Chapter 1

L15 — Forwarding and IP

1.1 What is the Network Layer?

The network layer is where we first see actual network devices called **routers**.
We have end-systems (like Alice's computer and Bob's computer) and routers that help move packets between them. A router is a smart device that can look at packets and decide where to send them next.

1.2 Packet Headers We Care About

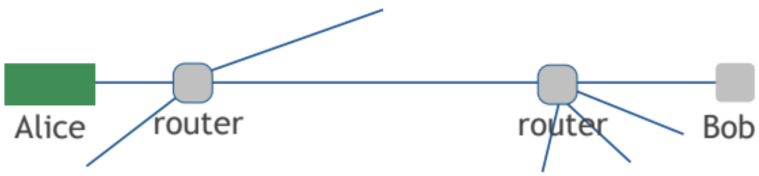
We need to understand two types of packet headers:

- **TCP header**: Contains *source port* and *destination port* (plus other things like SYN flag, checksum, sequence numbers, etc.)
- **IP header**: Contains *source IP address* and *destination IP address*

The IP addresses are what routers use to figure out where packets should go.

1.3 Two Main Jobs of the Network Layer

The network layer does two main things: **forwarding** and **routing**.



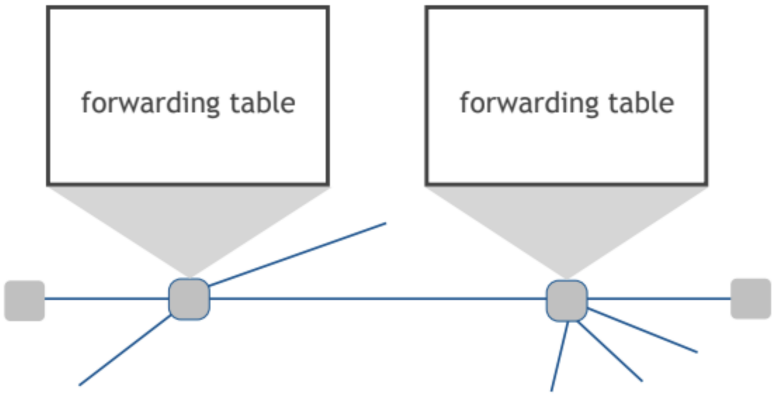
These work together to get packets from source to destination.

1.3.1 Forwarding: What Each Router Does

Forwarding is what happens when a packet arrives at a router. The router asks: "Where should I send this packet next?"
This is a quick decision that each router makes for every packet that comes through.

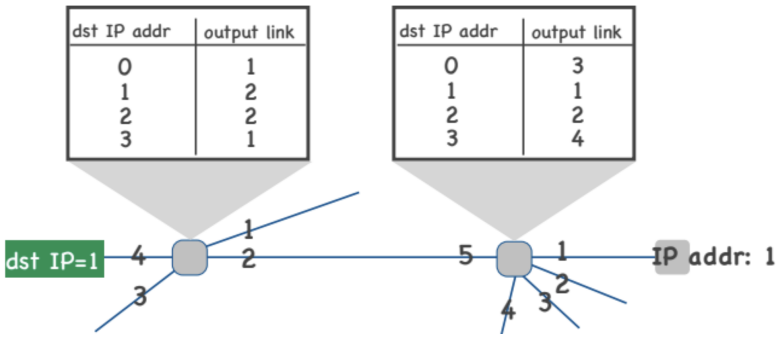
How Routers Are Set Up

Each router has several connections (called links). The router gives each link a number, like Link 1, Link 2, Link 3, etc.
The router also has a **forwarding table** - think of it like a phone book that says "if a packet is going to IP address X, send it out Link Y."



Step-by-Step: How Forwarding Works

Let's say Alice wants to send a packet to Bob, and Bob's IP address is 1.

