CS168
How the Internet Works:
A bottom-up view

Sylvia Ratnasamy

Fall 2024

Goal for the next few lectures is to give you a broad overview of how the Internet works

- This lecture: bottom-up
  - Identify the fundamental pieces that make up the overall picture
- Next lecture: top-down
  - Identify the important architectural choices involved in the picture together

Today

Today

- How is data transferred across the Internet?

Today

- How is data transferred across the Internet?

- How are network resources shared?

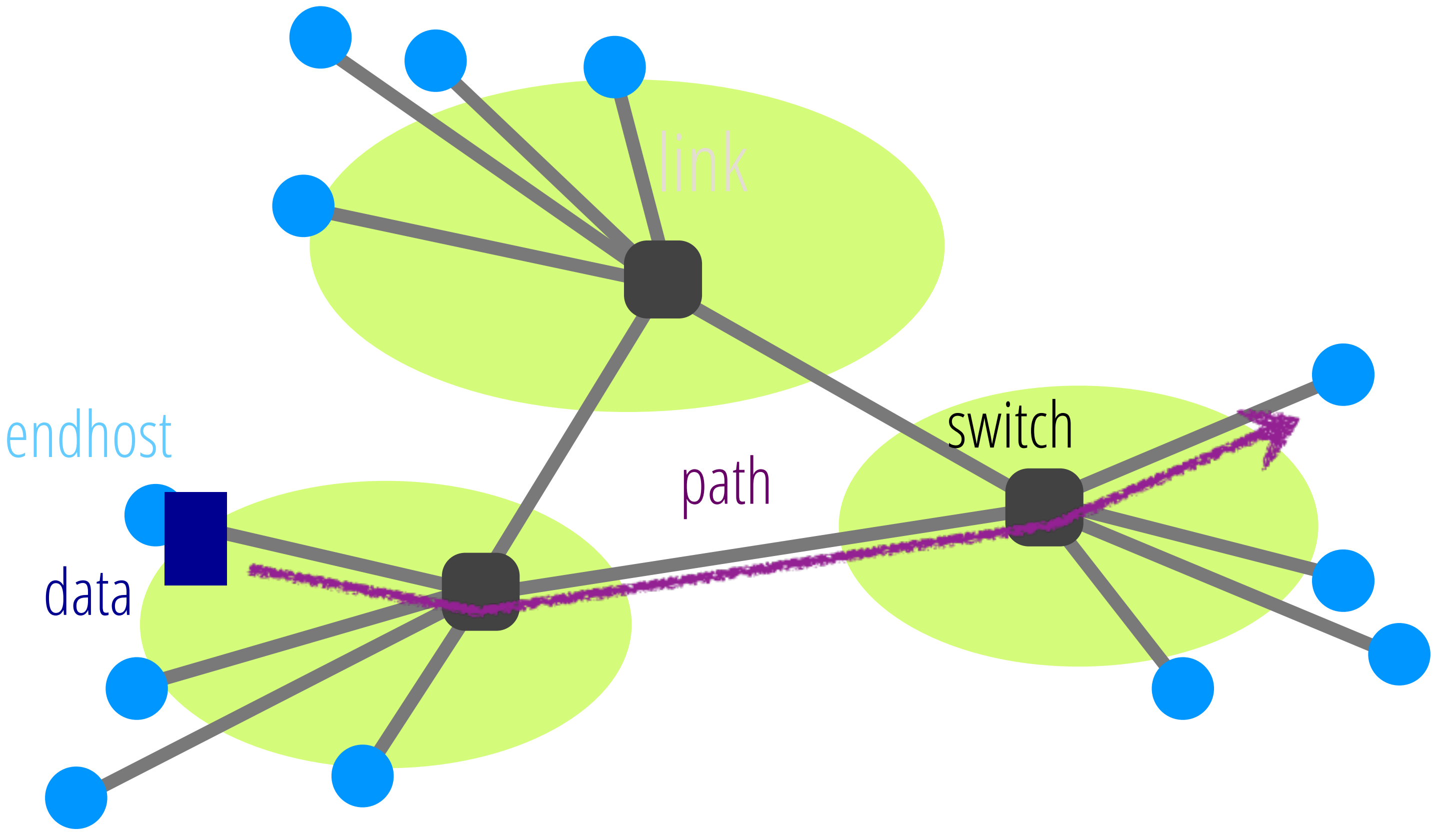
Today

- How is data transferred across the Internet?
- How are network resources shared?
- Start understanding of the “life of a packet” through the network

Today

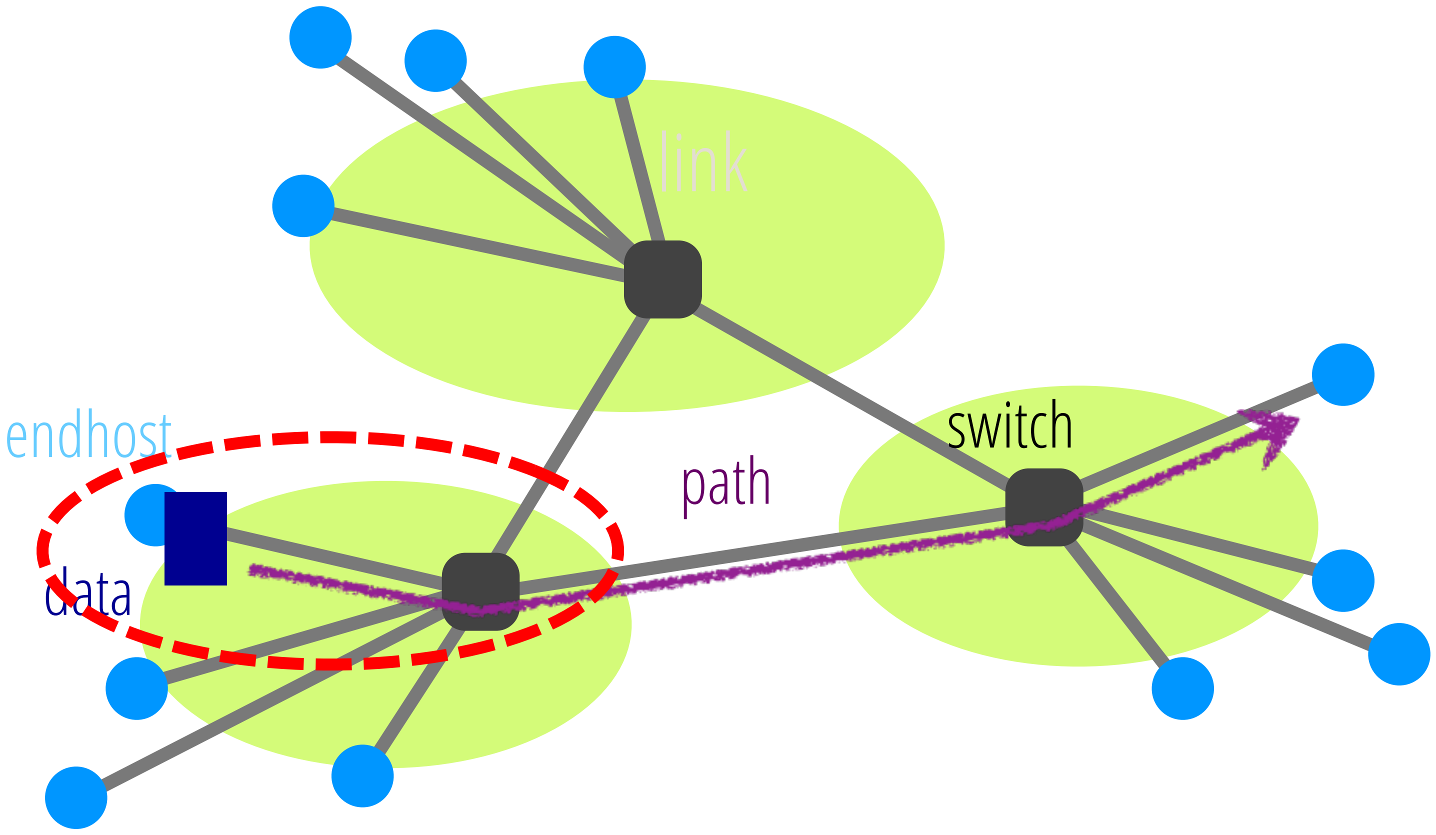
- How is data transferred across the Internet?
- How are network resources shared?
- Start understanding of the “life of a packet” through the network
- Along the way: identify the key topics we’ll be studying this semester

Recall, from last lecture



The goal of the Internet is to transfer data between end hosts

Recall, from last lecture



The goal of the Internet is to transfer data between end hosts

How is data organized (in the network)?

How is data organized (in the network)?

Application data

How is data organized (in the network)?

Application data

“packet”

How is data organized (in the network)?

Application data

"packet"

01000111100010101001110100011001

How is data organized (in the network)?

Application data

"packet"

01000111100010101001110100011001	Header

Metadata that describes how data is to be delivered

How is data organized (in the network)?

Application data

"packet"

Payload	Header

Metadata that describes how data is to be delivered

Recap: packets

Recap: packets

- Packets are a chunk of bits with:
  - Payload: meaningful only to the endpoints
    - Bits from a file, video, etc.
  - Header: meaningful to the network *and* endpoint
    - What information <u>must</u> a header contain?

Recap: packets

- Packets are a chunk of bits with:
  - Payload: meaningful only to the endpoints
    - Bits from a file, video, etc.
  - Header: meaningful to the network *and* endpoint
    - What information must a header contain?

The destination address!

Recap: packets

- Packets are a chunk of bits with:
  - Payload: meaningful only to the endpoints
    - Bits from a file, video, etc.
  - Header: meaningful to the network *and* endpoint
    - What information must a header contain?

The destination address!

- In practice, a packet has multiple headers (next lecture)

Recap: packets

- Packets are a chunk of bits with:
  - Payload: meaningful only to the endpoints
    - Bits from a file, video, etc.
  - Header: meaningful to the network *and* endpoint
    - What information must a header contain?

The destination address!

- In practice, a packet has multiple headers (next lecture)

Recap: packets

- Packets are a chunk of bits with:
  - Payload: meaningful only to the endpoints
    - Bits from a file, video, etc.
  - Header: meaningful to the network *and* endpoint
    - What information must a header contain?

The destination address!

