Chapter 1

L16 — Routing and BGP

1.1 Quick Review: Forwarding vs. Routing

Let's quickly review the two main jobs of the network layer from last time.

1.1.1 Forwarding: What Each Router Does

Forwarding is a local operation that happens whenever a packet arrives at a router:

- Goal: Figure out which output link to send the packet to
- How: Read the destination IP address from the packet header and look it up in the forwarding table
- It's fast and happens for every single packet

1.1.2 Routing: How Forwarding Tables Get Filled

Routing is a network-wide operation that populates forwarding tables:

- Goal: Figure out what should go in each router's forwarding table
- How: Run routing algorithms either on a centralized controller or on the routers themselves
- It's slower and happens when the network changes

1.2 How Internet Forwarding Works

The Internet uses packet switching with best-effort delivery.
Remember these key points about Internet forwarding:

- Uses packet switching (no virtual circuits or network-layer connections)
- Provides best-effort service (no guarantees about delivery or performance)
- Each forwarding table entry contains: destination IP prefix + output link
- These entries are populated by routing protocols

We talked a lot about forwarding tables in the last lecture, but we didn't really discuss how big they are. Today we'll get more concrete about forwarding table sizes and how routing works.

1.3 Every Router Knows About Every Destination

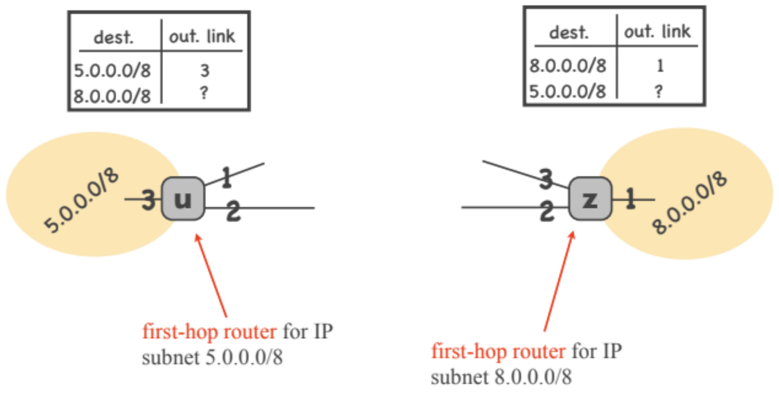
Every router on the Internet can reach every public IP address.
Let's say there's an end-system somewhere with IP address 8.0.0.1. Here's the amazing part:

- Every router on the Internet has an entry in its forwarding table that matches 8.0.0.1
- That entry tells the router which output link to use to get packets closer to 8.0.0.1
- This is true for ANY public Internet IP address, not just 8.0.0.1

This is what makes global connectivity possible - every router knows how to reach every destination!

1.4 First-Hop Routers: Where It All Starts

Some routers have a special job - they're the first ones to handle packets from local networks.



Let's look at an example with two IP subnets:

- Left subnet: uses IP prefix 5.0.0.0/8
- Right subnet: uses IP prefix 8.0.0.0/8

Each subnet connects to a router:

- Router u connects to the left subnet (5.0.0.0/8)
- Router z connects to the right subnet (8.0.0.0/8)

We call these **first-hop routers** because they're the first routers to handle packets coming from their local subnets.

1.4.1 What First-Hop Routers Know Automatically

Each first-hop router automatically knows about its own local subnet:
Router u knows:

- "Any packet for 5.0.0.0/8 goes out my link 3 (to the local subnet)"
- The network administrator configures this when setting up the router

Router z knows:

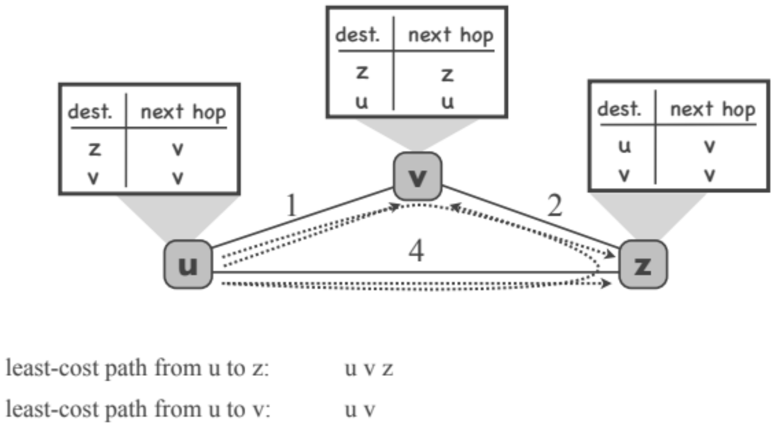
- "Any packet for 8.0.0.0/8 goes out my link 1 (to the local subnet)"
- This is also configured manually by the administrator

