Lecture 7:

The Network Layer

Katerina Argyraki, EPFL

Today we will start exploring the network layer, the first one that includes actual network devices.

Outline

- Network-layer functions
  - forwarding
  - routing
- Network-layer types
  - virtual-circuit networks
  - datagram networks
- IP forwarding
- IP routing

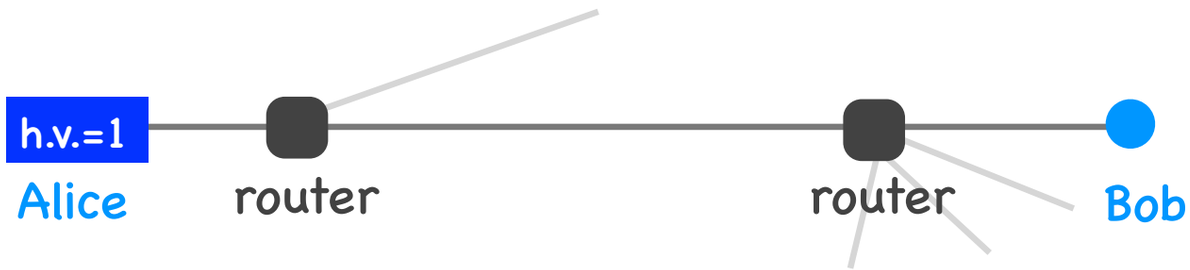
Here is our outline:

- We will first discuss the two basic functions of a network layer, which are forwarding and routing.
- Then, we will discuss two basic types of network layers, which are virtual-circuit networks and datagram networks.
- And then we will focus on the network layer of the Internet, which is IP, and which happens to be a datagram network layer, and we will discuss how it does forwarding and how it does routing.

Outline

- Network-layer functions
  - forwarding
  - routing
- Network-layer types
  - virtual-circuit networks
  - datagram networks
- IP forwarding
- IP routing

Let’s start from the two basic network-layer functions.

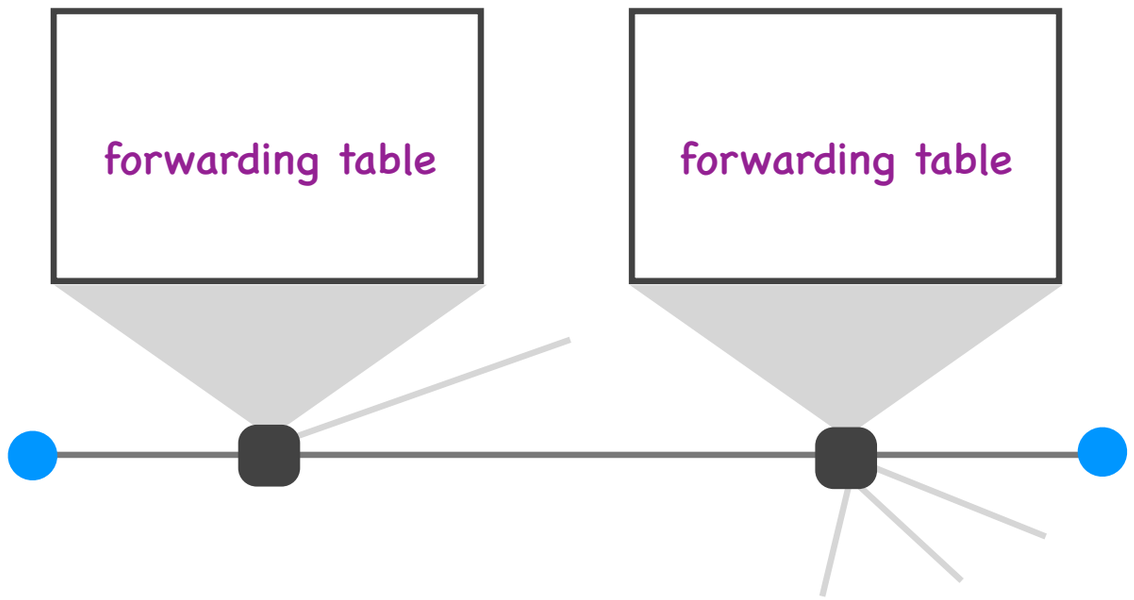


Forwarding is a process that takes place inside a router (also called network-layer switch)
and determines what to do with a piece of data received by that device, e.g., where to send it next.

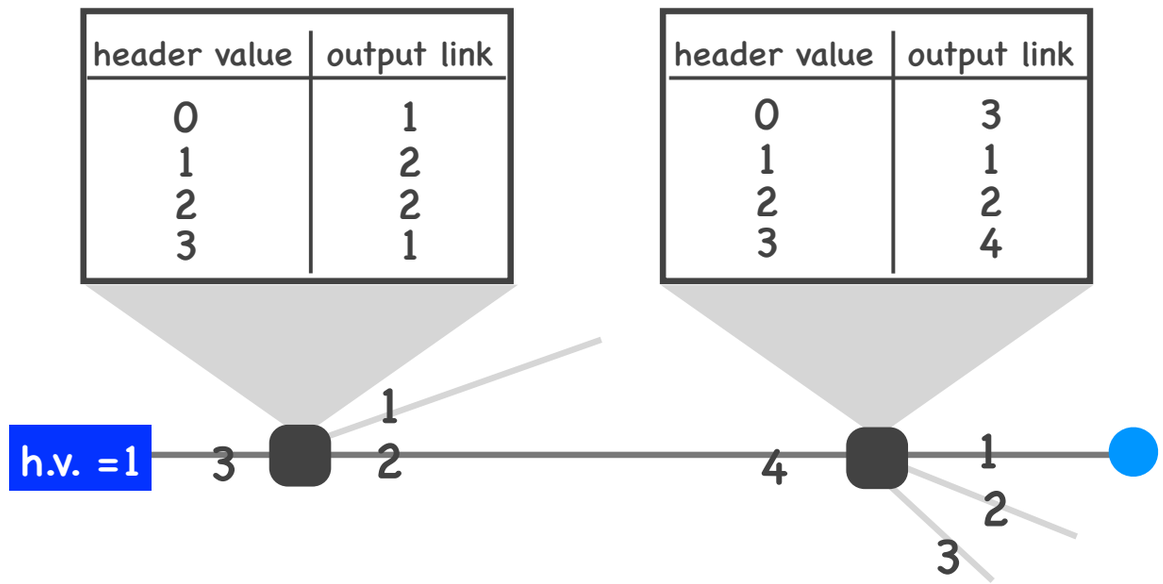
Here is the typical way to do forwarding in a packet-switched network:

- In the network-layer header of each packet, there is a special field that provides information about the packet's destination.
- When Alice sends a packet to Bob, she sets this field to the proper value. In our example, this value is 1.
- Each router that receives the packet reads the value of this field and determines what to do with the packet.
- This happens at each router on the path, until the packet reaches its destination.

Let's make this more specific:



Each router maintains a special data structure called a “forwarding table.”