- In practice, a packet has multiple headers (next lecture)

- And communication between a pair of endhosts involves multiple packets
  - "Flow": stream of packets exchanged between two endpoints (more on this later)

Packets on a link

Application data

Packets on a link

Properties of links

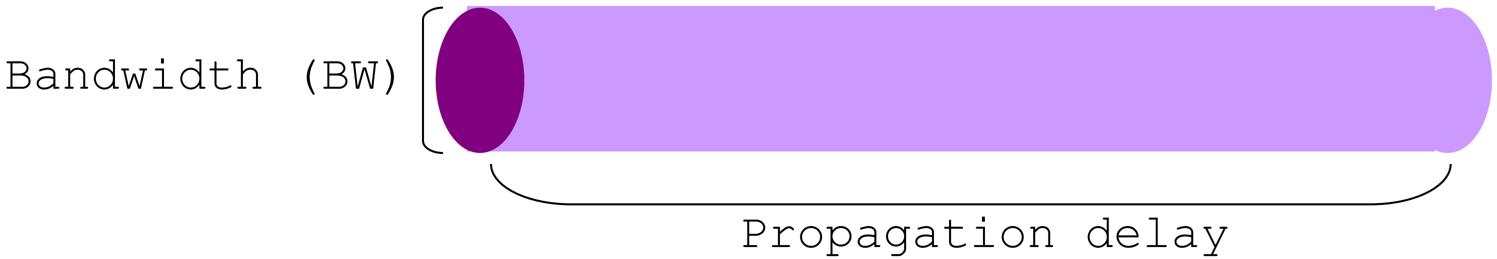
Properties of links

- Bandwidth: number of bits sent (or received) per unit time (bits/second or bps)
    - "width" of the link

Properties of links

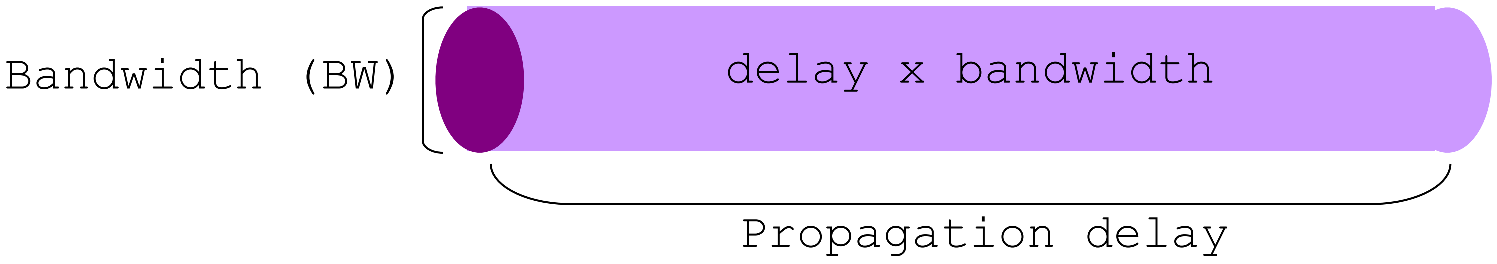
- Bandwidth: number of bits sent (or received) per unit time (bits/second or bps)
  - “width” of the link
- Propagation delay: time it takes a bit to travel along the link (seconds)
  - “length” of the link

Properties of links



- Bandwidth: number of bits sent (or received) per unit time (bits/second or bps)
  - "width" of the link
- Propagation delay: time it takes a bit to travel along the link (seconds)
  - "length" of the link

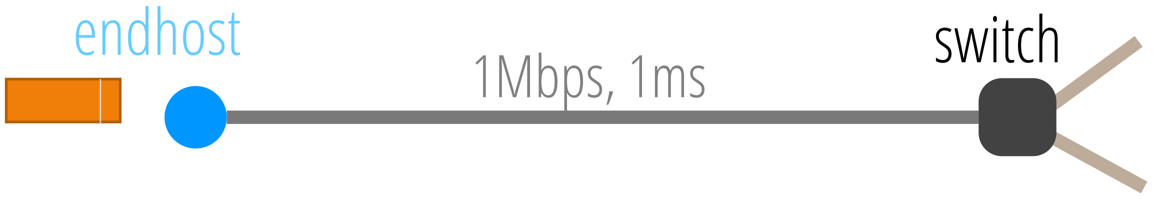
Properties of links



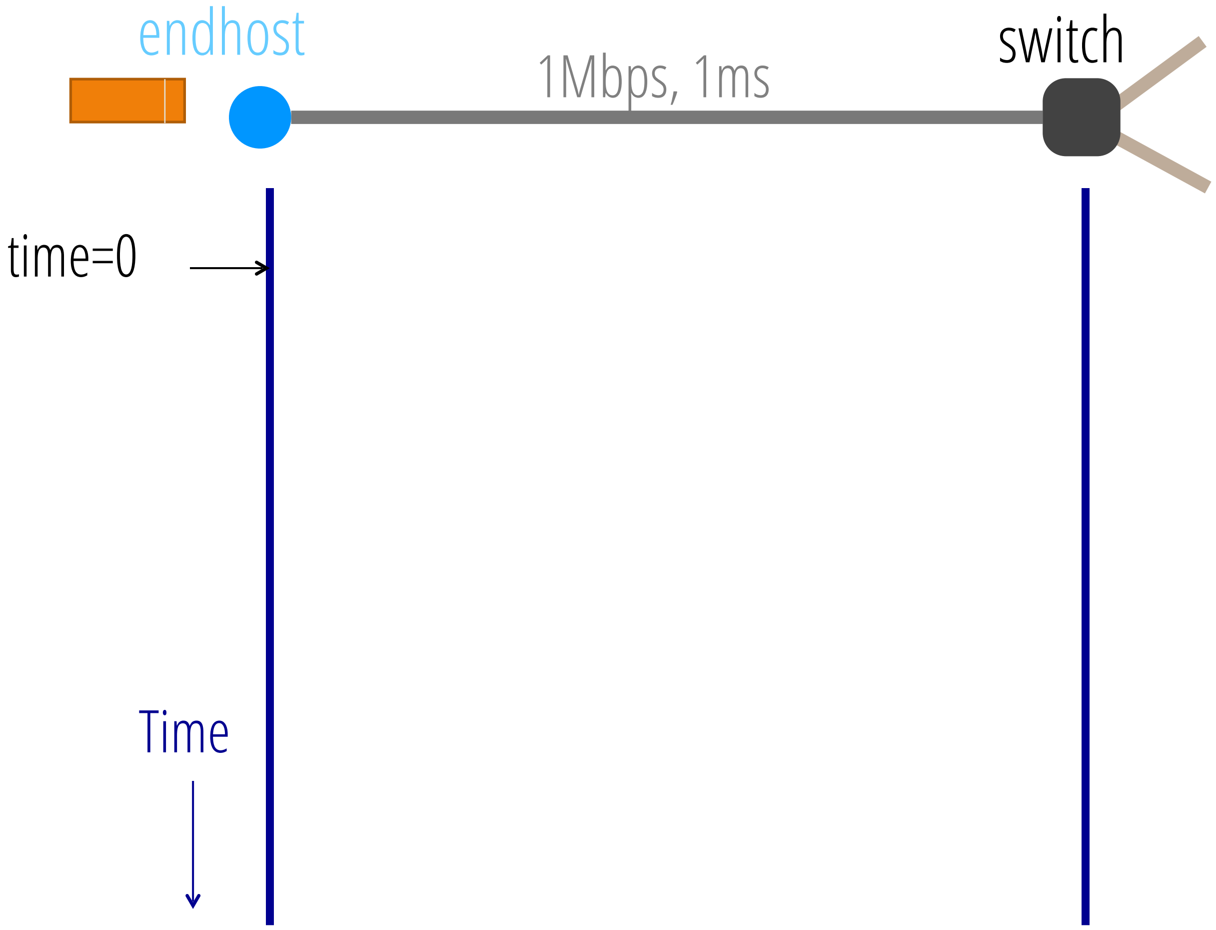
- Bandwidth: number of bits sent (or received) per unit time (bits/second or bps)
  - "width" of the link
- Propagation delay: time it takes a bit to travel along the link (seconds)
  - "length" of the link
- Bandwidth-Delay Product (BDP): bits/time x propagation delay (bits)
  - "capacity" of the link

Packets on a link: sending a 100B packet

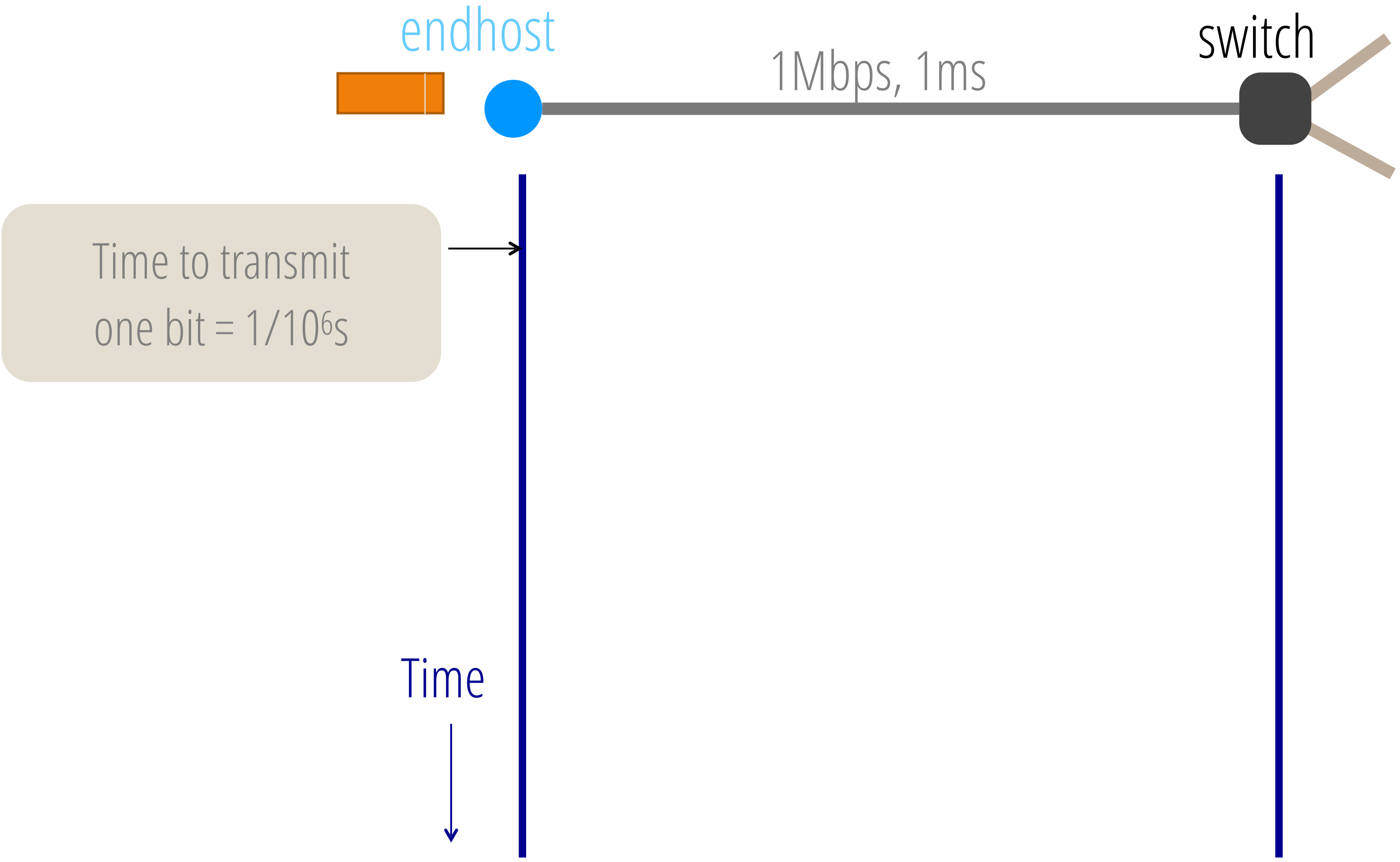
Packets on a link: sending a 100B packet