1.4.2 The Big Question: What About Foreign Subnets?

But here's the problem: how does router u learn to forward packets to router z's subnet (8.0.0.0/8)? And how does router z learn about router u's subnet (5.0.0.0/8)?
The network administrator can't manually configure every possible destination - there are millions of them!
The answer: A routing protocol.
Routing protocols are software systems that automatically figure out how to reach all destinations and populate the forwarding tables accordingly.

1.5 How Routing Protocols Work: A Simple Example

Let's see what happens when routers need to figure out the best paths to each other.
Suppose we have 3 routers: u, v, and z. When these routers participate in a routing protocol, their goal is to learn the best path to reach each other.


dest. next hop
z z
u u
dest. next hop
z v
v v
dest. next hop
u v
v v
v
u
z
1
2
4
least-cost path from u to z: u v z
least-cost path from u to v: u v

Let's think about router u. It needs to answer these questions:

- "When I want to send a packet to router z, should I send it directly or through router v?"
- "When I want to send a packet to router v, what's the best way?"

The other routers (v and z) need to answer similar questions about reaching their destinations.

1.5.1 What Does "Best Path" Mean?

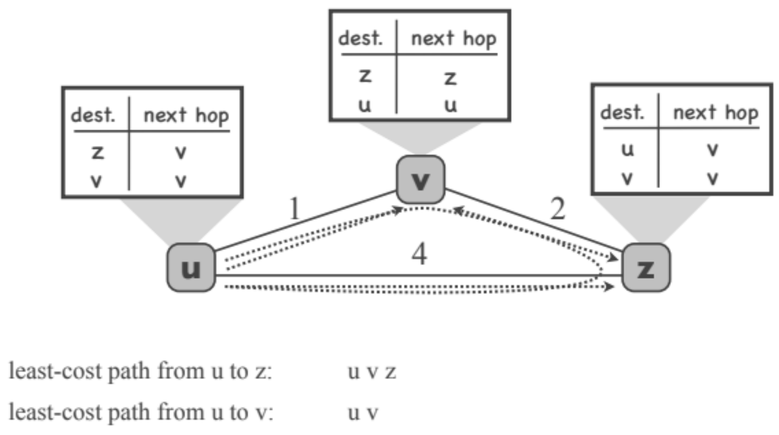
To pick the best path, we need to define what "best" means.
Each link in the network has a **cost** that represents how "bad" or expensive it is to use that link. The cost could be based on:

- How long it takes for signals to travel across the link (propagation delay)
- How much money it costs to send traffic over that link
- How congested the link is
- Some other cost metric that network administrators choose

The cost of a **path** is simply the sum of the costs of all links in that path. The **best path** from one router to another is the path with the lowest total cost—this is called the **least-cost path**.
Important note: In our examples, a link's cost is the same in both directions. In reality, links can have different costs in each direction, but we'll keep it simple for now.

1.6 Working Out the Best Paths: Step by Step

Let's figure out the best paths for router u in our example.
Looking at the network diagram, let's calculate the costs:


dest. next hop
z z
u u
dest. next hop
z v
v v
dest. next hop
u v
v v
v
u
z
1
2
4
least-cost path from u to z: u v z
least-cost path from u to v: u v

1.6.1 From Router u to Router z

Router u has two options to reach router z:

Option 1: Direct path

- Go directly from u to z
- Cost = 4

Option 2: Indirect path through v

- Go from u to v, then from v to z
- Cost = 1 + 2 = 3

Since 3 < 4, the best path is the indirect one through router v. Therefore, router u should send packets destined for z to router v as the next hop.

1.6.2 From Router u to Router v

Router u has two options to reach router v:
Option 1: Direct path

- Go directly from u to v
- Cost = 1

Option 2: Indirect path through z

- Go from u to z, then from z to v
- Cost = 4 + 2 = 6

Since 1 < 6, the best path is the direct one. Therefore, router u should send packets destined for v directly to v.