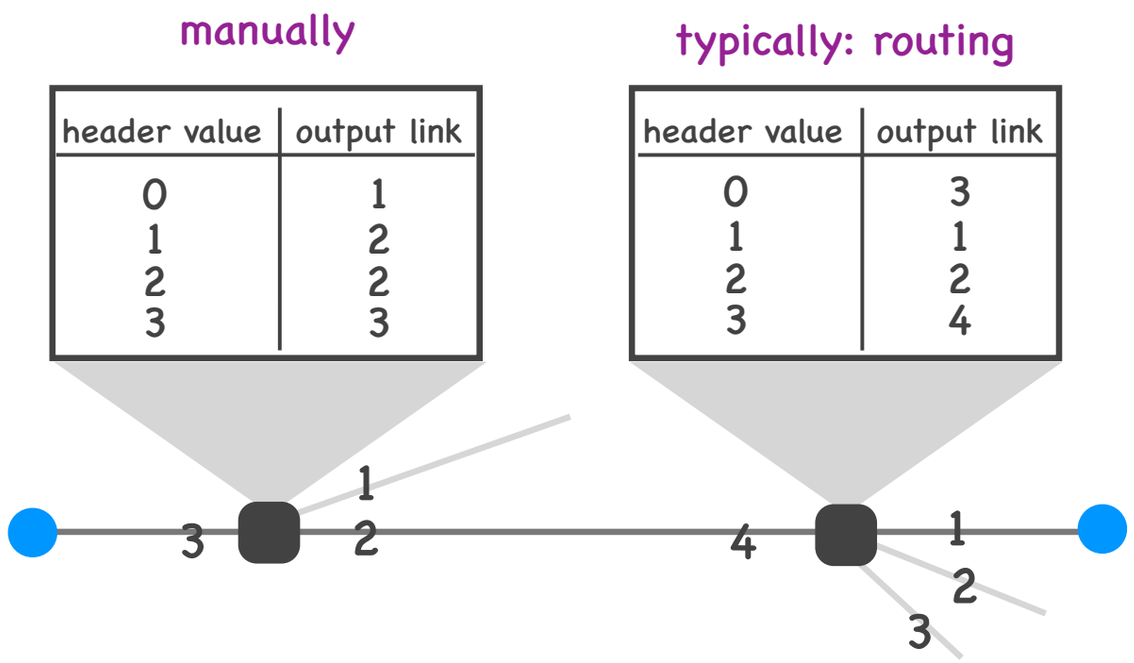
Moreover, each router assigns a number to each of its adjacent links.
For example, the router on the left has links #1 to #3, while the one on the right has links #1 to #4.

Inside the forwarding table, each router stores a mapping
from every possible value of the special network-layer header field mentioned on the previous slide to the correct output link.

In our example, the first router maps header value 1 to output link 2, while the second router maps header value 1 to output link 1. Hence, when Alice sends her packet on the network (with header value 1), the first router forwards it through link 2, while the second one through link 1, and then the packet arrives at Bob.

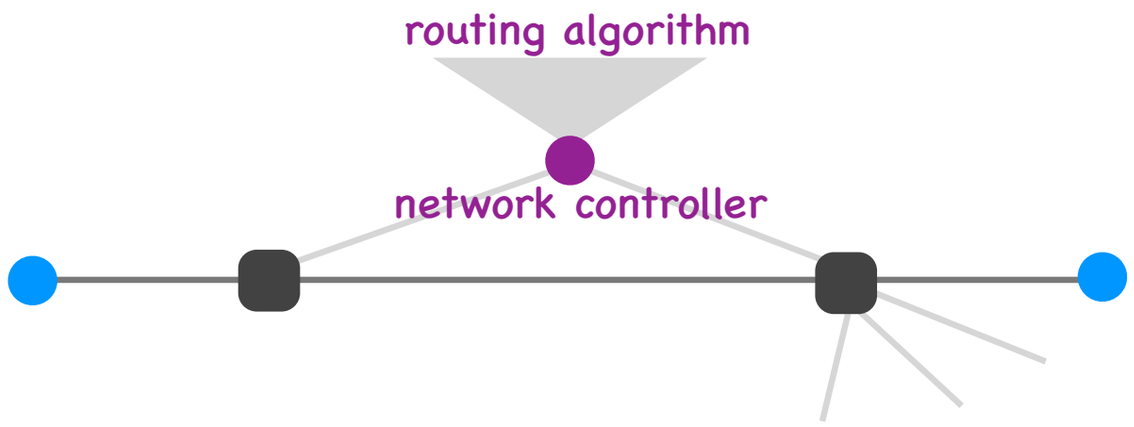
Forwarding

- Local process that takes place at a router and determines output link for each packet
- How: read value from packet's network-layer header, search forwarding table for output link



Who populates the forwarding tables of routers?
E.g., who determines that the router on the left should map header value 1 to output link 2?

Some entries are entered manually by network administrators.
Most entries are entered automatically, as a result of routing processes.

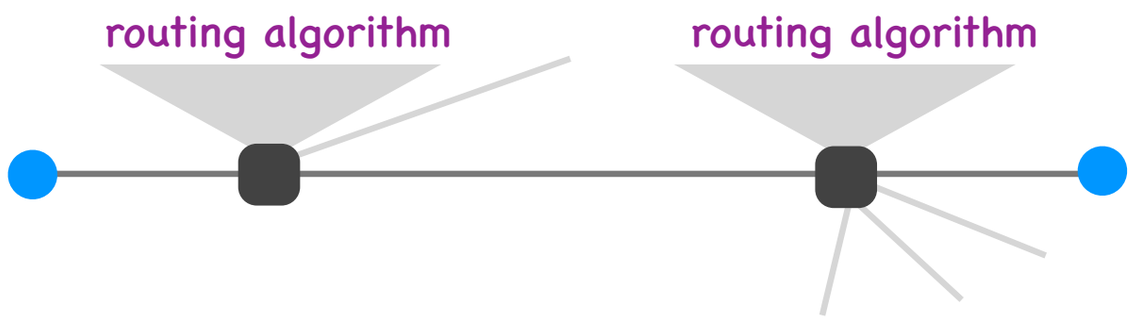


Let’s talk a little bit about routing.

As stated in the previous slide, the goal of routing is to populate the forwarding tables of routers.

One option is to connect all the routers in the network to a logically centralized device, which we will call the “network controller,” and run, on that device, a centralized routing algorithm.

A network controller has a global view of the network, i.e., it knows where all the routers are and how they are interconnected, hence it can determine what each forwarding table should contain and communicate this information to each router.



Another option is to run routing algorithms on the routers themselves. I.e., the routers exchange information with each other and collaborate to compute on their own what their forwarding tables should contain.

This is what mostly happens on the Internet today.

Routing

- Network-wide process that populates forwarding tables
- How: routing algorithm run on a logically centralised network controller or the routers themselves

Forwarding vs. routing

- Forwarding determines output link for each packet
- Routing populates forwarding tables

To conclude this first part, a network layer provides at least two basic functions: forwarding and routing. Forwarding relies on forwarding tables, while routing populate the forwarding tables that are needed for forwarding.

Outline

- Network-layer functions
  - forwarding
  - routing
- Network-layer types
  - virtual-circuit networks
  - datagram networks
- IP forwarding
- IP routing

Now let’s talk about the two basic network-layer types: virtual-circuit and datagram networks.

Outline

- Network-layer functions
  - forwarding
  - routing
- Network-layer types
  - virtual-circuit networks
  - datagram networks
- IP forwarding
- IP routing

Now let's talk about the two basic network-layer types: virtual-circuit and datagram networks.

Possible guarantees

- Guaranteed (in-order) delivery
- Guaranteed maximum delay
- Guaranteed minimum throughput
- Security (confidentiality, authenticity)

In the past, we talked about possible guarantees that a network might provide: reliable packet delivery, not more than a maximum packet delay, at least a minimum throughput, various security properties.