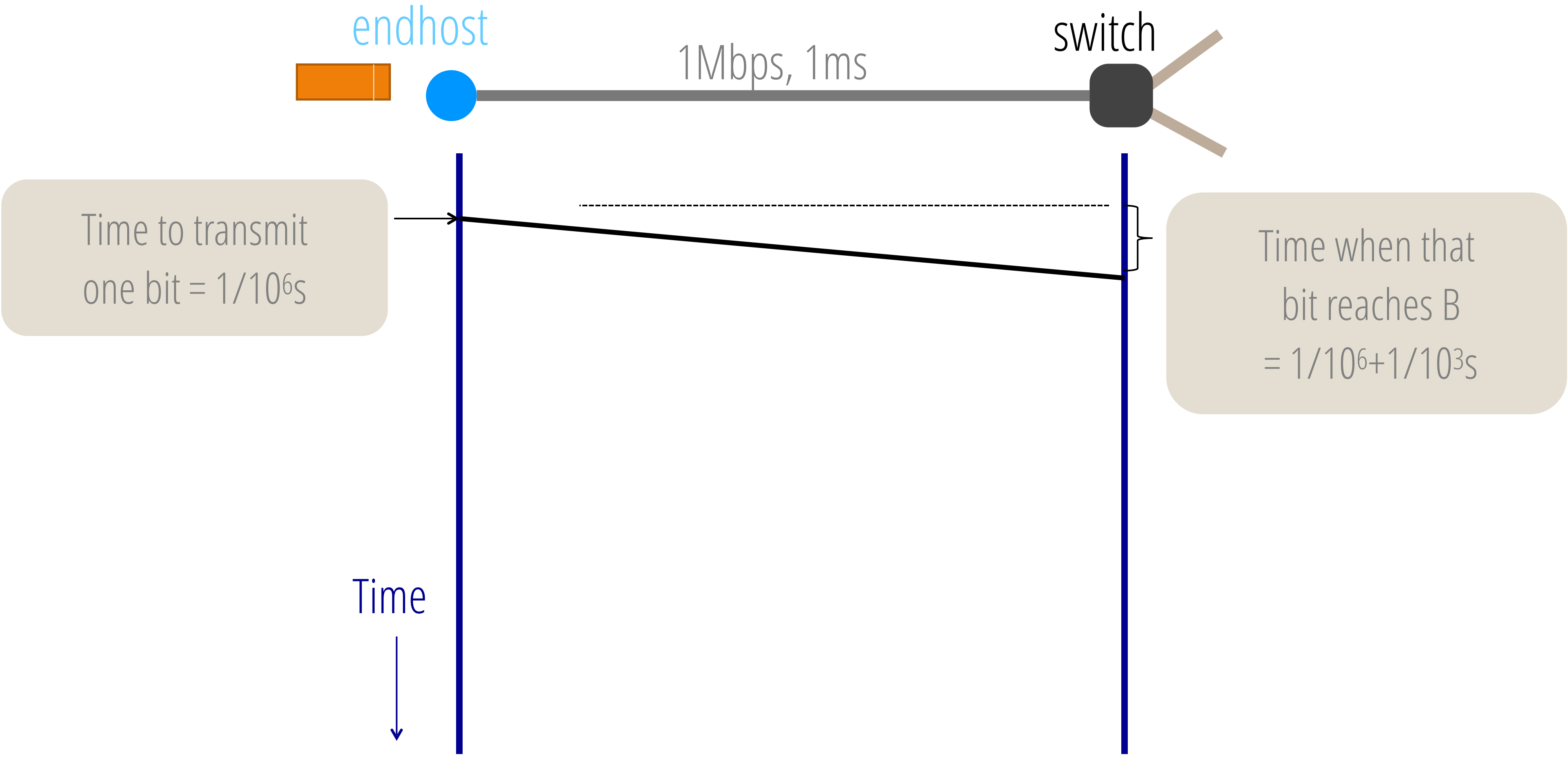
Packets on a link: sending a 100B packet



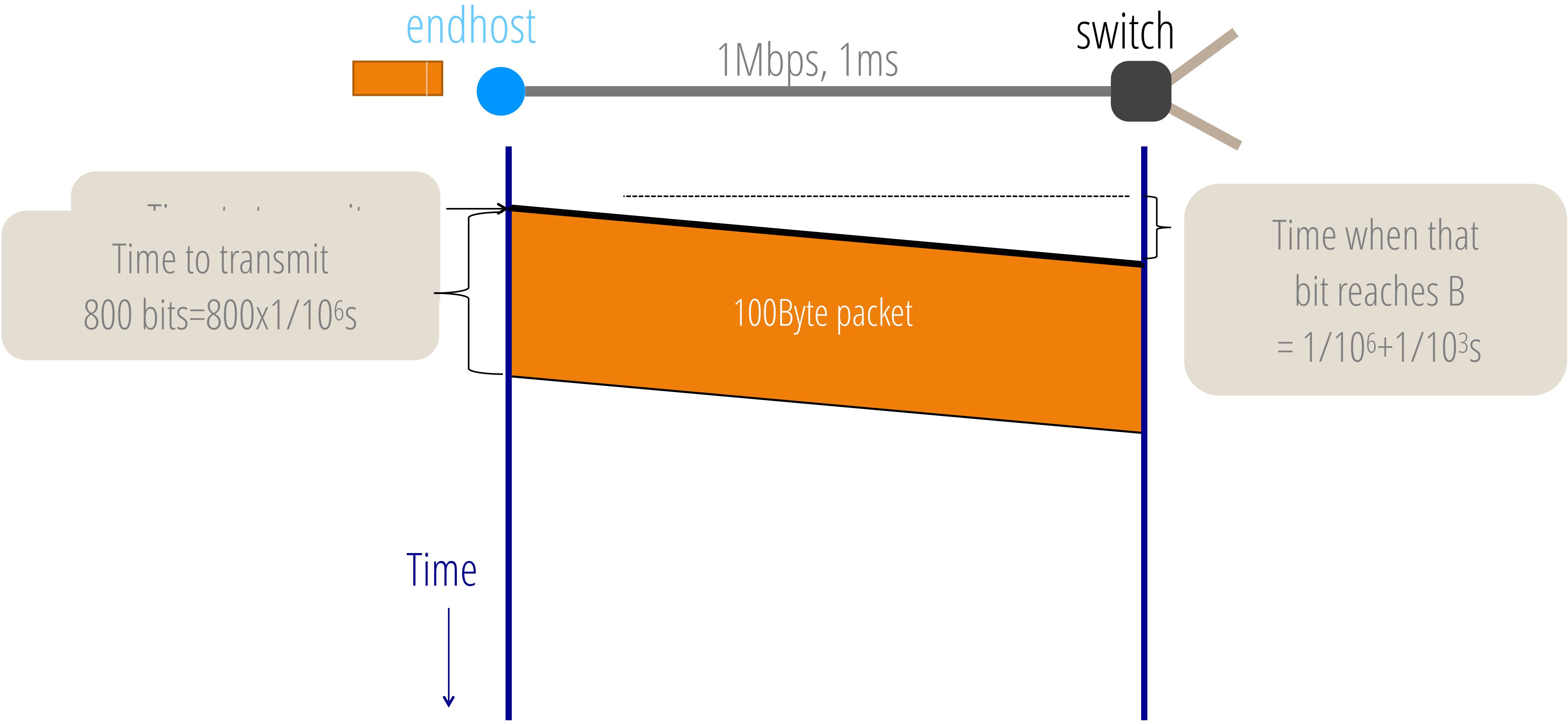
Packets on a link: sending a 100B packet



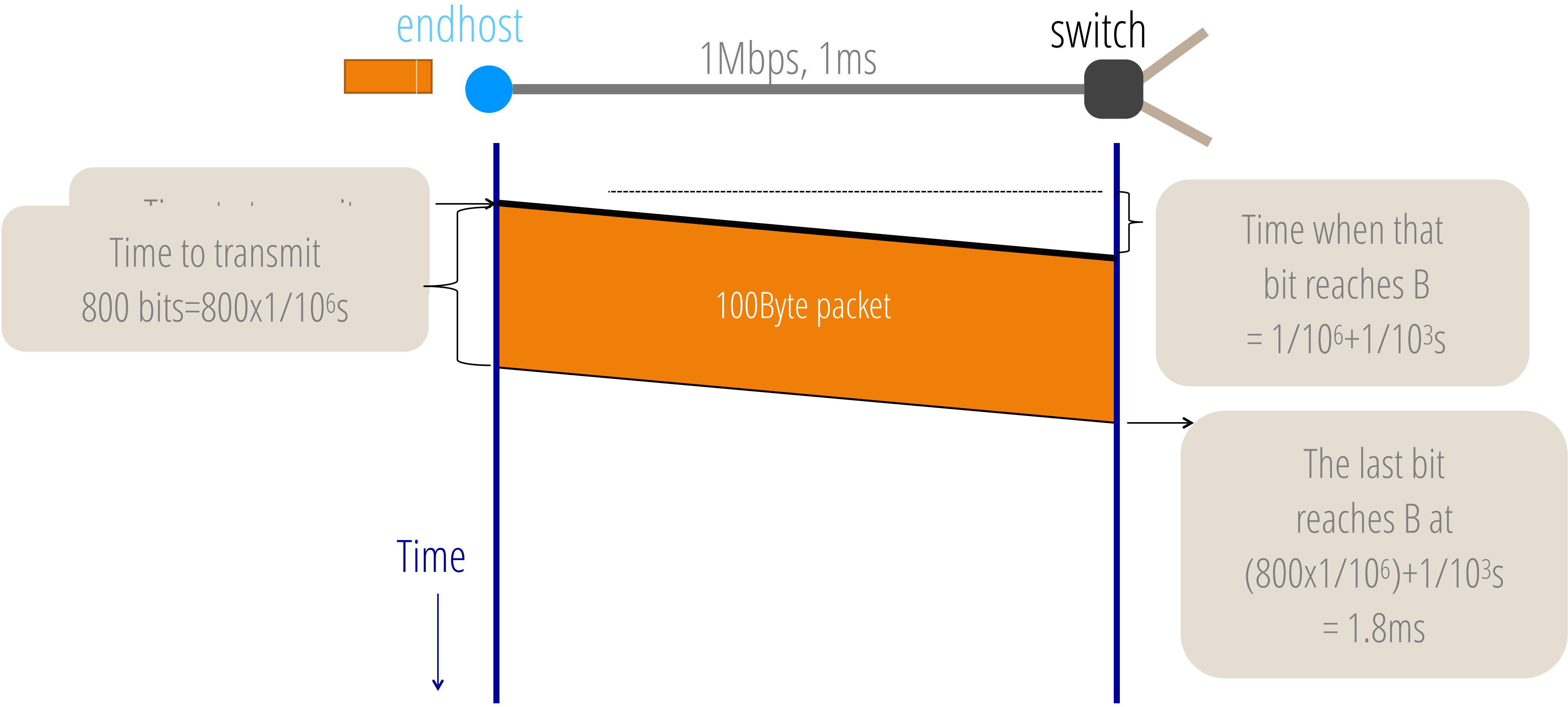
Packets on a link: sending a 100B packet