1.7 Link-State Routing Algorithms

This is the first major family of routing algorithms.
The process we just did by hand is an example of a **link-state routing algorithm**. Here's the general idea:

- **Input:** A complete map of the network—all the routers and the costs of all links connecting them.
- **Output:** The least-cost path from one starting router to every other router in the network.

The most famous link-state routing algorithm is called **Dijkstra's algorithm**.

Why It's Called "Centralized"

Link-state routing algorithms are sometimes called **"centralized"** algorithms.

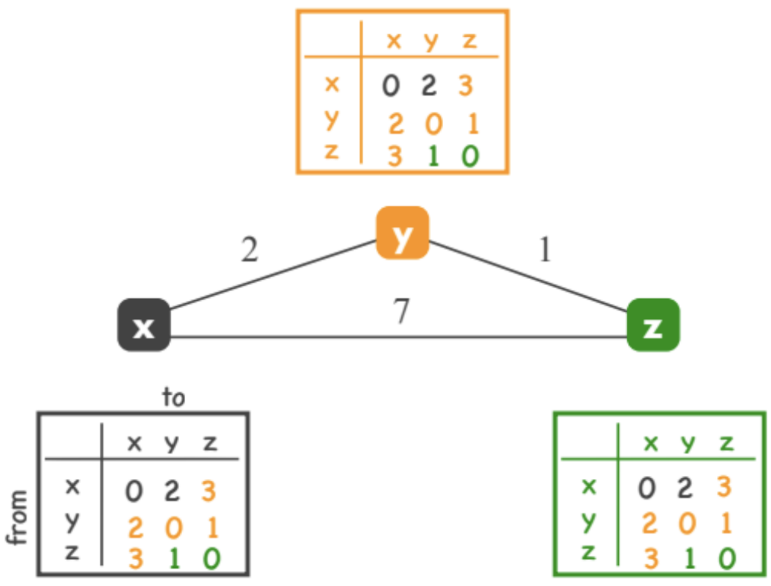
- Each router first gets a copy of the entire network map.
- Once a router has the full map, it can calculate the best paths to all other routers **by itself**, without any more communication.
- Each router runs the algorithm independently on its own copy of the map.

1.8 Distance-Vector Routing Algorithms

This is the second major family of routing algorithms, and it works very differently.
Instead of knowing the whole map, routers in a **distance-vector** algorithm only know about their immediate neighbors. They work together in rounds to figure out the best paths.

1.8.1 The Setup: What Each Router Knows Initially

Let's use our 3-router example again. In a distance-vector algorithm, each router starts by creating a table with the costs to its direct neighbors.



Router x knows:

- Cost to itself (x) is 0
- Cost to neighbor y is 2
- Cost to neighbor z is 7

Router y knows:

- Cost to neighbor x is 2
- Cost to itself (y) is 0
- Cost to neighbor z is 1

Router z knows:

- Cost to neighbor x is 7
- Cost to itself (z) is 0
- Cost to neighbor y is 1

They don't know about any indirect paths yet!

1.8.2 Round 1: Exchanging Information

In the first round, all routers send their tables to their immediate neighbors.

- Router x sends its table to y and z.
- Router y sends its table to x and z.
- Router z sends its table to x and y.

1.8.3 Round 1: Updating the Tables

Now, each router looks at the information it received from its neighbors and asks: "Can my neighbors get me somewhere cheaper than I can get there myself?"
Let's look at router x:

- Router x knows it can get to z with a direct cost of 7.
- But it just learned from router y that "y can get to z with a cost of 1".
- Router x knows it can get to y with a cost of 2.
- So, x thinks: "I can get to y (cost 2), and y can get to z (cost 1). The total cost is 2 + 1 = 3."
- Since 3 is cheaper than 7, router x updates its table: "The new best cost to reach z is 3, by going through y."

Now let's look at router z:

- Router z knows it can get to x with a direct cost of 7.
- But it just learned from router y that "y can get to x with a cost of 2".
- Router z knows it can get to y with a cost of 1.
- So, z thinks: "I can get to y (cost 1), and y can get to x (cost 2). The total cost is 1 + 2 = 3."
- Since 3 is cheaper than 7, router z updates its table: "The new best cost to reach x is 3, by going through y."

Router y doesn't find any new cheaper paths in this round because its direct connections are already the best.