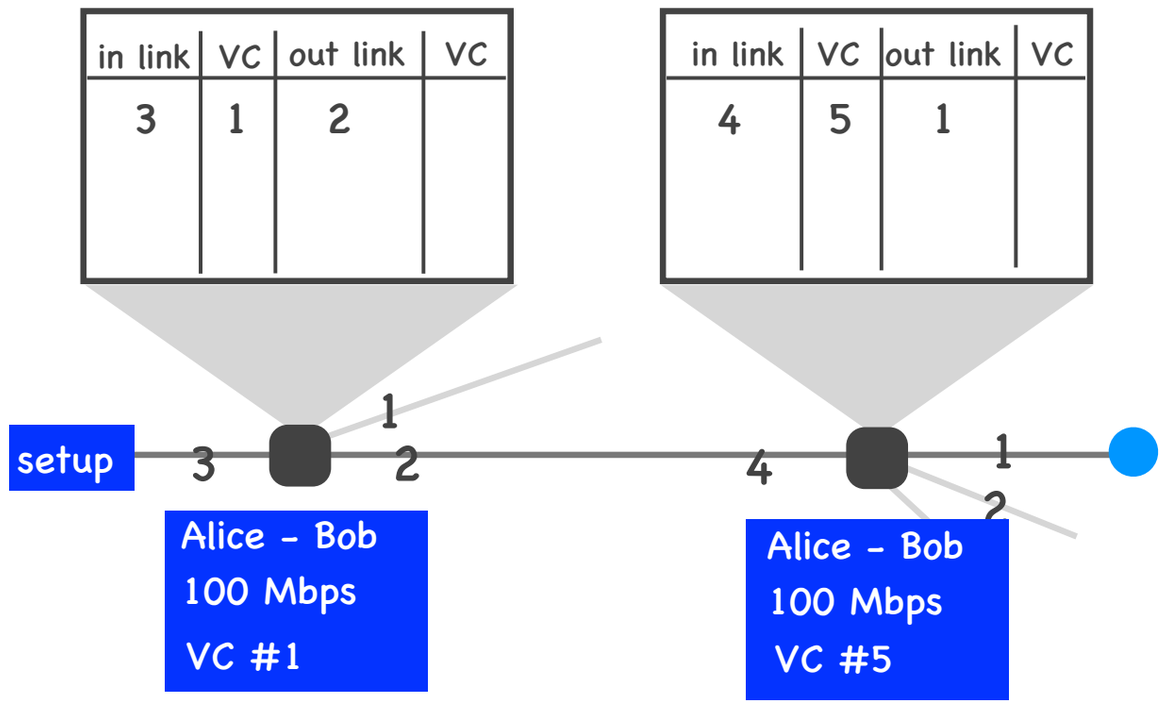
We said that the network layer of the Internet offers none of these guarantees, because that would be hard to do at the network layer, but we never said exactly why.

So, let’s pick…

Possible guarantees

- Guaranteed (in-order) delivery
- Guaranteed maximum delay
- **Guaranteed minimum throughput**
- Security (confidentiality, authenticity)

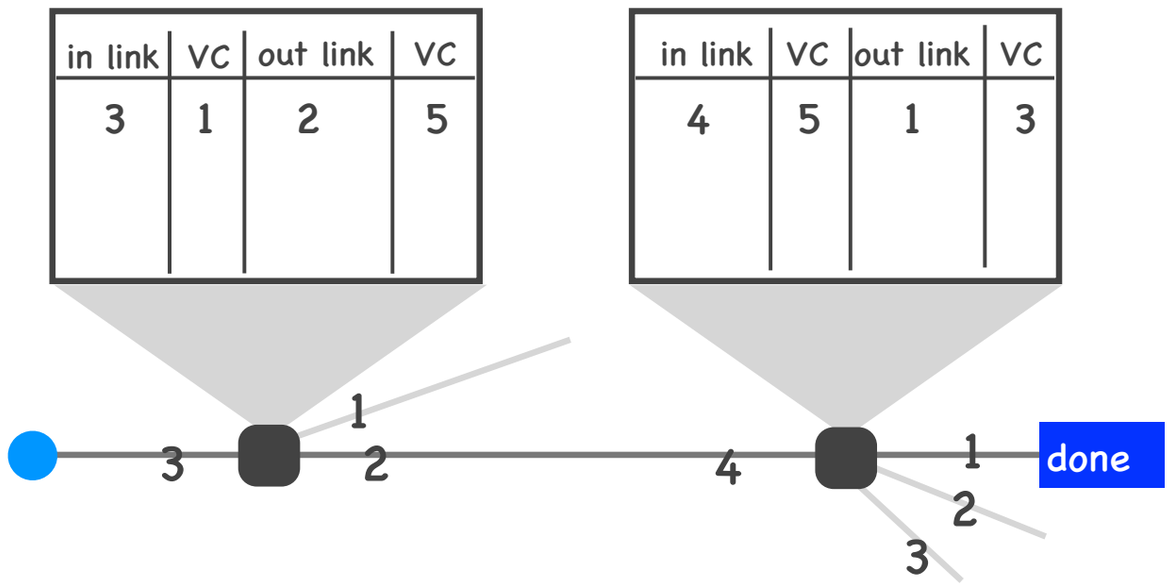
one of the guarantees and discuss, at a very high level, how a network layer could provide it.



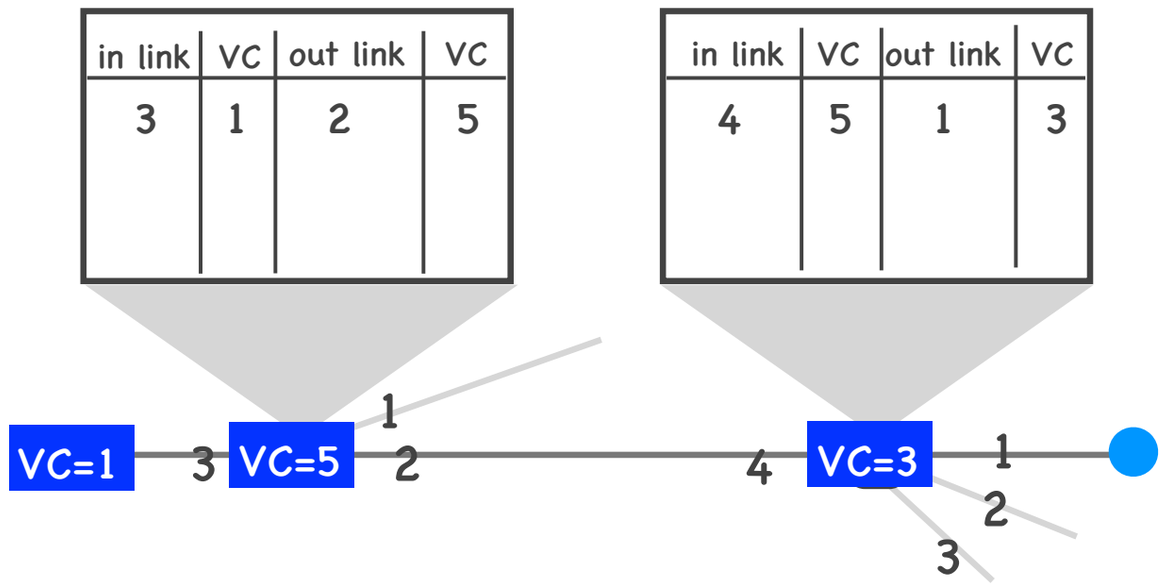
Alice has data to send to Bob, and we want the network layer to guarantee minimum throughout of 100 Mbps from Alice to Bob.