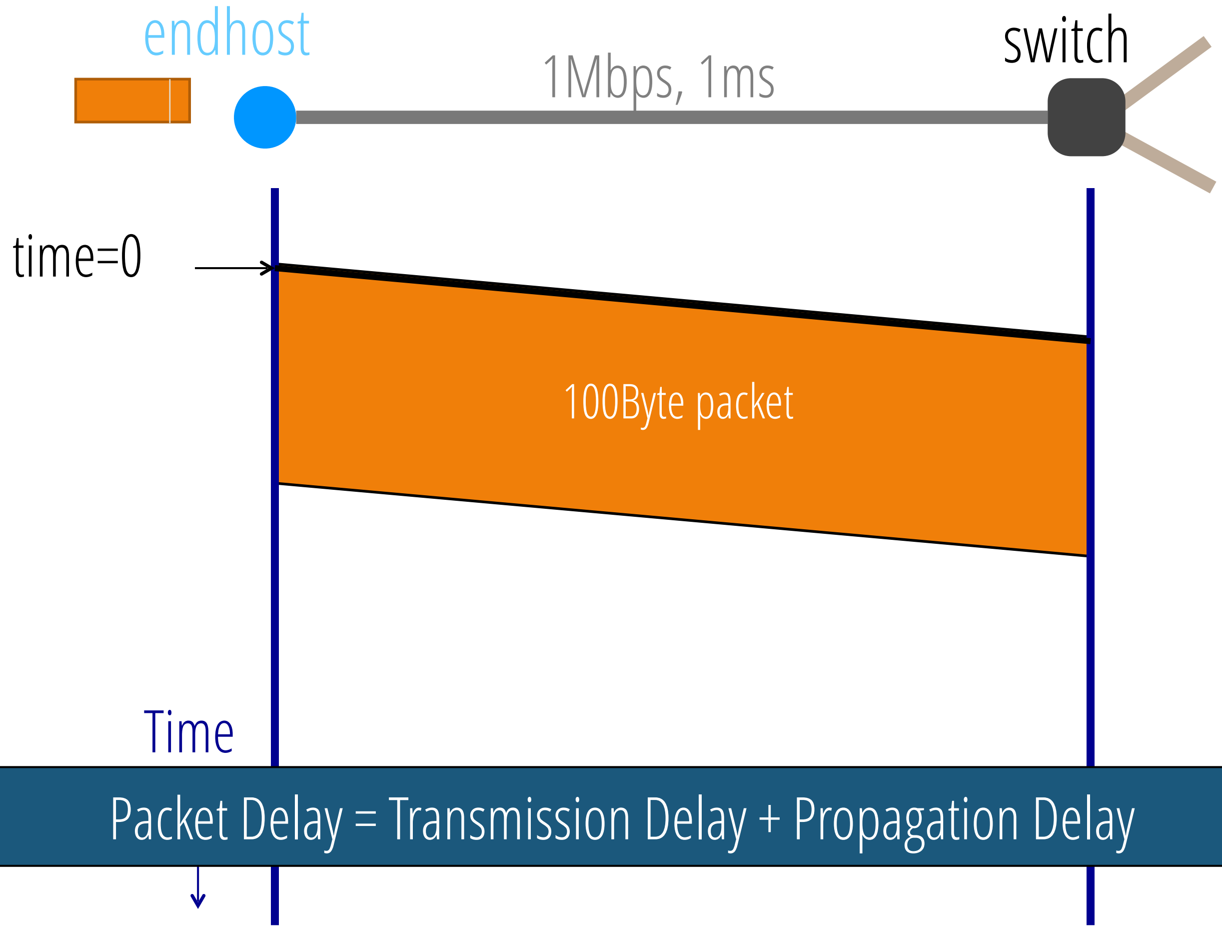
Packets on a link: sending a 100B packet



Packets on a link: sending a 100B packet

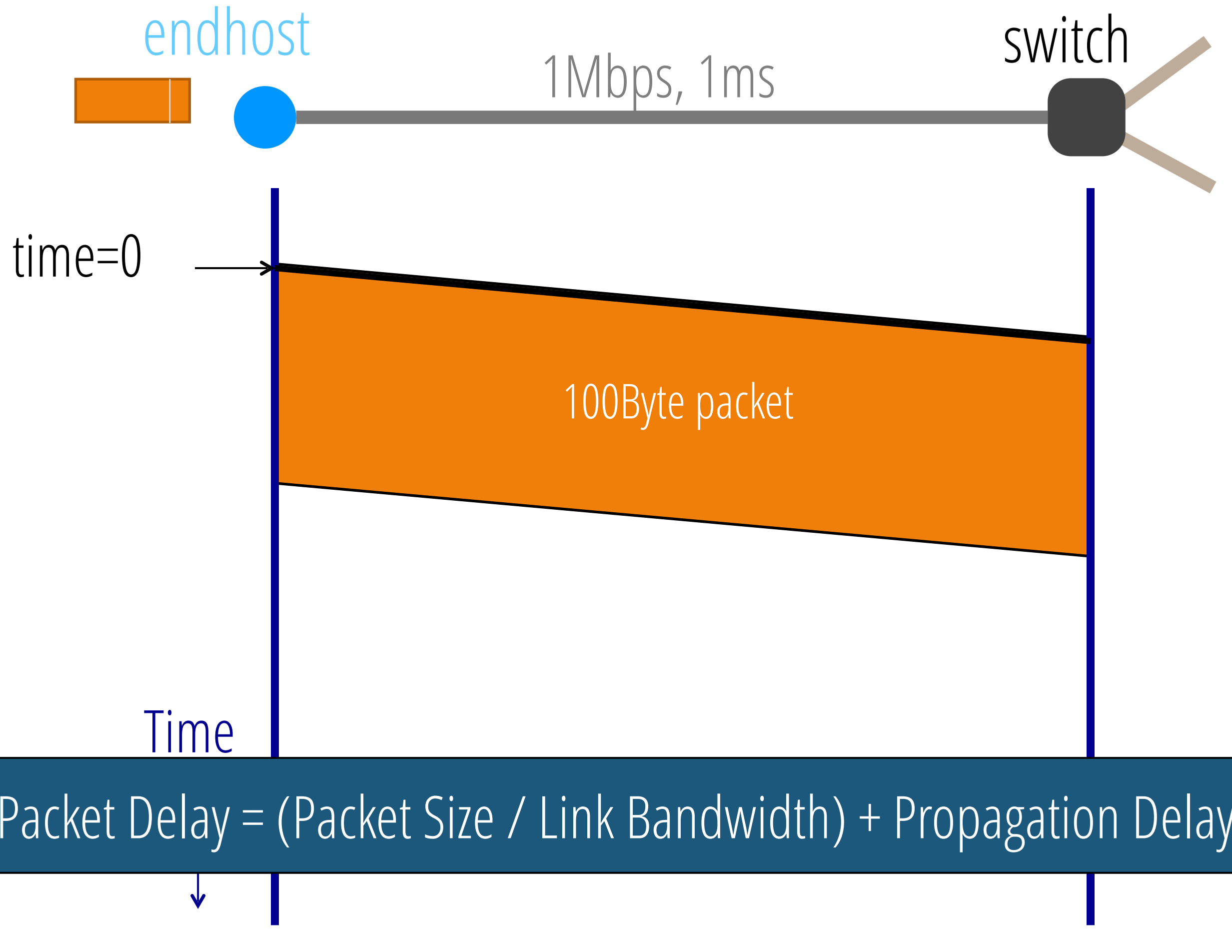


Packets on a link: sending a 100B packet



Packet Delay = Transmission Delay + Propagation Delay

Packets on a link: sending a 100B packet



Packet Delay = (Packet Size / Link Bandwidth) + Propagation Delay

Question: which link is better?

- Link-1: bandwidth=10Mbps and propagation delay = 10ms
- Link-2: bandwidth=1Mbps and propagation delay = 1ms

Sections will cover packet delay calculations in detail

Question: which link is better?

- Link-1: bandwidth=10Mbps and propagation delay = 10ms
- Link-2: bandwidth=1Mbps and propagation delay = 1ms

- Packet delay for a 10B packet:
  - With link 1: ~10ms
  - With link 2: ~1ms

Sections will cover packet delay calculations in detail

Question: which link is better?

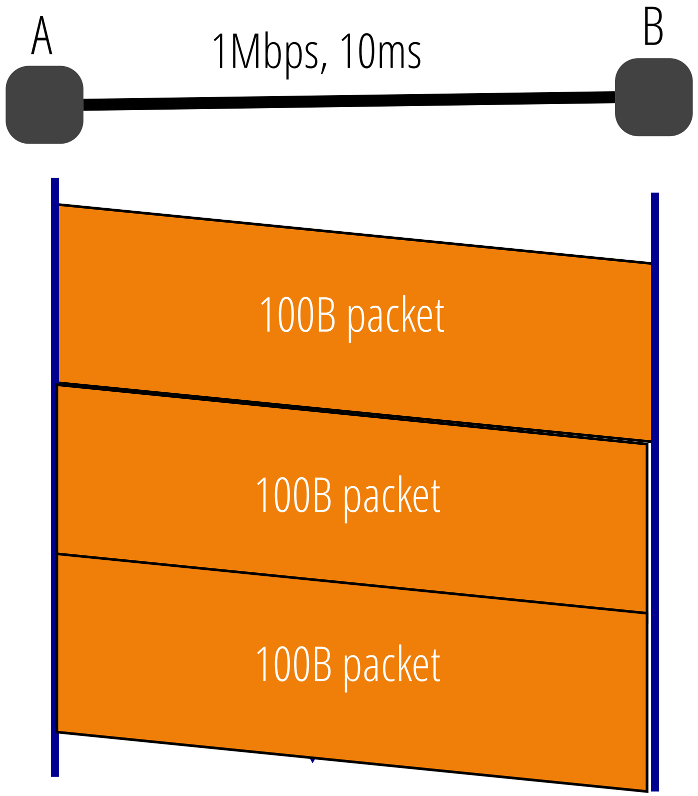
- Link-1: bandwidth=10Mbps and propagation delay = 10ms
- Link-2: bandwidth=1Mbps and propagation delay = 1ms