1.8.4 Round 2: The Final Check

In the second round, all routers exchange their new, updated tables with their neighbors again.
This time, when they check the new information, they find that they can't improve their paths any further. Everyone already has the best possible path.
When no router can find a cheaper path, the algorithm is done! Each router has now successfully computed the least-cost path to every other router in the network.

1.9 The Bellman-Ford Algorithm

The specific distance-vector algorithm we just learned has a name.
The particular distance-vector routing algorithm we discussed is called the **Bellman-Ford algorithm**.

1.9.1 How Bellman-Ford Works

Here's the process in summary:

- **Step 1:** All neighbors exchange their routing tables
- **Step 2:** Each router checks whether it can use the new information to improve its current paths
- **Step 3:** If improvements are found, update the table
- **Repeat:** Continue until no router can improve its paths any further

The algorithm ends when no improvement is possible, which means everyone has found their optimal paths.

1.10 Link-State vs. Distance-Vector: The Big Picture

Both approaches solve the same problem but in very different ways.
We've now learned about two major families of routing algorithms:

- **Link-state algorithms** (like Dijkstra's)
- **Distance-vector algorithms** (like Bellman-Ford)

The goal is the same for both: Compute the least-cost path from each router to every other router in the network.

1.11 Which Approach Is Better?

Each approach has its own advantages and trade-offs.

1.11.1 Link-State Advantages: Speed

Link-state converges faster:

- Each router starts with the full picture of the network
- Once a router has the complete map, it can calculate all paths immediately
- The computation time can be reduced by using faster computers
- No waiting for information to propagate through multiple rounds

Think of it this way: if you have the complete map, you can plan the fastest route right away.

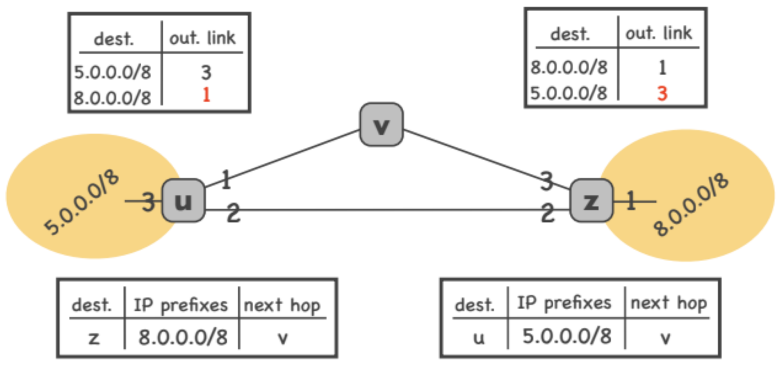
1.11.2 Distance-Vector Advantages: Efficiency

Distance-vector uses less bandwidth:

- Each router only talks to its immediate neighbors
- The number of messages per round stays constant regardless of network size
- No need to flood the entire network with information
- More efficient use of network resources

1.12 Bringing It All Together: How Routing Completes Forwarding Tables

Let's see how routing protocols solve the real problem we started with.
Remember our example from the beginning? Router u knew how to handle packets for its local subnet (5.0.0.0/8) but didn't know what to do with packets destined for 8.0.0.0/8.



Here's how a routing protocol solves this problem:

Step 1: Routers Advertise What They Own Each router announces which IP prefixes it "owns":

- Router u advertises: "I own IP prefix 5.0.0.0/8"
- Router z advertises: "I own IP prefix 8.0.0.0/8"

Step 2: Routing Protocol Finds Best Paths
Router u, router v, and router z all participate in a routing protocol (could be Dijkstra, Bellman-Ford, or any other routing algorithm). Through this protocol:

- Router u learns that the best next hop to reach router z is router v
- Router u also learns that router z owns IP prefix 8.0.0.0/8

Step 3: Complete the Forwarding Table Router u combines this information:

- "To reach 8.0.0.0/8, I need to get to router z"
- "To reach router z, my best next hop is router v"
- "Router v is reachable through output link 1"
- **Conclusion:** "Map IP prefix 8.0.0.0/8 to output link 1"

Similarly, router z learns that:

- The best next hop to reach router u is router v
- Router u owns IP prefix 5.0.0.0/8
- **Conclusion:** "Map IP prefix 5.0.0.0/8 to output link 3"

Now both routers have complete forwarding tables and can route packets anywhere!