Here is one way this could happen:

- Alice sends a special “connection-setup request” packet to Bob, requesting a “network-layer connection” or a “virtual circuit” with guaranteed minimum throughput 100 Mbps.
- The first router on the path from Alice to Bob receives the request and let’s say that it accepts the request:
  - It allocates enough resources for this network-layer connection to guarantee minimum throughput 100 Mbps.
  - It assigns a “virtual circuit” (VC) number to this connection, e.g., VC #1.
  - It sets up state about this connection, in its forwarding table, specifying that traffic coming from link 3 and belonging to VC #1, should be sent out link 2.
  - It updates the connection-setup request packet with new information, e.g, the VC number, and forwards the request to the next router on the path from Alice to Bob.
- The same thing happens at every router.
- If all the routers on the path from Alice to Bob accept the request, the request eventually reaches Bob.



- Bob sends back a “connection-setup ACK” packet, which goes through all the routers on the path from Alice to Bob (in reverse order). This way, each router learns what VC number the next router from itself has assigned to this connection. For example, the first router leans that the second router has assigned VC #5.
- Eventually Alice receives the ACK, at which point the connection is established.



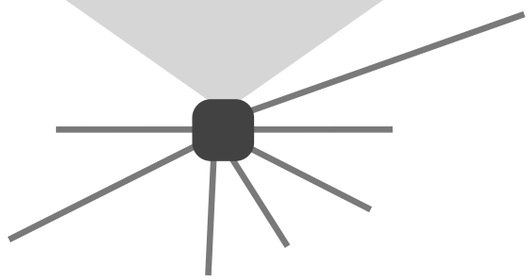
Now it's time for Alice to use the new network-layer connection to send packets to Bob:

- Each of her packets carries, in its network-layer header, a field called "virtual circuit" (VC). She sets this field to 1 (because this is the VC # that the first router assigned to this connection).
- When the packet arrives at the first router, the router reads the value of the VC field from the network-layer header, consults its forwarding table, and determines that it should update the VC field to 5 and send the packet out link 2.
- When the packet arrives at the second router, the router reads the value of the VC field, consults its forwarding table, and determines that it should update the VC field to 3 and forward the packet out link 1.

So, each router has enough information to update the VC field appropriately and determine which is the correct output link for this packet.

millions of concurrent connections

in link	VC	out link	VC
3	1	2	5
1	3	2	2
2	10	5	6
3	3	2	7
5	4	7	3
6	1	7	4



One challenge with this approach is state:

Each router must have an entry in its forwarding table **for each pair of end-systems** whose traffic is currently going through this router.

This can be many entries. E.g., a router in the middle of the Internet may observe concurrent traffic from millions of end-system pairs.

Here's a question for you: why is this a challenge? Why could it be problematic for router, in the middle of the Internet, to keep state on millions of communicating end-system pairs?

There are two issues with this:

- The amount of memory required to keep all this state.
- The security vulnerabilities that it opens up. An attacker could set up lots and lots of network-layer connections and exhaust the forwarding tables of routers. To prevent this, we would need some sort of authentication — e.g., before accepting a connection-setup request, a router would authenticate that the user requesting the connection has the proper credentials. But how are such credentials to be set up across the Internet? It could be done, but it would introduce complexity.

Virtual-circuit network

- Uses connection switching = network-layer connections
- Appropriate for performance guarantees
- Forwarding state: per connection
  - input link, VC #, output link, VC #
  - populated by connection setup (with help from routing)

This type of network that we just saw is called a “virtual-circuit network.”

It essentially performs what we called “connection switching” in the introductory lecture, it creates network-layer connections.

It is useful for providing network-layer performance guarantees.

The challenge is that it requires each router to keep forwarding state per active connection, i.e., per active end-system pair communicating through the router.

Connections & state

- Many different kinds of connections
  - transport-layer (TCP) connections
  - network-layer connections
- Connection => per-connection state
  - at transport layer = at end-systems (OK)
  - at network layer = at all the routers (not OK)
- State => complexity

A parenthesis about connections and state:

The term “connection” is used to describe lots of different situations. E.g., two lectures ago we discussed TCP connections. In this lecture we just discussed network-layer connections, which are a very different thing.

In general, when we say that a set of entities have established a connection we mean that they maintain certain state that allows them to synchronise with each other.

E.g., when Alice and Bob establish a TCP connection, they maintain state for each other: what segment numbers they have sent/received, the sender window, the receiver window, etc. This allows them to communicate using the TCP protocol.

When Alice and Bob establish a network-layer connection (as in the example of the past 5 slides), Alice, Bob, and all the routers on the path from Alice to Bob maintain state for the Alice/Bob pair: minimum throughput, incoming VC and link numbers, and outgoing VC and link numbers.

Maintaining state at the transport layer is OK, because the transport layer is implemented at the end-systems. It is reasonable to expect that Alice maintains state for every end-system that she is communicating with.

Maintaining state at the network layer is much more challenging, because it involves routers that are in the middle of the network, may observe traffic from lots of end-system pairs, and may not be contractually related to any of these end-systems.

Outline

- Network-layer functions
  - forwarding
  - routing
- Network-layer types
  - virtual-circuit networks
  - datagram networks
- IP forwarding
- IP routing

We just talked about virtual-circuit networks, which are useful for providing performance guarantees.

This is not how the Internet works.
The Internet uses a different type of network layer, which is called…

Outline

- Network-layer functions
  - forwarding
  - routing
- Network-layer types
  - virtual-circuit networks
  - datagram networks
- IP forwarding
- IP routing

…a datagram network.

Instead of taking about datagram networks in general, I will focus directly on…

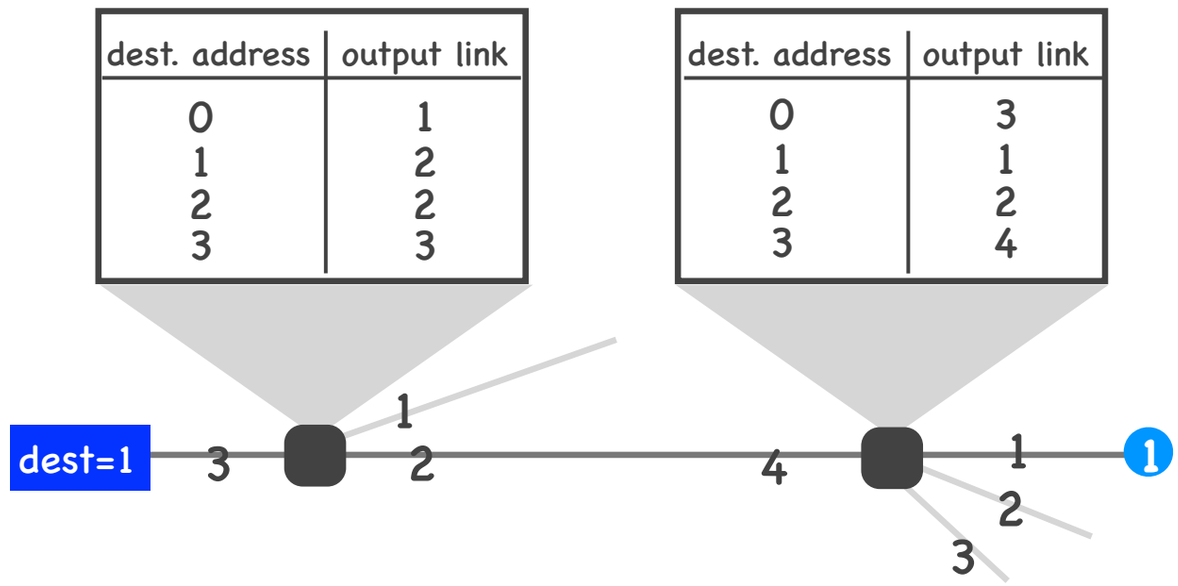
Outline

- Network-layer functions
  - forwarding
  - routing
- Network-layer types
  - virtual-circuit networks
  - datagram networks
- **IP forwarding**
- IP routing

Best-effort delivery

As we already know from prior lectures, the network layer of the Internet provides merely best-effort delivery, which means that it does it best to deliver a packet to its destination, without making any promises whatsoever.