- Packet delay for a 10B packet:
  - With link 1: ~10ms
  - With link 2: ~1ms

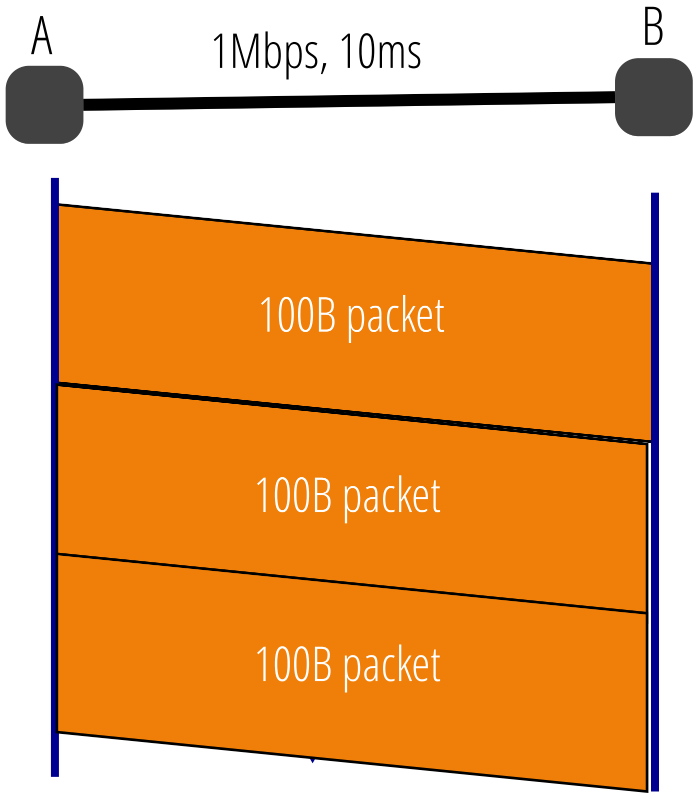
- For a 10,000B packet:
  - Link 1: ~18ms
  - Link 2: ~81ms

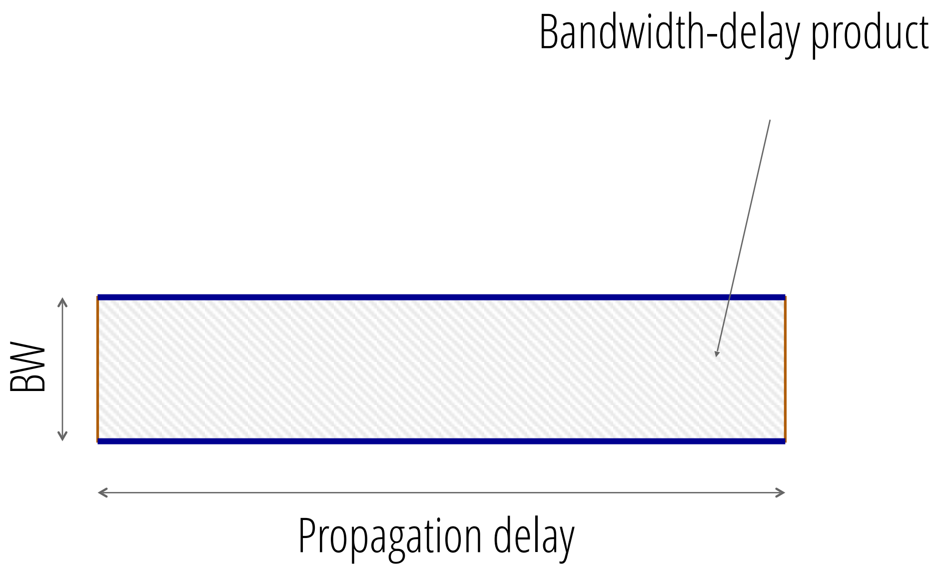
Sections will cover packet delay calculations in detail

Packets on a link: an alternate "pipe" view

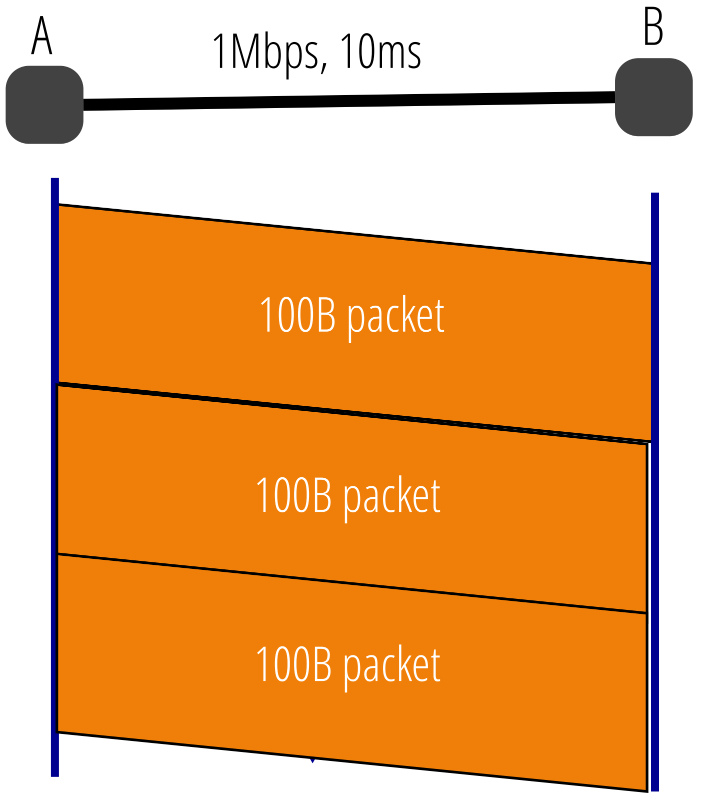


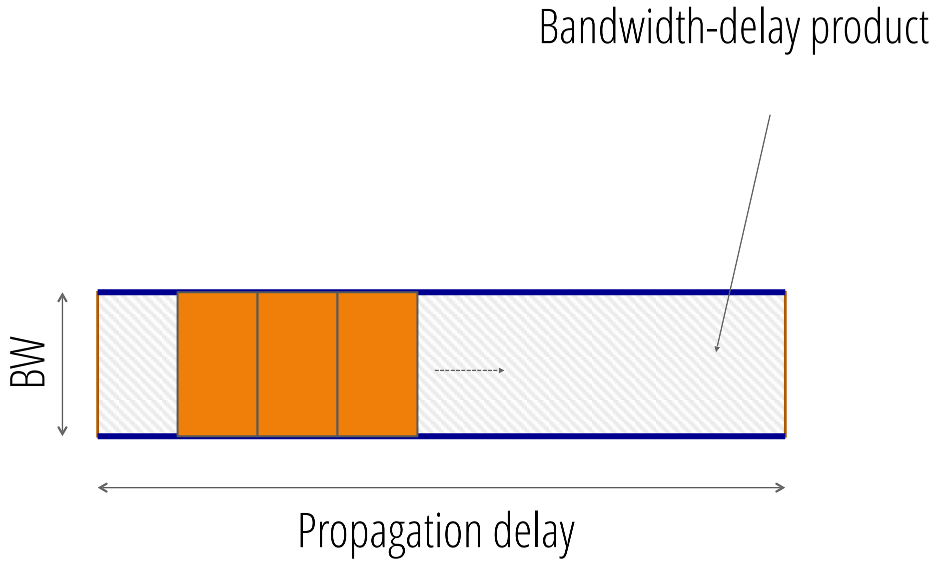
Packets on a link: an alternate "pipe" view



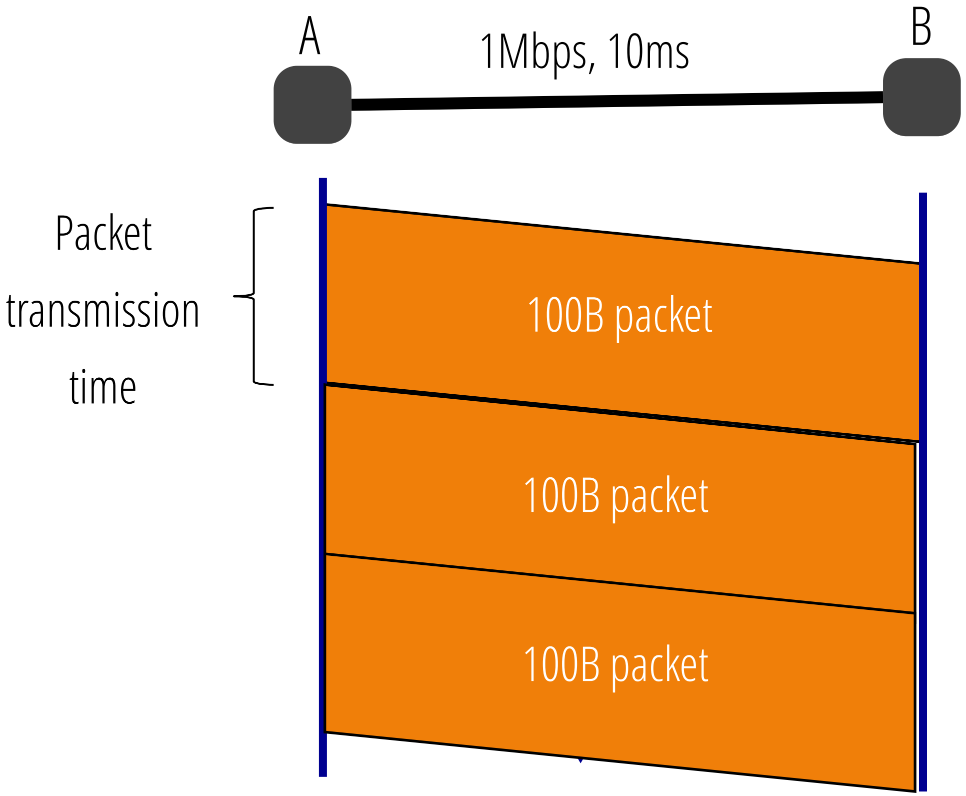


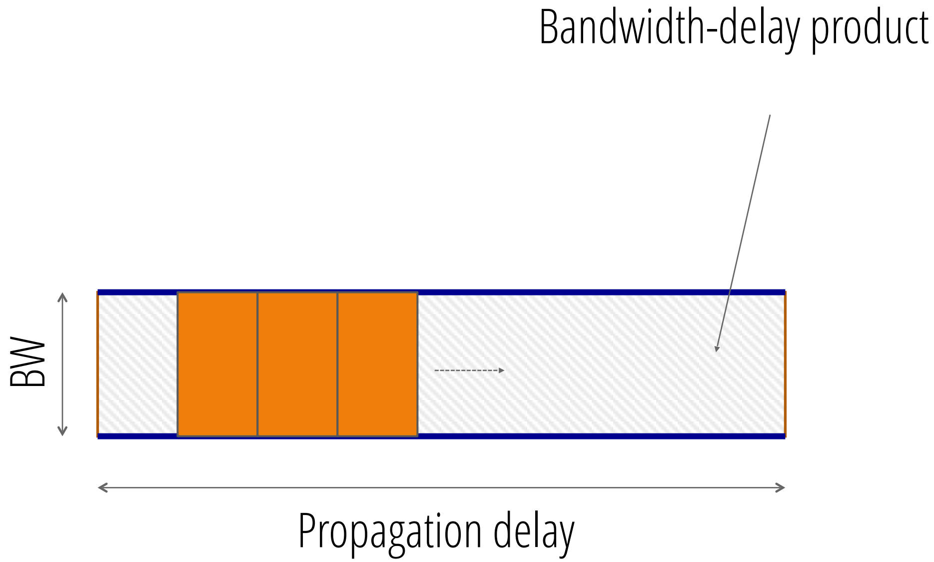
Packets on a link: an alternate “pipe” view