1.13 The Reality Check: Internet Routing Challenges

The algorithms we've learned work great in theory, but the real Internet is much more complex. Designing a routing algorithm for the entire Internet faces some serious challenges:

1.13.1 Challenge 1: Scale

The Internet is **massive**:

- Millions of routers worldwide
- Millions of IP subnets to track
- Constant changes as networks go up and down

Why our simple algorithms won't work:

- **Link-state** would cause flooding disasters - imagine every router trying to tell every other router about its links!
- **Distance-vector** would never converge - it would take too many rounds for information to propagate across millions of routers
- Forwarding tables would be enormous - one entry for every IP subnet in the world

1.13.2 Challenge 2: Administrative Autonomy

Different parts of the Internet are owned by different organizations:

- Internet Service Providers (ISPs) like Comcast,Swisscom
- Universities like EPFL
- Companies like Google, Facebook, Amazon
- Government networks

The problems this creates:

- ISPs may not want to do least-cost routing (they have business reasons for their choices)
- ISPs want to hide their internal network details from competitors
- Different organizations have different policies about who can send traffic through their networks

1.14 The Internet's Solution: Hierarchical Routing

The Internet addresses these challenges through **hierarchy**: dividing into separate networks called **Autonomous Systems** (ASes).

1.14.1 Autonomous Systems

An AS is a collection of routers under single administrative control (e.g., ISP network, university network, company network). Each AS gets a unique AS number.

1.14.2 Two-Level Routing

Intra-AS routing: Within each AS, routers run their chosen algorithm (Dijkstra, Bellman-Ford, etc.)
Inter-AS routing: Between ASes, border routers use BGP to exchange routes

1.14.3 Benefits

Scale: Thousands of small algorithms instead of one massive algorithm
Autonomy: Each AS controls its own routing policies and can hide internal details

1.15 Examples of Intra-AS Routing

Different ASes can choose different routing algorithms for their internal networks.

AS 1 might choose:

- Dijkstra's algorithm for fast convergence
- Link costs based on bandwidth and delay

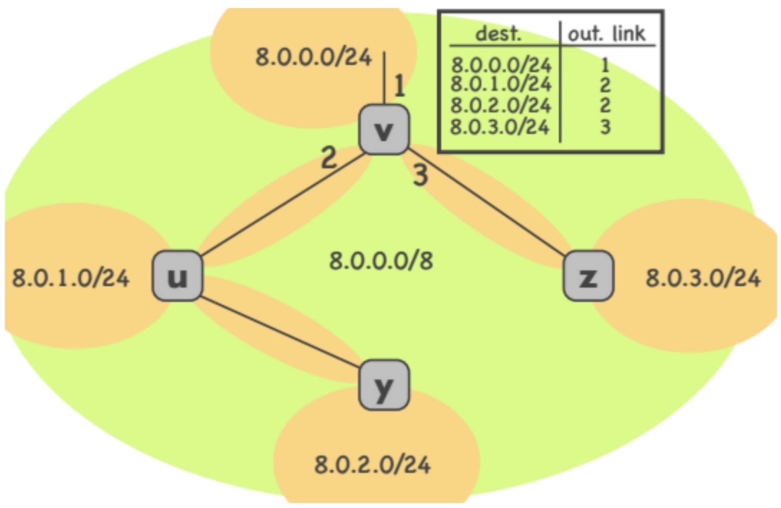
AS 2 might choose:

- Bellman-Ford algorithm to save bandwidth
- Link costs based on monetary cost

Each AS makes its own decision based on its specific needs and constraints.

1.16 A Concrete Example: How an AS Works

Consider an AS that has 4 routers, each providing connectivity to 1 IP subnet:



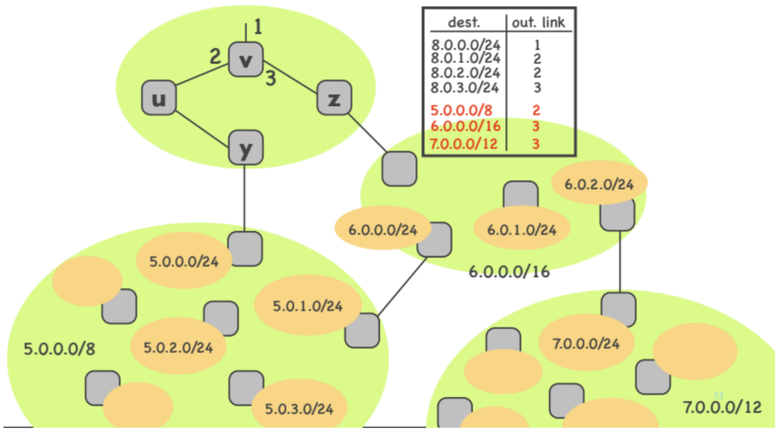
dest.	out. link
8.0.0.0/24	1
8.0.1.0/24	2
8.0.2.0/24	2
8.0.3.0/24	3