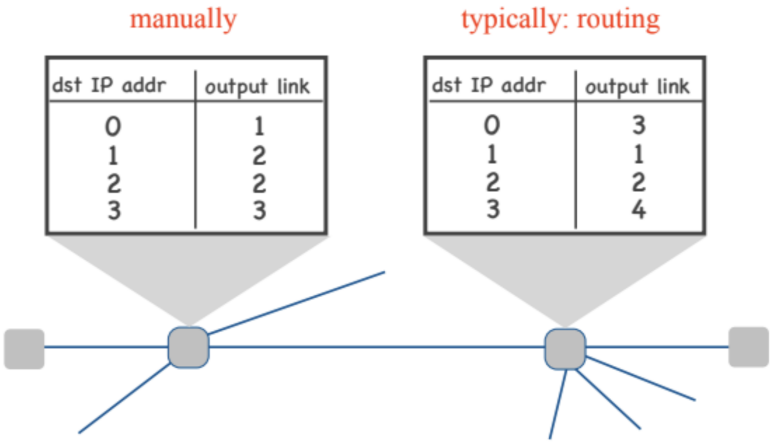


Here's what happens:

1. Alice puts "1" as the destination IP address in her packet
2. The first router gets the packet and reads "destination = 1"
3. The router looks in its forwarding table: "packets for IP 1 go out Link 2"
4. The router sends the packet out Link 2
5. The next router does the same thing with its own table
6. This continues until the packet reaches Bob

1.3.2 Routing: How Do Forwarding Tables Get Filled?

Routing is about filling up those forwarding tables. Someone has to tell each router "for IP address X, use Link Y."



Forwarding asks "where does this packet go?" Routing asks "how do we figure out where packets should go?"

Two Ways to Fill Forwarding Tables

- **Manual**: A network administrator types in the rules
- **Automatic**: Software figures it out and fills in the tables

Most of the time, it's automatic.

Centralized Routing: One Brain Controls Everything

Imagine one smart computer (called a **network controller**) that knows about all the routers in the network.

Good things about this approach:

- The controller sees the whole network, so it can make good decisions
- All routers get consistent information
- Easy to implement complex policies

The controller:

- Knows where all routers are and how they connect
- Figures out the best forwarding table for each router
- Sends this information to each router

Distributed Routing: Routers Talk to Each Other

Instead of one controller, the routers talk directly to each other and work together to figure out the best routes.

Good things about this approach:

- If one router breaks, the others keep working
- Works better when you have lots of routers
- Routers can quickly adapt to changes nearby

Real Internet: Mix of Both

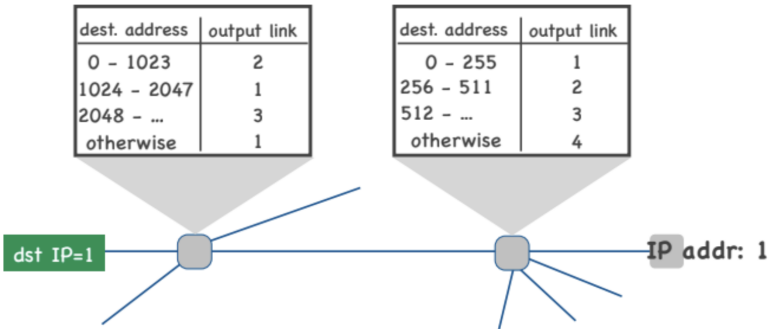
The actual Internet uses both approaches in different places.

1.4 Making Forwarding Tables Smaller

Here's a problem: there are about 4 billion possible IP addresses. Do routers really need 4 billion entries in their forwarding tables?
No! That would be way too big.

1.4.1 Using Ranges Instead of Individual Addresses

Instead of storing every single IP address, routers store **ranges** of IP addresses.



For example, instead of having separate entries for IP addresses 0, 1, 2, ..., 1023, a router can have one entry that says "all IP addresses from 0 to 1023 go out Link 2."