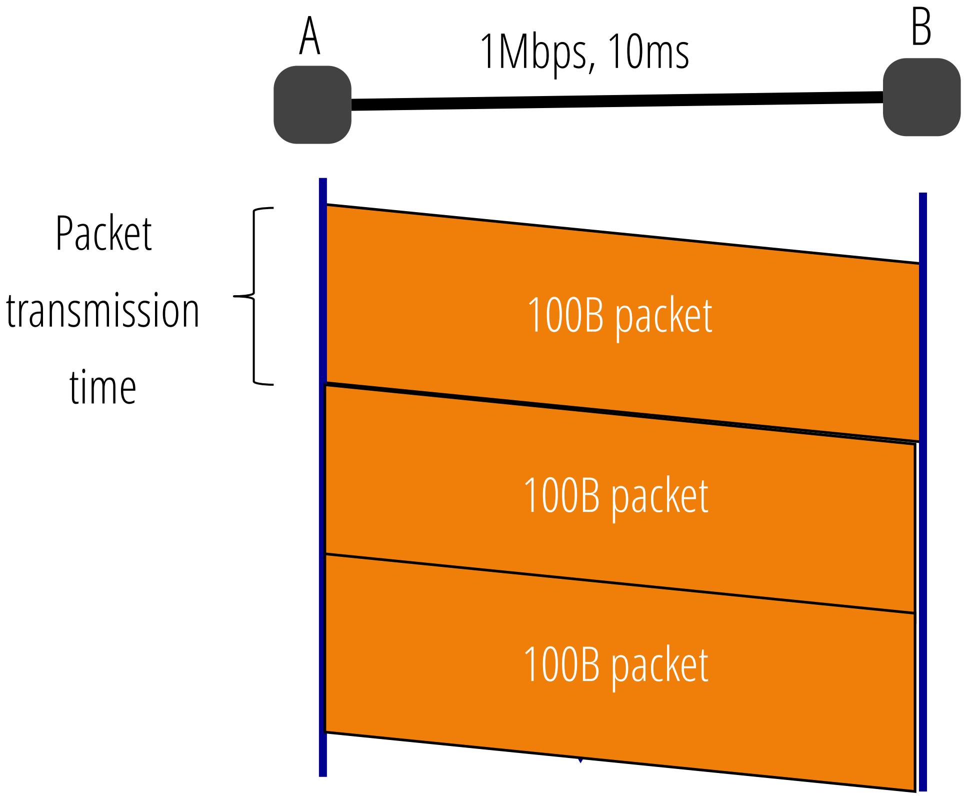


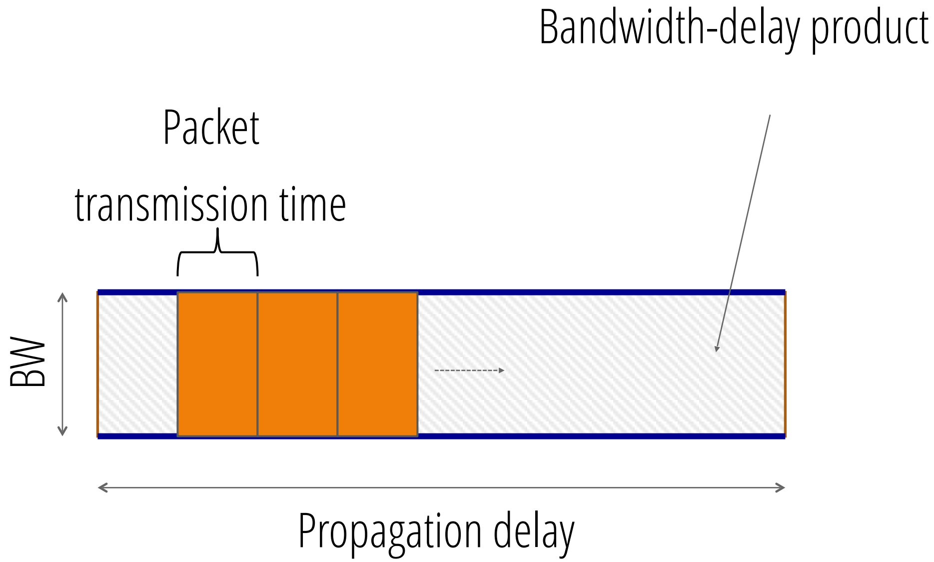
Packets on a link: an alternate "pipe" view



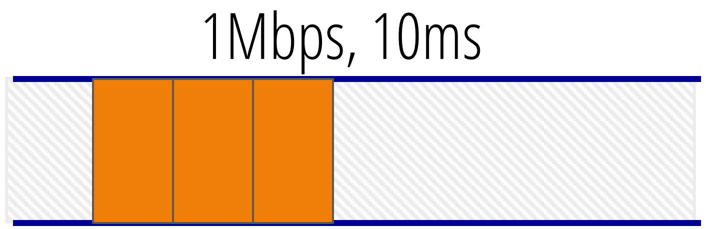


Packets on a link: an alternate "pipe" view

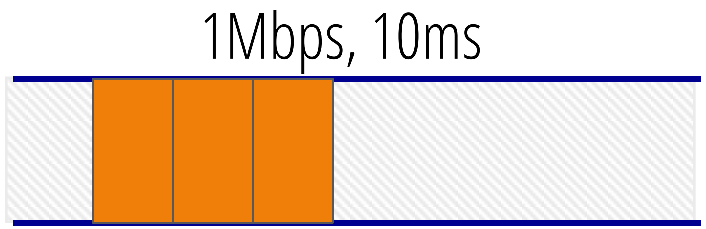




Packets on a link: an alternate “pipe” view



Packets on a link: an alternate "pipe" view

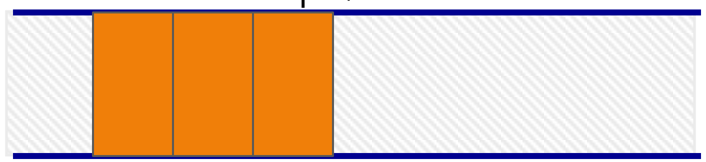


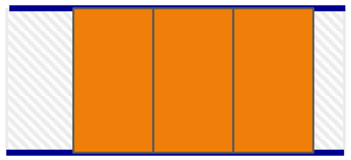
1Mbps, 5ms ?

10Mbps, 1ms ?

Packets on a link: an alternate "pipe" view

1Mbps, 10ms

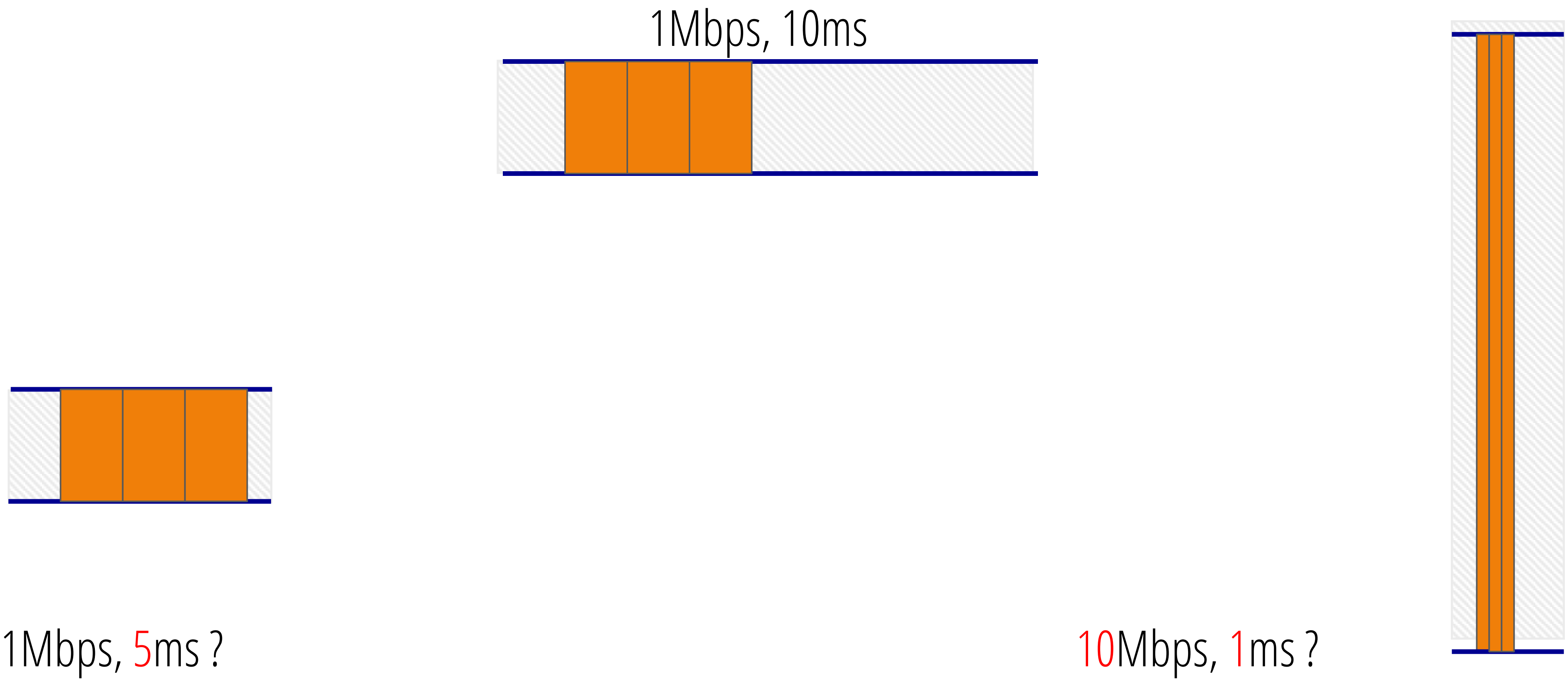




1Mbps, 5ms ?

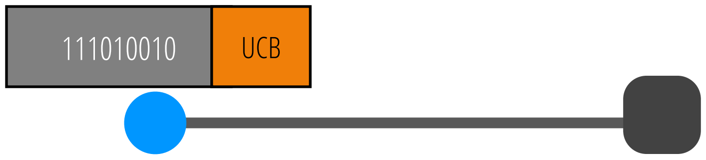
10Mbps, 1ms ?

