strength and weakness.
   - NETCONF/YANG align with SDN, treating configuration as a programmable, model-driven system.
   - NETCONF's ¡lock¿ and transactional ¡edit-config¿ prevent conflicting changes.
   - YANG constraints move correctness into the data model.

1. X
   (a) Y