How This Works

When a packet arrives:

1. Router looks at the destination IP address
2. Router finds which range this IP address fits into
3. Router sends the packet out the link for that range

dest. address range			output link
0 - 3	0000 - 0011	00**	1
4 - 7	0100 - 0111	01**	2
8 - 11	1000 - 1011	10**	3
12 - 15	1100 - 1111	11**	4

1.4.2 IP Prefixes: A Smart Way to Write Ranges

Let's say we only have 16 IP addresses (0 through 15) to make this easier to understand.

Range	Binary	Prefix	Link
0-3	0000-0011	00**	1
4-7	0100-0111	01**	2
8-11	1000-1011	10**	3
12-15	1100-1111	11**	4

The "00**" means "any address that starts with 00". The ** can be anything (00, 01, 10, or 11).

1.4.3 Longest Prefix Matching: Handling Exceptions

Sometimes we need exceptions. What if most addresses starting with "00" go to Link 1, but address "0011" (which is 3) needs to go somewhere else?

Prefix	Link	What This Covers
00**	1	addresses 0, 1, 2, 3
0011	2	address 3 (exception!)

When a packet for address 3 arrives:

- It matches both "00**" and "0011"
- The router picks the **longer** (more specific) match: "0011"
- The packet goes to Link 2

This is called **longest prefix matching**.

More Exceptions = Bigger Tables

The more exceptions you need, the bigger your forwarding table gets:

dest. address range			output link
0 - 1	0000 - 0001	000*	1
2	0010	0010	2
3	0011	0011	3
4, 6, 7	0100, 0110, 0111	01**	2
5	0101	0101	4
8 - 15	1000 - 1111	1***	1
10	1010	1010	3

This is why we want to avoid lots of exceptions.

1.5 Why Location Matters for IP Addresses

For the Internet to work well, IP addresses should be related to location.

1.5.1 The Basic Idea

- Devices that are close together should have similar IP addresses
- Devices in the same building/city/country should share the same prefix
- Similar IP addresses should mean similar locations

1.5.2 Why This Helps

- **Smaller forwarding tables**: One entry can cover many destinations
- **Easier routing**: Routes can be grouped together
- **Faster updates**: Changes affect fewer table entries

1.5.3 What Happens When Location Rules Break

Here's an example of what goes wrong:

1. Let's say all EPFL computers have IP addresses starting with the same prefix
2. Routers worldwide can have one simple rule: "send all EPFL traffic toward Switzerland"
3. Now imagine EPFL students travel around the world but keep their EPFL IP addresses
4. Suddenly routers need special exception rules for each traveling student
5. If lots of people do this, forwarding tables become huge and unmanageable

This shows why keeping IP addresses tied to location is important for the Internet to work efficiently.

1.6 How IP Addresses Are Written

Now let's learn how to actually write and read IP addresses and IP prefixes.

1.6.1 IP Address Format

An IP address is just a number from 0 to $2^{32} - 1$ (that's about 4 billion different numbers).
We could write it in binary (all 0s and 1s), but that would be really long. Instead, we use something called "dot format."
Here's how it works:

- Take the 32-bit binary number
- Split it into 4 groups of 8 bits each
- Convert each group to a regular decimal number (0-255)
- Put dots between them

Example:

$$\text{Binary:} \quad 11011111\ 00000001\ 00000001\ 00000001 \tag{1.1}$$

$$\text{Dot format:} \quad 223.1.1.1 \tag{1.2}$$

1.6.2 IP Prefix Format

Remember how we said routers use ranges of IP addresses? We call these ranges **IP prefixes**.
An IP prefix is written as: **IP address / mask number**
For example: `223.1.1.0/24`

What the Mask Number Means

The mask number tells us how many bits from the left are "fixed" (can't change).
For `223.1.1.0/24`:

- The first 24 bits must stay the same
- The last 8 bits can be anything
- This covers all addresses from 223.1.1.0 to 223.1.1.255

In binary, this looks like:

$$\text{Fixed part:} \quad 11011111\ 00000001\ 00000001\ ******** \tag{1.3}$$

$$\text{Or simply:} \quad 223.1.1.* \tag{1.4}$$

The * means "can be any combination of 0s and 1s."