Packets on a link: an alternate “pipe” view

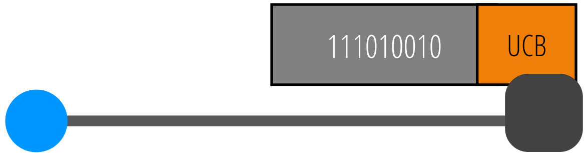


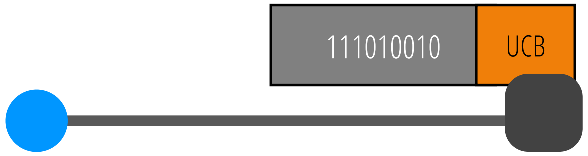
Recap: packet on a link

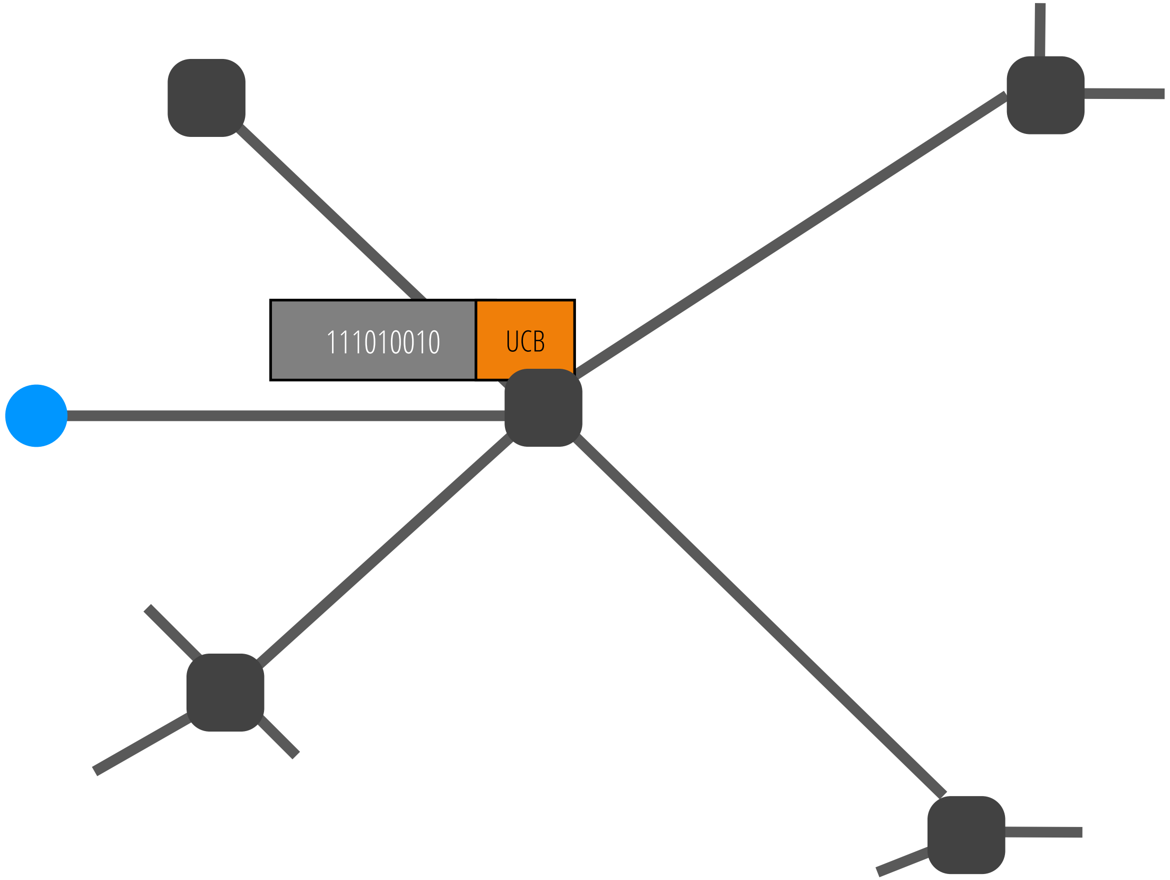
Recap: packet on a link

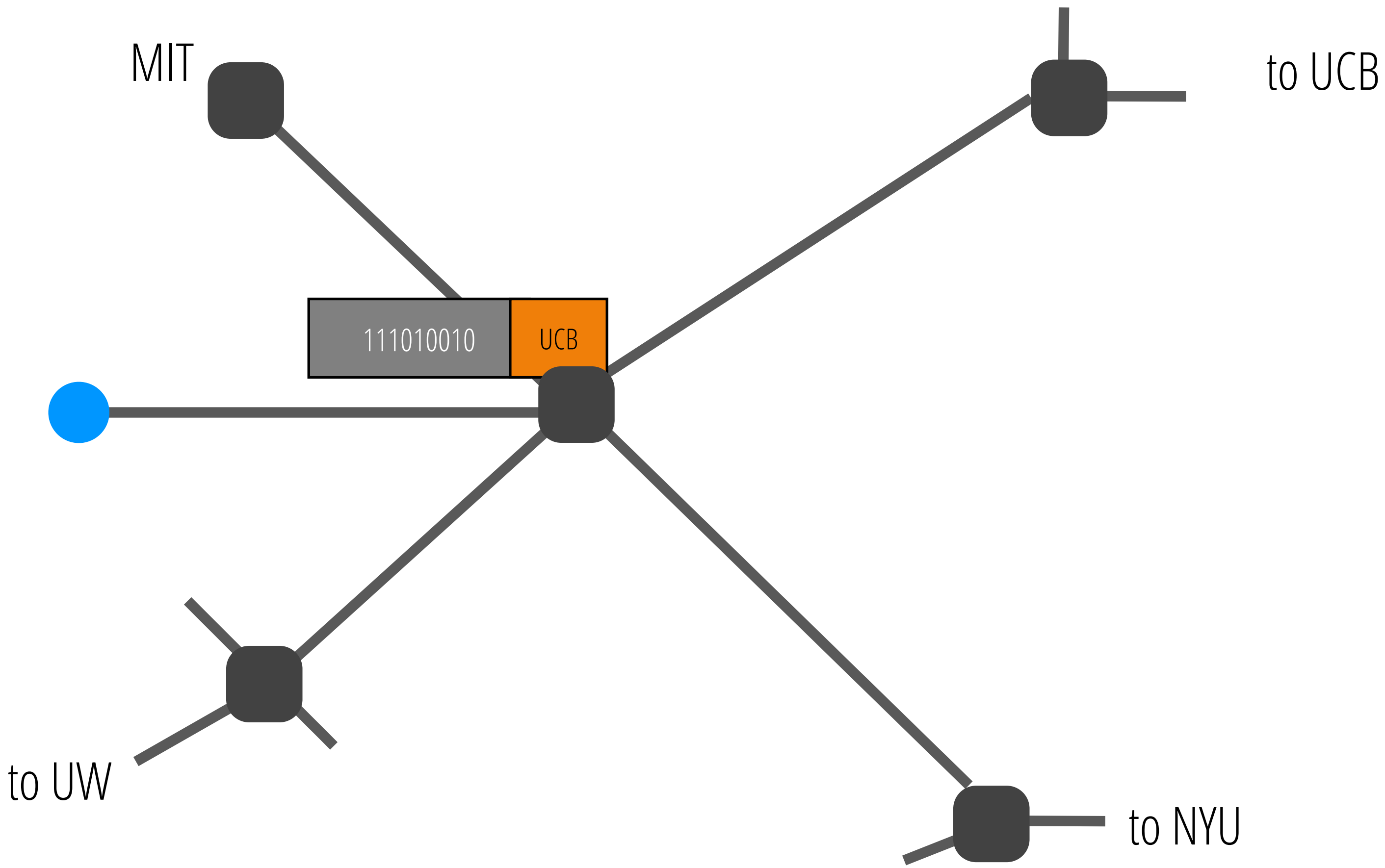


Recap: packet on a link

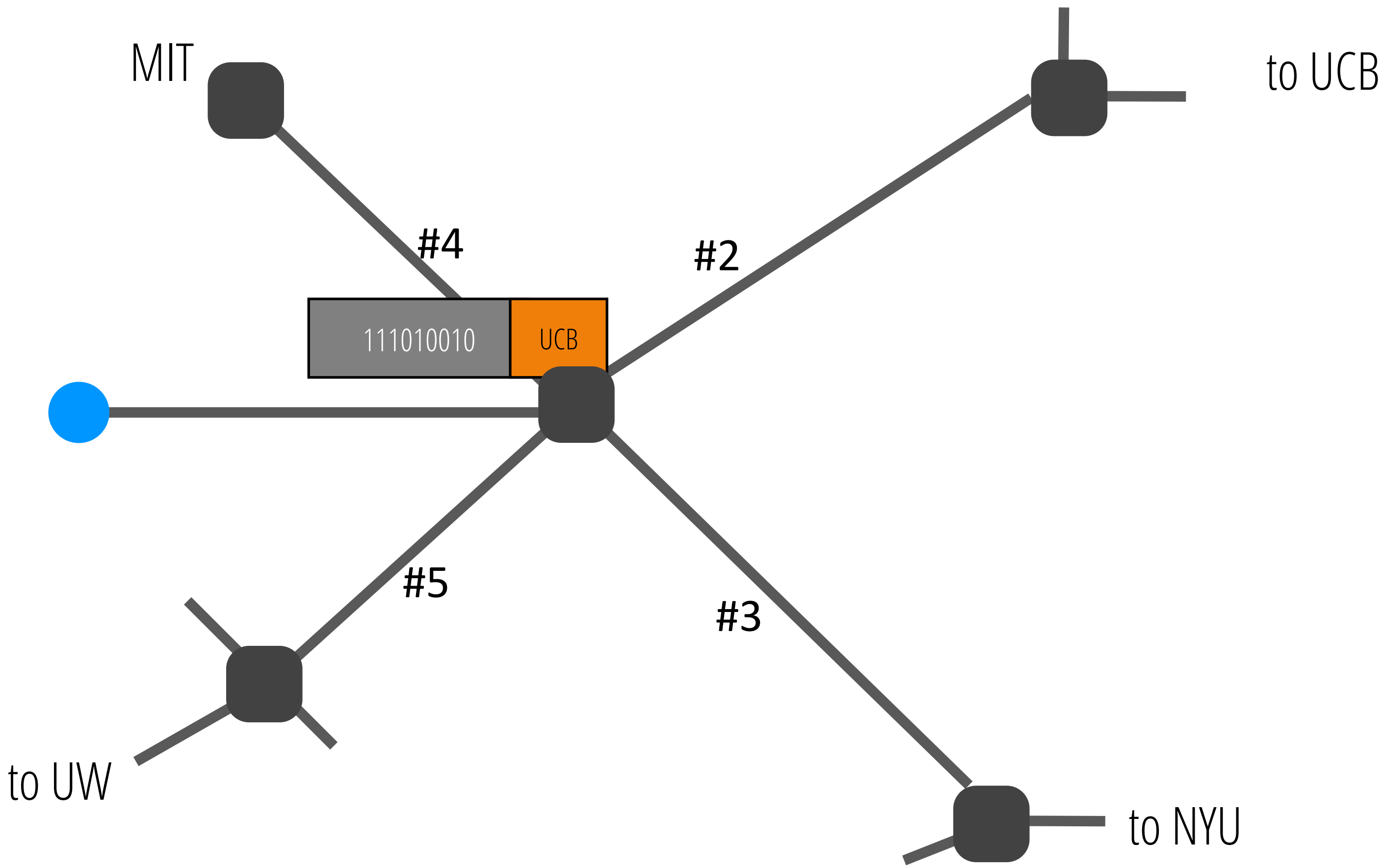