1.16.1 Step 1: Learn About Local Destinations

Each of the 4 routers must discover the best path to each local router and subnet. They achieve this by participating in an intra-AS routing protocol (like Dijkstra or Bellman-Ford).

1.16.2 Foreign Destinations

Each router must also learn routes to foreign ASes, but **not** to every individual subnet in those ASes.

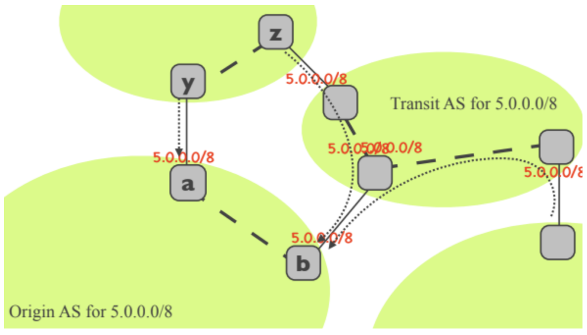


dest.	out. link
8.0.0.0/24	1
8.0.1.0/24	2
8.0.2.0/24	2
8.0.3.0/24	3
5.0.0.0/8	2
6.0.0.0/16	3
7.0.0.0/12	3

Router v needs only 1 entry for each of the 3 foreign ASes, not separate entries for every subnet within those ASes. This dramatically reduces forwarding table sizes.

1.17 Border Routers and BGP

Border routers sit at the "edge" of each AS and handle communication between different ASes.



All border routers participate in **Border Gateway Protocol (BGP)**, a variant of Bellman-Ford. Through BGP, each border router:

- Advertises prefixes from its own AS to neighbors
- Learns routes to foreign prefixes

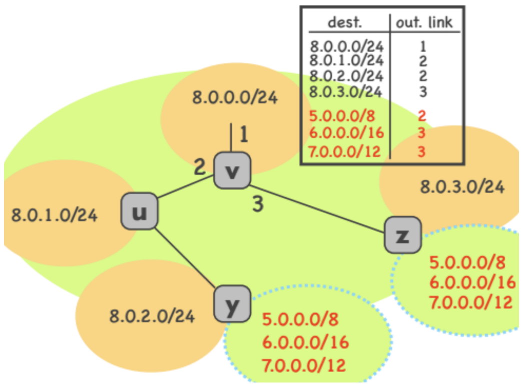
1.17.1 BGP Example

- Bottom left AS aggregates local subnets into 5.0.0.0/8 (becomes the "origin AS")
- Border routers a and b advertise "route to 5.0.0.0/8" to their neighbors
- Top right AS decides to do transit, propagating the route further
- Eventually all ASes learn how to reach 5.0.0.0/8

1.18 How Non-Border Routers Learn External Routes

Router v (not a border router):

- Learns local routes from other local routers (intra-AS routing)
- Learns foreign routes from border routers y and z
- Chooses least-cost route when multiple options exist



dest.	out. link
8.0.0.0/24	1
8.0.1.0/24	2
8.0.2.0/24	2
8.0.3.0/24	3
5.0.0.0/8	2
6.0.0.0/16	3
7.0.0.0/12	3

1.19 Internet Routing Summary

1.19.1 Intra-AS Routing

- **Participants:** All routers in the same AS
- **Protocols:** OSPF, RIP, others (each AS chooses)
- **Goal:** Propagate routes within local AS

1.19.2 Inter-AS Routing

- **Participants:** Border routers between ASes
- **Protocol:** BGP (universal - only one protocol used)
- **Goal:** Propagate routes between ASes

Internet Architecture Foundation

The Internet's network layer rests on two components:

- **IP:** Specifies forwarding, packet format, addressing
- **BGP:** Inter-domain routing protocol enabling global connectivity