This sounds easier than providing performance guarantees. Let’s see how much easier.



Each end-system has an address, which is called IP address (as we have already seen in previous lectures).
Let's say our destination has IP address #1 (which is not really a valid IP address, but we will pretend it is, to keep the picture simple).

The forwarding tables, instead of storing information about VCs, they store information about IP addresses.

For instance, the second router knows that the end-system with IP address 1 can be reached through output link 1. The first router knows that the end-system with IP address 1 can be reached through output link 2. Etc.

Each router knows how to reach every single IP address in the world.

When an end-system wants to send a packet, it writes the IP address of the destination end-system on the packet's network-layer (IP) header.

Each router that receives the packet reads the destination IP address from the network-layer header, and finds from the forwarding table which is the right output link.

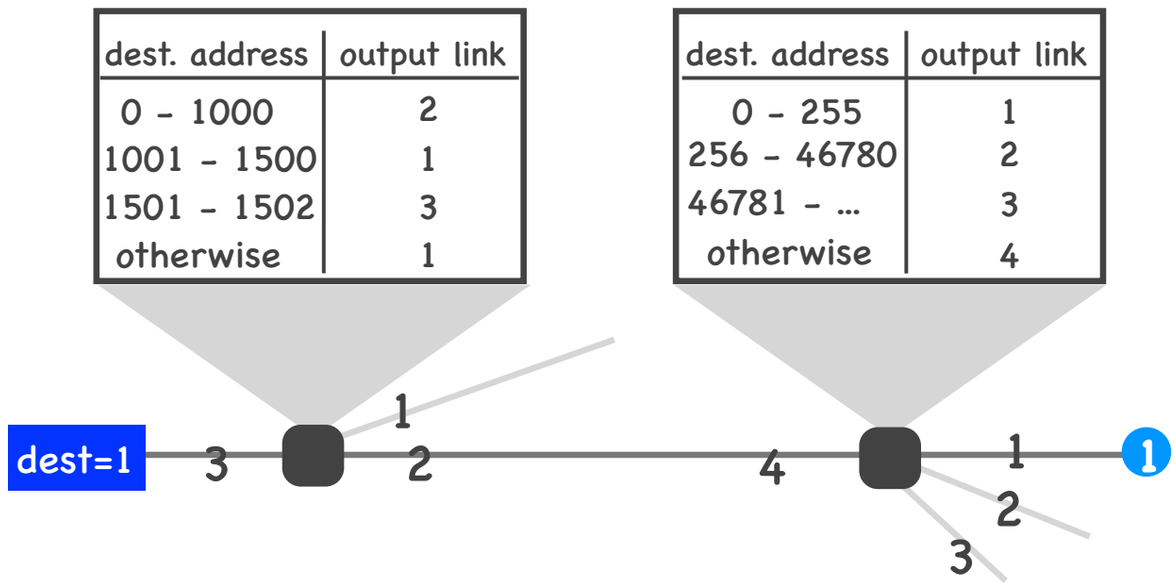
—

I said previously that virtual-circuit networks are bad, because they require network-layer devices to maintain a lot of state.

But what I am describing now does not seem any better.

There are about 4 billion IP addresses in the world.
Can a router maintain, in its forwarding table, an entry for each one?

Fortunately, it does not have to.



Each router actually maintains, in its forwarding table, a mapping from **ranges** of destination IP addresses (not individual IP addresses) to output links.

For example, the router on the left knows that all packets with destination IP address from 0 to 1000
must be sent out link 2.

When a packet arrives at the router, the router reads the destination IP address from the network-layer header, finds the range to which this IP address belongs (in our example, it is the first range in the forwarding table), and determines the right output link.

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

0100

Let’s see in a bit more detail how a router finds the right range for each destination IP address.

Let’s pretend that there are only 16 IP addresses in the world, from 0 to 15.

I am showing here a forwarding table with 4 entries, i.e., 4 ranges of IP address.

In the second column, I am showing the first and last IP address of each range in binary form.

In the third column, I am showing how an IP router typically represents each range.

For example, focus on the first row — IP addresses from 0 to 3.
These are all the IP addresses that, in their binary form, start with 2x 0’s.
Hence, we can represent this range as 00**, where * can take value either 0 or 1.

dest. address range			output link
0 - 2	0000 - 0010	00**	1
3	0011	0011	2
4, 6, 7	0100, 0110, 0111	01**	3
5	0101	0101	2
8 - 15	1000 - 1111	1***	4

0000

But what if we cannot nicely group IP addresses as in the previous example?

For instance, consider the forwarding table on this slide:

- Packets with destination IP address from 0 to 2 should go to output link 1, but packets with destination IP address 3 should go to output link 2.
- Packets with destination IP address 4, 6, or 7 should go to output link 3, but packets with destination address 5 should go to output link 2.
- Etc.

The third column shows how a router would represent this forwarding table: it will still have an entry for 00**, but it will also have a separate entry for 0011. These 2 entries together say: all packets with destination IP address in the range 0 - 3 should go to output link 1, except for packets with destination IP address 3, which should go to output link 2.

dest. address range			output link
0 - 1	0000 - 0001	000*	1
2 - 3	0010 - 0011	001*	2
4 - 7	0100 - 0111	01**	3
8	1000	1000	2
9 - 15	1001 - 1111	1***	4

packets 1000

longest prefix matching

One more example…

A range of contiguous IP addresses that can be represented as a sequence of 0s and 1s followed by a sequence of stars is called an “IP prefix.”

The process by which a router matches a packet’s destination IP address to an entry in its forwarding table is called “longest prefix matching.”

IP / datagram network

- Uses packet switching =
  no network-layer connections
- Appropriate for best-effort service
- Forwarding state: per destination prefix
  - destination prefix, output link
  - populated by routing

This type of network that we just saw is called a datagram network.

It essentially uses what we called “packet switching” in the introductory lectures.

It is appropriate for best-effort service.

It requires that routers maintain forwarding state per destination prefix.

Why does the Internet employ the datagram approach as opposed to the virtual-circuit approach? This relates to the challenges of virtual circuits.

Why the datagram approach?

- It makes forwarding tables smaller
  - no per-connection state in routers
- It makes routers simpler
  - no support for connection setup and teardown in routers

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

In this example,
we have nice convenient ranges,
that we can represent with 4 entries.

dest. address range			output link
0 - 2	0000 - 0010	00**	1
3	0011	0011	2
4, 6, 7	0100, 0110, 0111	01**	3
5	0101	0101	2
8 - 15	1000 - 1111	1***	4

However, in this example,
we have exceptions to each range,
for example, packets with destination IP address 3
must go elsewhere than packets with addresses 0, 1, or 2.

This results in more entries in the forwarding table.

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

The more exceptions we have, the larger the forwarding table.

This is why it is important that IP addresses that belong to the same range
correspond to end-systems that are mostly located in the nearby geographical areas.

Suppose all EPFL IP addresses belong to the same IP prefix, so all routers can represent them with a single forwarding entry.