Different Ways to Write the Same Prefix

Here's something interesting: these are all the same IP prefix!

- `223.1.1.0/24`
- `223.1.1.74/24`
- `223.1.1.113/24`
- `223.1.1.*`

Why? Because they all have the same first 24 bits. The last 8 bits don't matter for defining the prefix.

More Examples

Bigger range: `223.1.1.0/8`

- Only the first 8 bits are fixed
- Covers: `223.*.*.*`
- That's all addresses from 223.0.0.0 to 223.255.255.255

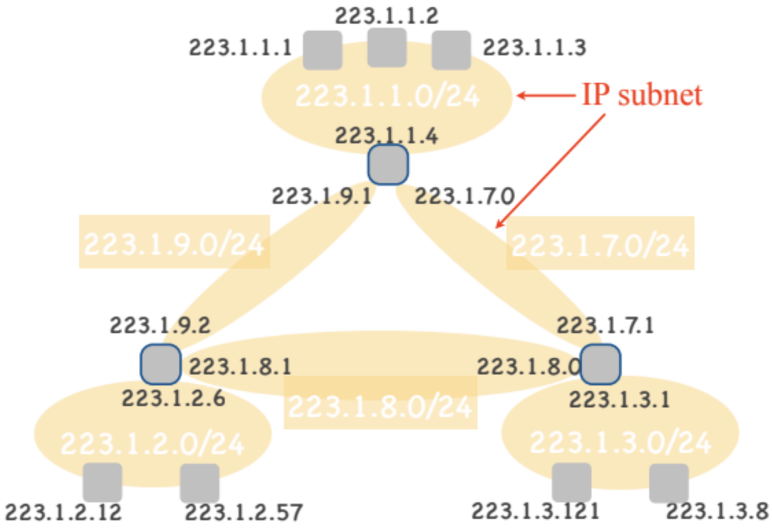
Tricky example: `223.1.1.0/12`

- First 12 bits are fixed
- In binary: `11011111 0000**** ******** ********`
- Covers: 223.0.0.0 to 223.15.255.255

Pro tip: When the mask isn't a multiple of 8 (like /12), it's tricky to figure out the range in your head. It's better to convert to binary and count bits carefully!

1.7 How the Internet Is Organized

The Internet is divided into chunks called **IP subnets**.



1.7.1 What Is an IP Subnet?

Think of an IP subnet as a neighborhood:

- It contains end-systems (computers, phones, etc.)
- It has routers at its "borders" that connect to other neighborhoods
- Everyone in the neighborhood has similar addresses (same IP prefix)
- The routers don't live "inside" the neighborhood - they're at the edges

1.7.2 How IP Addresses Are Assigned in Subnets

Each subnet gets its own IP prefix. For example:

- Top subnet might get `223.1.1.0/24`
- Bottom subnet might get `223.1.2.0/24`
- Middle subnets might get `223.1.3.0/24`, etc.

All devices in a subnet get IP addresses from that subnet's prefix.

1.7.3 Routers Have Multiple IP Addresses

Here's something cool: each router has one IP address for each subnet it touches.
Look at the top router in the picture:

- It touches 3 different subnets
- So it has 3 different IP addresses
- Each address belongs to the prefix of that subnet

It's like living at the corner of three neighborhoods - you need an address in each one!