Now suppose each of us takes their laptop and we go to various places in the world, and we keep our EPFL IP addresses, the routers will need to be populated with our exception entries, so that they can route our traffic separately from the rest of the EPFL traffic.

If many people do this, we will end up with impractically large forwarding tables.

Location-dependent addresses

- Address embeds location information
  - address proximity implies location proximity
- Significantly reduces forwarding state
  - per destination prefix
  - (otherwise, it would be per destination)

So: IP addresses are location-dependent, meaning that an IP address embeds location information, and address proximity typically implies location proximity.

IP address format

IP address = number from 0 to $2^{32}-1$

11011111 00000001 00000001 00000001

223 . 1 . 1 . 1

Now a few words about how to represent IP addresses and IP prefixes.

An IP address is a number from 0 to 2^32-1.

We could represent it in its binary format.

We typically represent in what we call the “dot format”, where we translate each sequence of 8 bits, or 1 byte, to its decimal format.

IP prefix format

IP prefix = range of IP addresses

223.1.1.0 / 24 ← mask

11011111 00000001 00000001 00000000

11011111 00000001 00000001 ********

223.1.1.*

An IP prefix is a range of IP addresses.

We typically represent it as an IP address followed by a / and an integer that is called a “mask.” If an IP prefix has IP address A and mask M, it covers all IP addresses whose M most significant bits are the same as A’s.

E.g., consider IP prefix 223.1.1.0/24. It covers all IP addresses whose 24 most significant bits are the same as 223.1.1.0’s.

So, we can write 223.1.1.0 in its binary format, keep the 24 most significant bits, and turn the rest into *s, which means they could be 0s or 1s. And that’s another way to represent the same prefix.

Alternatively, we can represent the same IP prefix as 223.1.1.*.

IP prefix format

IP prefix = range of IP addresses

223.1.1.74 / 24 ← mask

11011111 00000001 00000001 01001001

11011111 00000001 00000001 ********

223.1.1.*

Here is another IP prefix example: 223.1.1.74.

We can write 223.1.1.74 in its binary format, keep the 24 most significant bits, and turn the rest into *s. And that's another way to represent the same prefix.

Alternatively, we can represent the same IP prefix as 223.1.1.*.

So, this IP prefix is the same as the one on the previous slide. The fact that the 8 least significant bits are different is irrelevant, because this prefix concerns the 24 most significant bits.

IP prefix format

IP prefix = range of IP addresses

223.1.1.74 / 24 ← mask

223.1.1.0 / 24

223.1.1.113 / 24

223.1.1.*

In fact, all the IP prefixes shown here are different representations of the same IP prefix (because they all have mask 24, and they all have the same 24 most significant bits).

IP prefix format

IP prefix = range of IP addresses

223.1.1.0 / 8 ← mask

11011111 00000001 00000001 00000000

11011111 ******** ******** ********

223.*.*.*

Here is another example, this one with an 8-bit mask:…

IP prefix format

IP prefix = range of IP addresses

223.1.1.0 / 12 ← mask

11011111 00000001 00000001 00000000

11011111 0000**** ******** ********

from: 11011111 00000000 00000000 00000000
to: 11011111 00001111 11111111 11111111

When the mask is not a multiple of 8, it's a bit harder to visually determine the range of IP addresses covered by an IP prefix. Please don't try it! It's better to be conservative and follow the right steps: write the entire prefix in binary form and identify the smallest and largest IP address that it covers.

IP prefix format

IP prefix = range of IP addresses

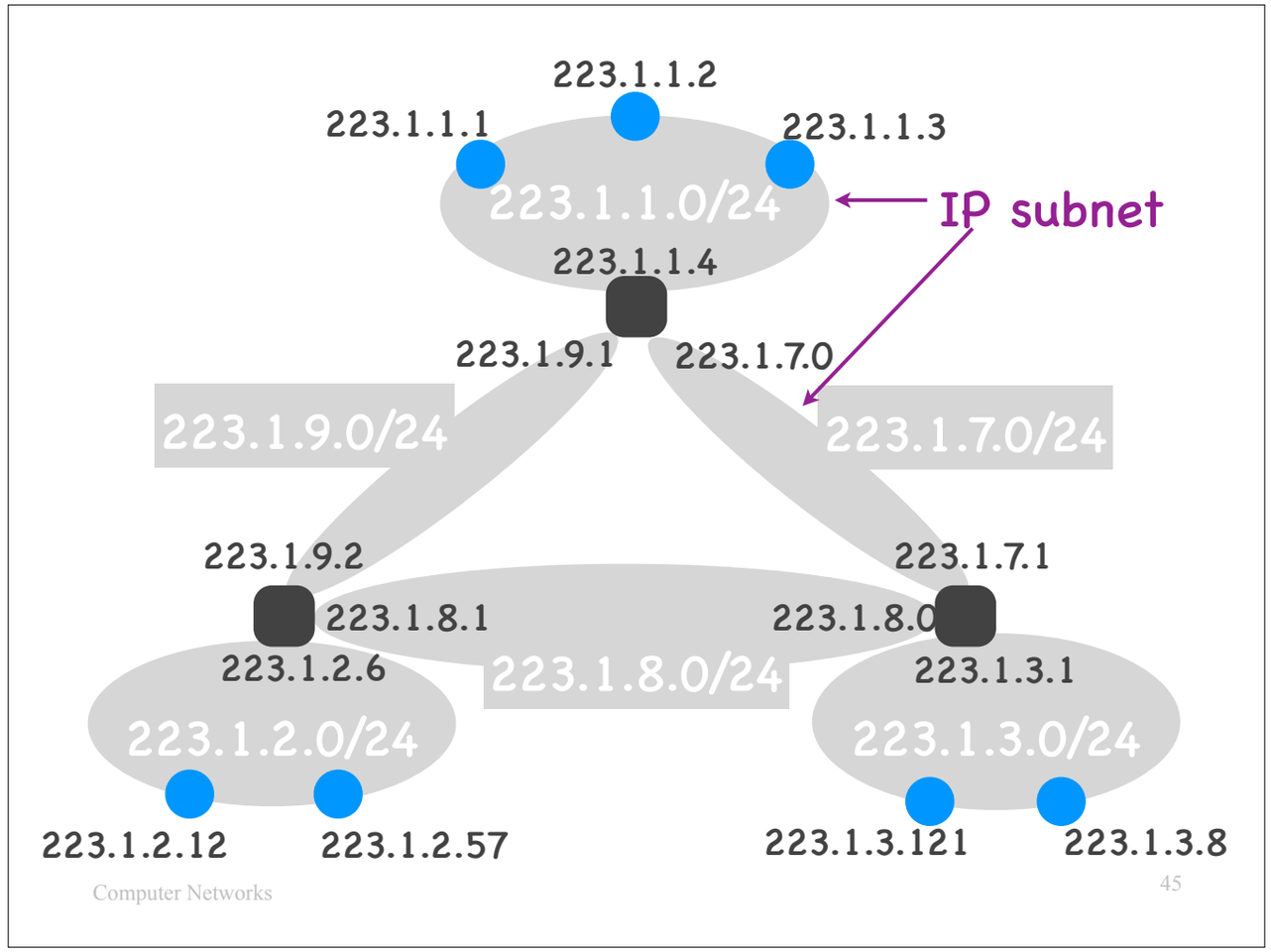
223.1.1.0 / 12 ← mask

11011111 00000001 00000001 00000000

11011111 0000**** ******** ********

from: 223.0.0.0

to: 223.15.255.255



The Internet is organised in “IP subnets” (grey blobs).

An IP subnet contains end-systems (blue circles) and it also has routers (black squares) at its borders with other subnets. An IP subnet does not “contain” any routers inside it.

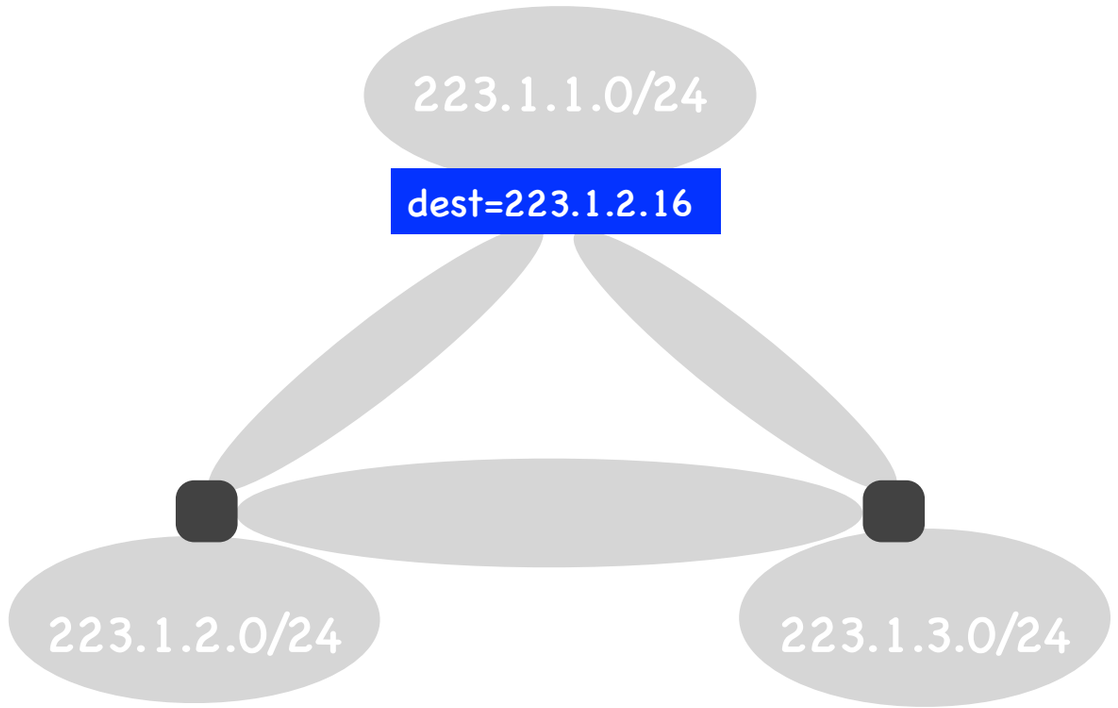
Each IP subnet is assigned an IP prefix.

- All end-systems in the subnet have IP addresses from this prefix.
- Each router that “touches” the subnet has a network interface that belongs to that subnet, which has an IP address from the subnet’s prefix.

E.g., the top-most router on this slide touches 3 subnets and has 1 network interface in each subnet. Each of these network interfaces has an IP address that belongs to the corresponding subnet.

Note that a subnet may include only network interfaces of routers, i.e., no end-systems. This is the case with the 3 subnets in the middle of the slide.

I have not told you yet how exactly the routers are interconnected between them. Two routers might be interconnected through a direct physical link, or through some other means. We will discuss this issue when we explore the link layer of the Internet.



If the top-most router receives a packet with destination IP address 223.1.1.1, it determines that this IP address belongs to one of its local subnets (the one at the top) and forwards the packet that way.

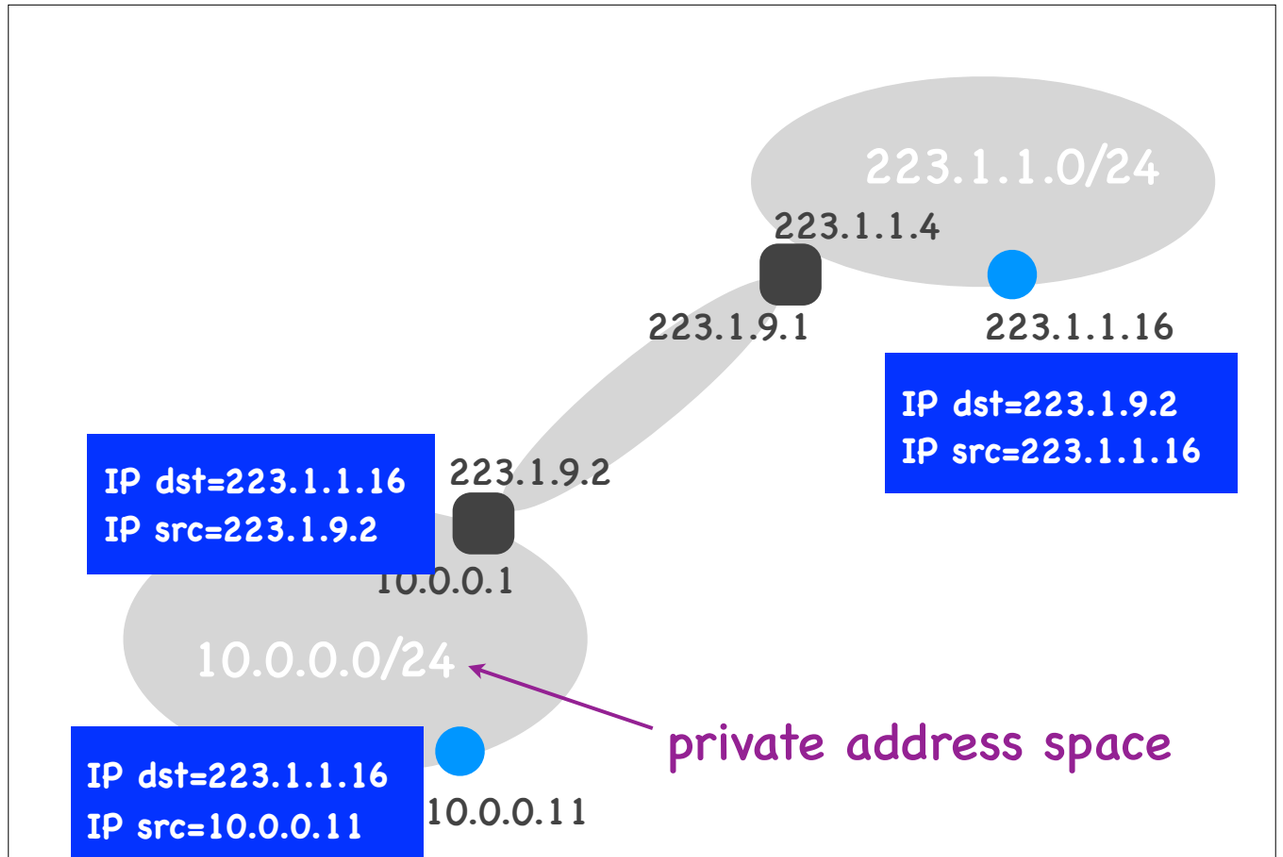
If it receives a packet with destination IP address 223.1.2.16, it determines that this IP address belongs to a foreign subnet and forwards it toward that subnet.

IP subnet

- (Informal) Contiguous network area that does not "contain" any routers
- All its end-systems and incident routers have IP addresses from the same IP prefix

IP address assignment

- Each organization obtains IP prefixes
  - from its ISP or from a regulatory body
- Network operator assigns IP addresses
  - to router interfaces: manually
  - to end-systems: manually or through DHCP



One problem faced by the current Internet is that we are running out of IP addresses.