1.7.4 How Routers Use This Information

When a router gets a packet, it looks at the destination IP address and asks:

- "Is this address in one of my local subnets?" → Send it directly there
- "Is this address in a foreign subnet?" → Forward it toward that subnet

For example:

- Packet for `223.1.1.1` → "That's in my top subnet!" → Send it up
- Packet for `223.1.2.16` → "That's in a different subnet" → Forward it toward that subnet

1.8 Special IP Addresses

1.8.1 Broadcast Address

Each subnet has a special **broadcast address**:

- It's the biggest IP address in the subnet
- When you send a packet to this address, it goes to *everyone* in the subnet
- For subnet `223.1.1.0/24`, the broadcast address is `223.1.1.255`

It's like shouting "Hey everyone!" in a room.

1.9 How Do Organizations Get IP Addresses?

1.9.1 Getting IP Prefixes

Organizations get their IP prefixes from:

- Their Internet Service Provider (ISP)
- A regulatory body (for big organizations)

1.9.2 Assigning Individual IP Addresses

Once an organization has its prefix, it assigns individual addresses:
For router interfaces:

- Usually done manually by network administrators

For end-systems (computers, phones):

- Can be done manually
- More often done automatically using DHCP (Dynamic Host Configuration Protocol)

DHCP is like having an automatic address assignment system - when your laptop joins a network, DHCP gives it an available IP address from that network's range.

1.10 Best-Effort Delivery

Here's an important thing to understand about the Internet: it provides **best-effort delivery**.

1.10.1 What Best-Effort Means

- The Internet tries its best to deliver your packets
- But it makes **no promises** that they'll actually arrive
- Packets might get lost, delayed, or arrive out of order
- The network says "I'll do my best, but no guarantees!"

1.10.2 Why Best-Effort?

This might sound bad, but it's actually a smart design choice:

- Keeps the network simple and fast
- Makes it cheaper to build and operate
- Applications can add their own reliability on top if they need it
- Works well for most uses (web browsing, email, etc.)

Think of it like the postal service - they try to deliver your mail, but sometimes letters get lost. For important things, you can pay extra for certified mail (like how applications can add extra reliability).

1.11 Virtual Circuits: A Different Way to Do Networking

So far we've talked about how the Internet works today (packet switching with best-effort delivery). But there's another way to build networks called **virtual circuits**.

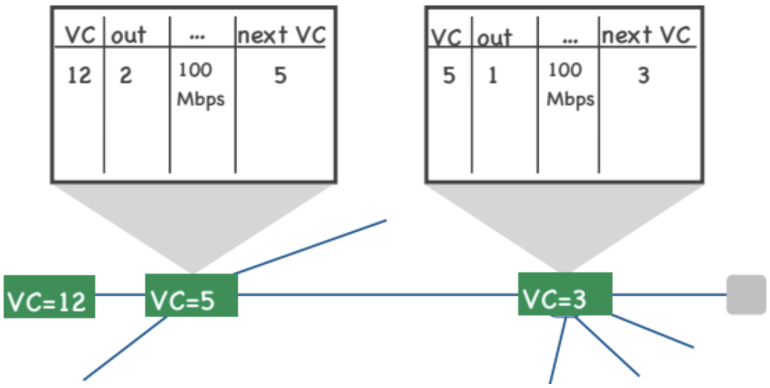
1.11.1 The Problem with Best-Effort

Remember how the Internet makes no promises about packet delivery? Sometimes you want guarantees. For example:

- Video calls need consistent, fast delivery
- Important business applications can't afford lost packets
- Some applications need a minimum guaranteed speed

1.11.2 How Virtual Circuits Work

Virtual circuits are like making a reservation at a restaurant - you book resources in advance.
Here's how Alice could get a guaranteed 100 Mbps connection to Bob:



Step 1: Connection Setup Request

Alice sends a special "connection-setup request" packet that says: "I want a network connection to Bob with guaranteed 100 Mbps speed."

Step 2: Each Router Decides

The packet travels through routers on the path to Bob. Each router asks itself:

- "Do I have enough spare capacity to guarantee 100 Mbps?"
- "Can I reserve these resources for Alice and Bob?"