One solution is to replace the current version of IP (IPv4) with a new one (IPv6). IPv6 has already been deployed in many areas of the world, but not everywhere.

Another solution is Network Address Translation (NAT), which is illustrated on this slide.

There exist certain IP prefixes that constitute “private IP address spaces.” This means that many different IP subnets may internally use this same IP prefix.

One such IP prefix is 10.0.0.0/24. In the example on the slide, an IP subnet uses this prefix, i.e., all the end-systems in this subnet have IP addresses from this prefix. This is fine when two end-systems inside the subnet want to communicate.

But what happens when an end-system inside the subnet wants to communicate with an end-system outside the subnet?

E.g., end-system with IP address 10.0.0.11 (inside the subnet) wants to communicate with end-system with IP address 223.1.1.16 (outside the subnet).

First of all, notice the router at the border of the private subnet. Notice that it has an “internal” network interface, which is inside the private subnet, and an “external” network interface, which is outside the private subnet.

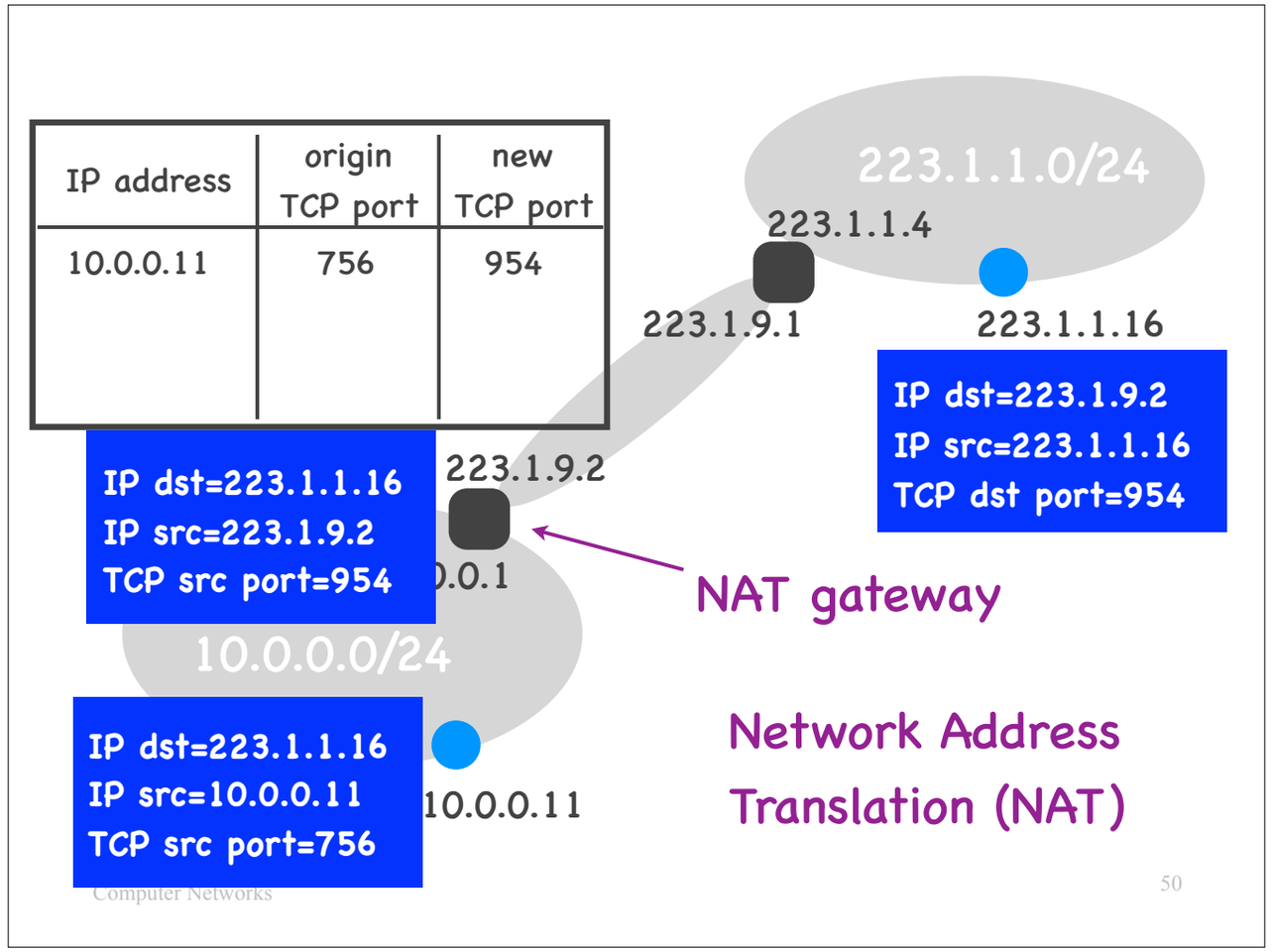
Here’s how it works:

- The source end-system prepares and sends a normal packet.
- The router at the border of the subnet replaces the source IP address with its own external IP address (i.e., its IP address outside the private subnet).
- When the destination end-system receives the packet, it thinks that it came from an end-system with IP address 223.1.9.2, so it responds to that IP

address.

So far so good.

The problem arises when the router at the border of the private subnet receives the response packet. The destination IP address is 223.1.9.2, which means that it could be addressed to any of the end-systems inside the private subnet (or the router itself). How can the router determine which is the right end-system?



When the border router receives a packet coming from the private subnet, it first creates a piece of state, mapping IP address 10.0.0.11 and port number 756 to a new port number 954. Then it rewrites, not only the source IP address to 223.1.9.2; but also the source port number to 954.

Now, when 223.1.1.16 responds, its packet has destination IP address 223.1.9.2 and destination port number 954. When the border router receives the response packet, it maps 954 to the proper original IP address and port number and replaces the packet's destination IP address and port number with these values.

This technique is called NAT, and the border router that does the rewriting and keeps the necessary state is called a "NAT gateway."

—

I have two questions for you:

First: Suppose an end-system that is outside the private subnet wants to initiate communication with a system that is inside the private subnet. Is that possible?

The answer is: It is possible, but complicated, it requires creating a-priori special state inside the NAT gateway. Given that the end-systems that are inside the private subnet do not have global IP addresses, they cannot be normally addressed. So, NAT breaks one of the basic promises of the Internet, namely that any two end-systems connected to the Internet can communicate with each other through the IP protocol, without having to create any state inside the network.

The second question was the following: Earlier in the lecture, I made a fuss about the fact that routers should not participate in network-layer connections, i.e., they should not need to keep per-connection state in their forwarding tables. How is NAT better? Based on what I described, the NAT gateway is very

clearly maintaining per-connection state. Why is this OK?

The answer is: It is OK because the NAT gateway is typically located toward the edge of the Internet, for example, between your home network and the rest of the Internet. So, a private subnet typically includes a relatively small number of end-systems, which would normally not initiate millions of concurrent connections.

Network Address Translation

- Resolves IP address depletion problem
- Requires per-connection state at the border routers of the private IP subnets
- End-systems inside private IP subnets are unreachable from the outside world

…meaning that they can initiate communication with the outside world, but cannot accept communication toward them that is initiated by the outside world.

—

To read, if networks interest you:

- Broadcasting & DHCP
  - Ed. 6, p. 371 | Ed. 7, p. 370
- ICMP & Traceroute
  - Ed. 6, p. 379 | Ed. 7, p. 447
- IPv6
  - Ed. 6, p. 382 | Ed. 7, p. 376