If a router says yes:

- It reserves 100 Mbps of capacity for this connection
- It assigns a Virtual Circuit (VC) number to this connection (like giving it a name)
- It creates a table entry to remember this reservation

For example:

- First router assigns VC #12
- Second router assigns VC #5
- Third router assigns VC #3

Step 3: Connection Establishment

If ALL routers on the path agree, Bob gets the request and sends back an "OK" message. This travels back through all the routers so they can coordinate their VC numbers.
Now Alice has a guaranteed 100 Mbps "highway" to Bob!

Step 4: Using the Virtual Circuit

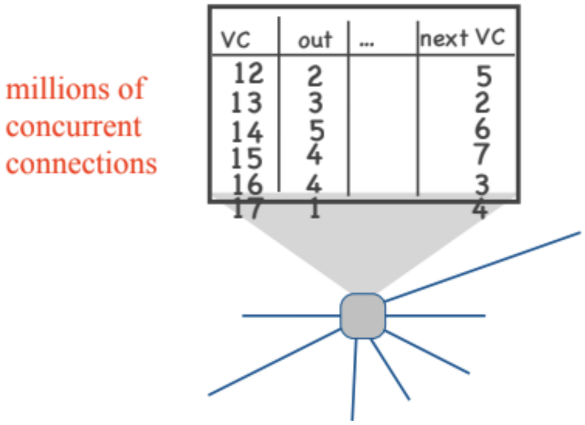
When Alice sends packets to Bob:

- She puts "VC #12" in each packet header
- First router sees "VC #12", knows this is Alice's guaranteed connection
- Router changes the header to "VC #5" and forwards to next router
- Second router sees "VC #5", changes to "VC #3", forwards to third router
- And so on until the packet reaches Bob

Each router knows exactly how to treat these packets because of the reservation.

1.11.3 The Big Challenge: Too Much State

Here's the problem with virtual circuits: routers have to remember LOTS of information.



Memory Requirements

Each router needs to keep a table entry for every active connection passing through it. A busy Internet router might have:

- Millions of concurrent connections
- Each connection needs memory for state information
- This adds up to huge memory requirements

1.11.4 Packet-Switched Networks (Like Today's Internet)

How they work:

- Use packet switching - no network-layer connections
- Best-effort delivery (no guarantees)
- Routers only store destination prefixes and output links
- State is populated by routing protocols

Good for: Best-effort service

1.11.5 Why the Internet Chose Packet Switching

Packet switching won because it:

- Makes forwarding tables smaller (no per-connection state in routers)
- Makes routers simpler (no connection setup/teardown needed)
- Eliminates security risks from connection-based attacks

1.12 A Fundamental Internet Principle

Every computer should be able to talk to every other computer.

1.12.1 Global Reachability

The Internet is built on this idea:

- Every end-system must be reachable from any other end-system
- This requires a globally unique IP address for every end-system
- No two devices anywhere in the world should have the same IP address

This is like having a postal system where every house has a unique address - no duplicates allowed anywhere!

1.13 The IP Address Crisis

In the 2000s, we started running out of IP addresses.

1.13.1 The Problem

- IPv4 has about 4 billion possible addresses
- The Internet grew faster than expected
- We were running out of unique addresses to assign

1.13.2 Two Solutions

Solution 1: IPv6

- A new version of IP with way more addresses
- Deployed in many areas, but not everywhere yet
- Long-term solution but takes time to implement

Solution 2: Network Address Translation (NAT)

- A clever workaround that lets multiple devices share one public IP address
- Widely deployed and working today
- Has some limitations (which we'll explain)

1.14 Network Address Translation (NAT)

How to let many devices share one public IP address.

1.14.1 The Basic Idea: Private Address Spaces

Some IP address ranges are designated as "private":

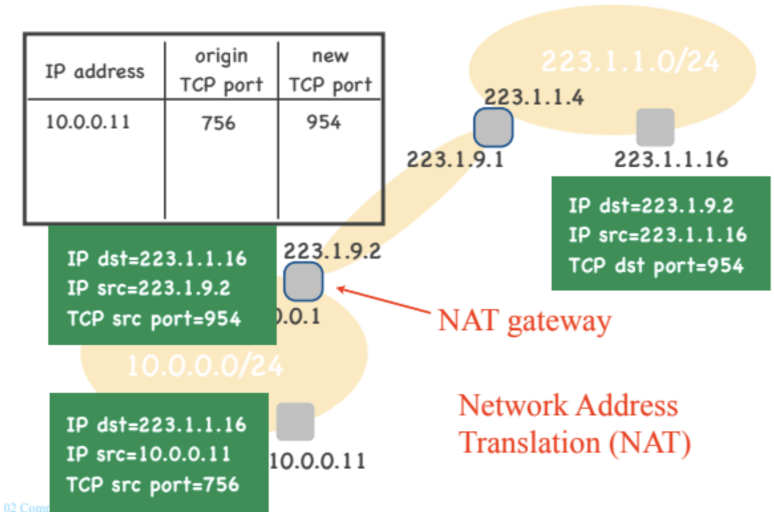
- Example: 10.0.0.0/24 is a private range
- Multiple networks can use the same private addresses
- Like having multiple houses with address "123 Main Street" - it's OK as long as they're in different towns

1.14.2 The Rule for Private Addresses

- Private IP addresses can only be used within their local network
- Packets with private addresses can NEVER leave the local network
- It's like calling "John!" in your house - only works if there's only one John there

1.14.3 How NAT Works: A Step-by-Step Example

Let's say device 10.0.0.11 (private address) wants to talk to 223.1.1.16 (public address):



Step 1: Outgoing Packet

- Device 10.0.0.11 sends packet with source=10.0.0.11, destination=223.1.1.16
- It also has a TCP source port (let's say 756)

Step 2: NAT Gateway Translation

The NAT gateway (border router) does magic:

- Replaces source IP 10.0.0.11 with its own public IP 223.1.9.2
- Replaces source port 756 with a new port number (say 954)
- Remembers this mapping in a table: "10.0.0.11:756 $\leftrightarrow$ 954"

Step 3: Response Packet

- Server 223.1.1.16 responds to 223.1.9.2:954
- It thinks it's talking to 223.1.9.2, not 10.0.0.11

Step 4: Return Translation

- NAT gateway gets response packet addressed to 223.1.9.2:954
- Looks up port 954 in its table: "Oh, this goes to 10.0.0.11:756"
- Changes destination to 10.0.0.11:756 and forwards internally

1.14.4 What NAT Does

For outgoing packets:

- Rewrites source IP address and port number
- Maps original address:port to new port number
- Stores this mapping for future use

For incoming packets:

- Rewrites destination IP address and port number
- Uses stored mapping to find the right internal device

1.15 Problems with NAT

NAT works, but it breaks some fundamental Internet principles.

1.15.1 Problem 1: You Can't Reach Devices from Outside

- Devices with private addresses are "hidden" behind the NAT gateway
- External devices can't initiate connections to them
- Internal devices can call out, but external devices can't call in
- Unless you manually configure the NAT gateway with special rules

This breaks the global reachability principle!

1.15.2 Problem 2: NAT Gateways Need State

Remember how we said routers shouldn't keep per-connection state? Well, NAT gateways do exactly that:

- They remember mappings for every active connection
- This is the same problem we had with virtual circuits!

Why is this OK?

- NAT gateways are usually at the edge (like your home router)
- They only handle connections from a small number of devices
- Not millions of users like a core Internet router

1.15.3 Problem 3: It's a Layering Violation

NAT breaks the clean separation between network layers:

- The network layer (IP) shouldn't care about transport layer info (TCP ports)
- But NAT has to look at and modify TCP port numbers
- This makes the system more complex and fragile