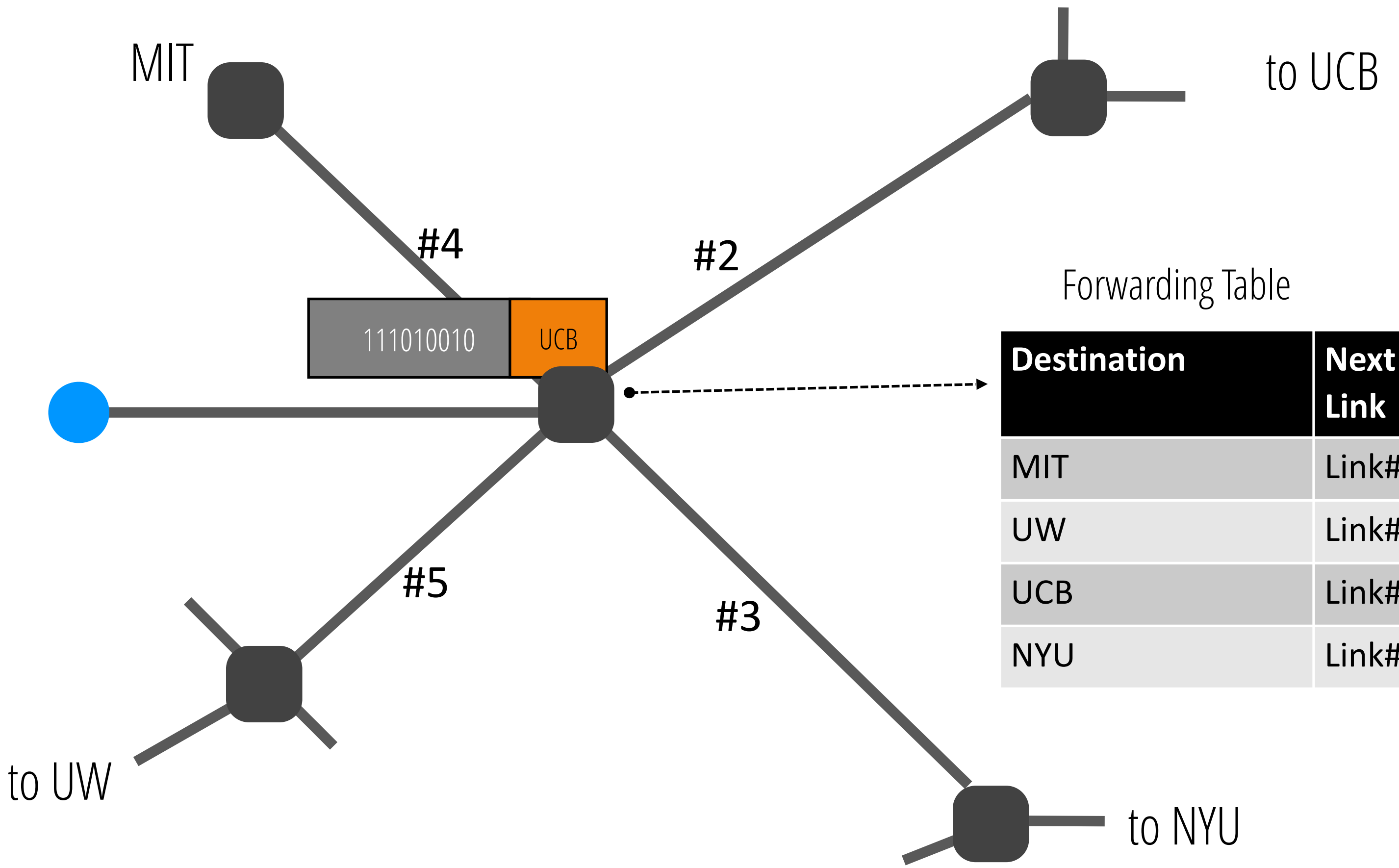

111010010
UCB


111010010
UCB



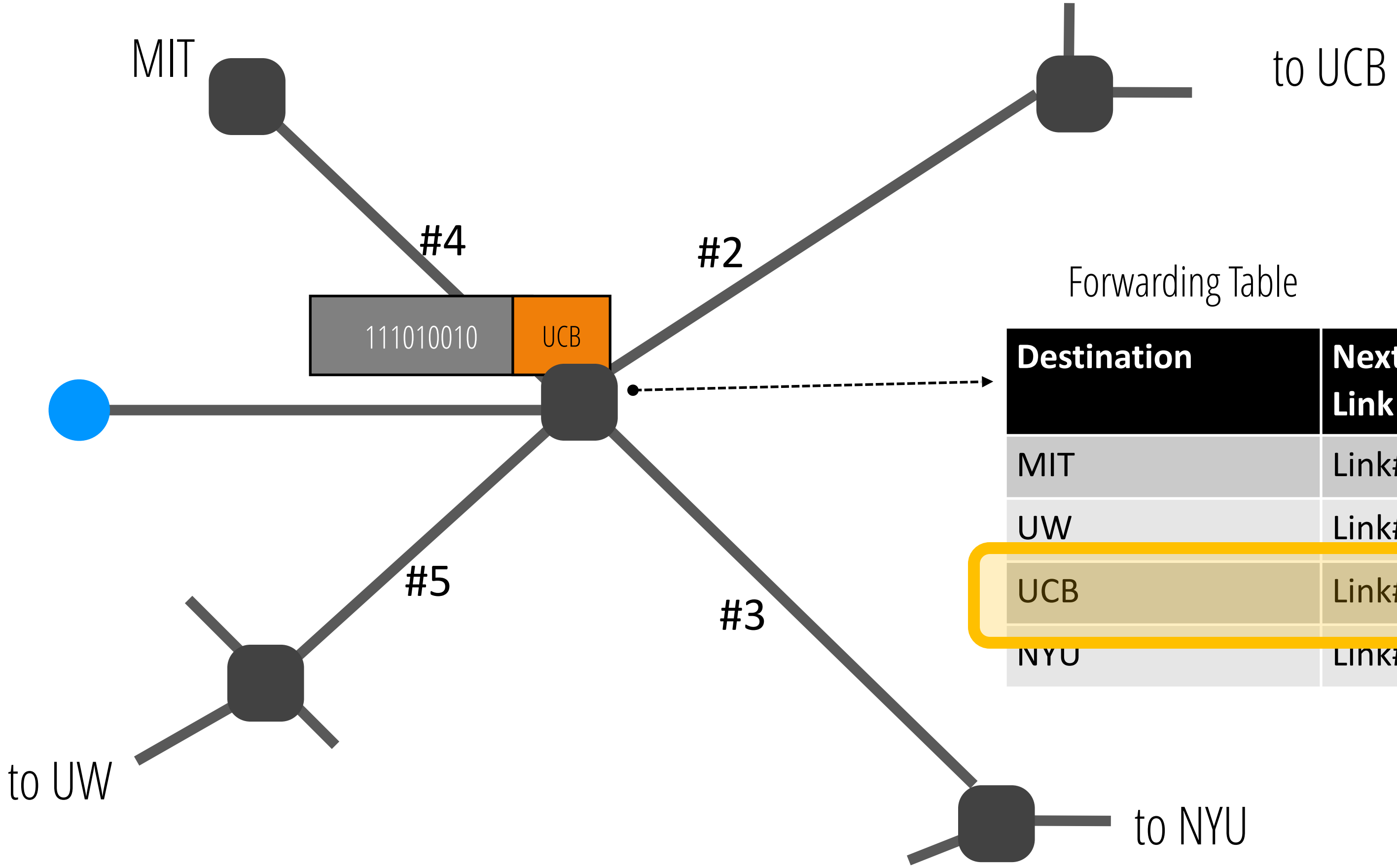
MIT
to UCB
111010010
UCB
to UW
to NYU


MIT
to UCB
#4
#2
111010010
UCB
#5
#3
to UW
to NYU



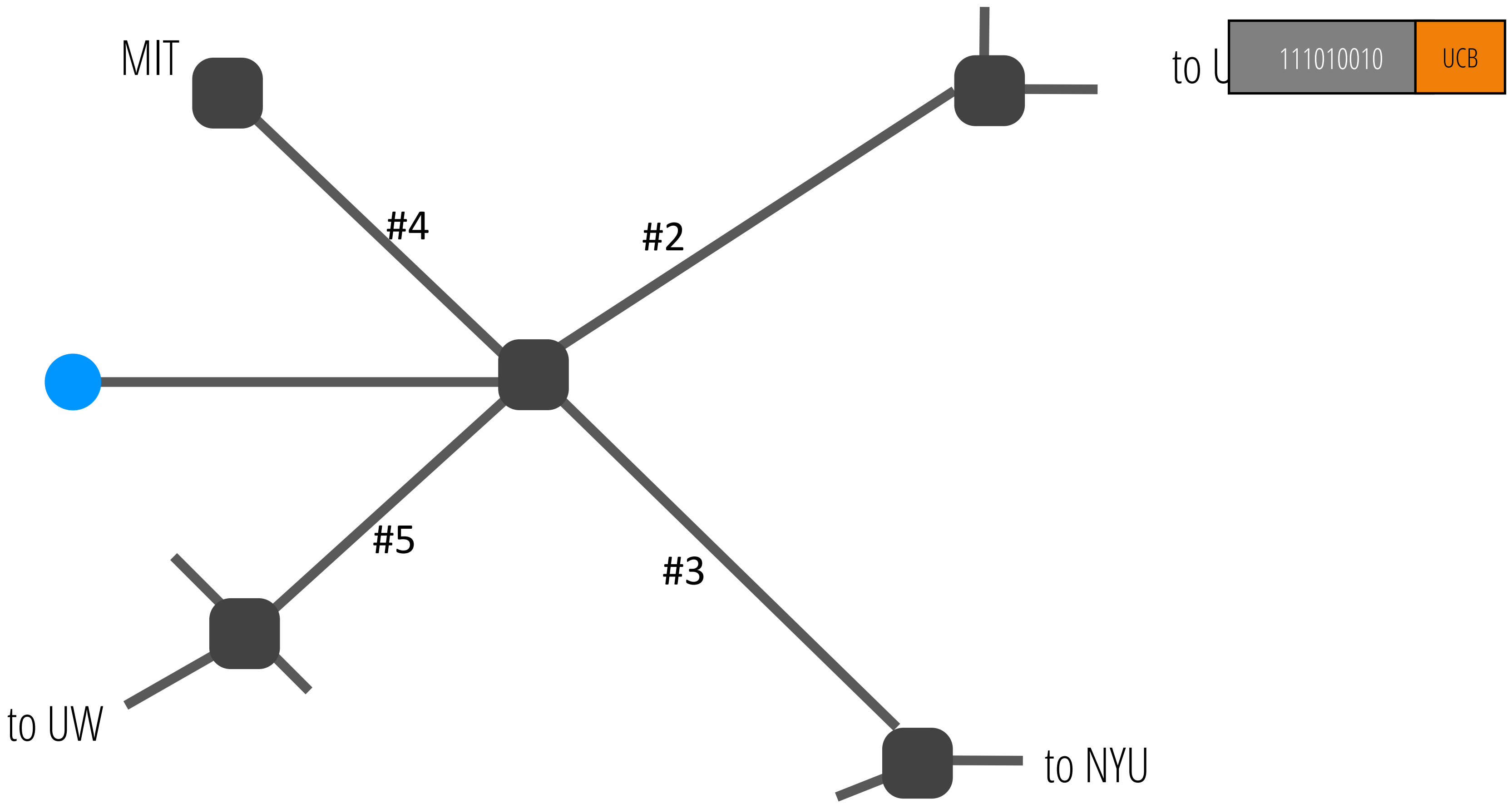
Forwarding Table

Destination	Next Hop Link
MIT	Link#4
UW	Link#5
UCB	Link#2
NYU	Link#3



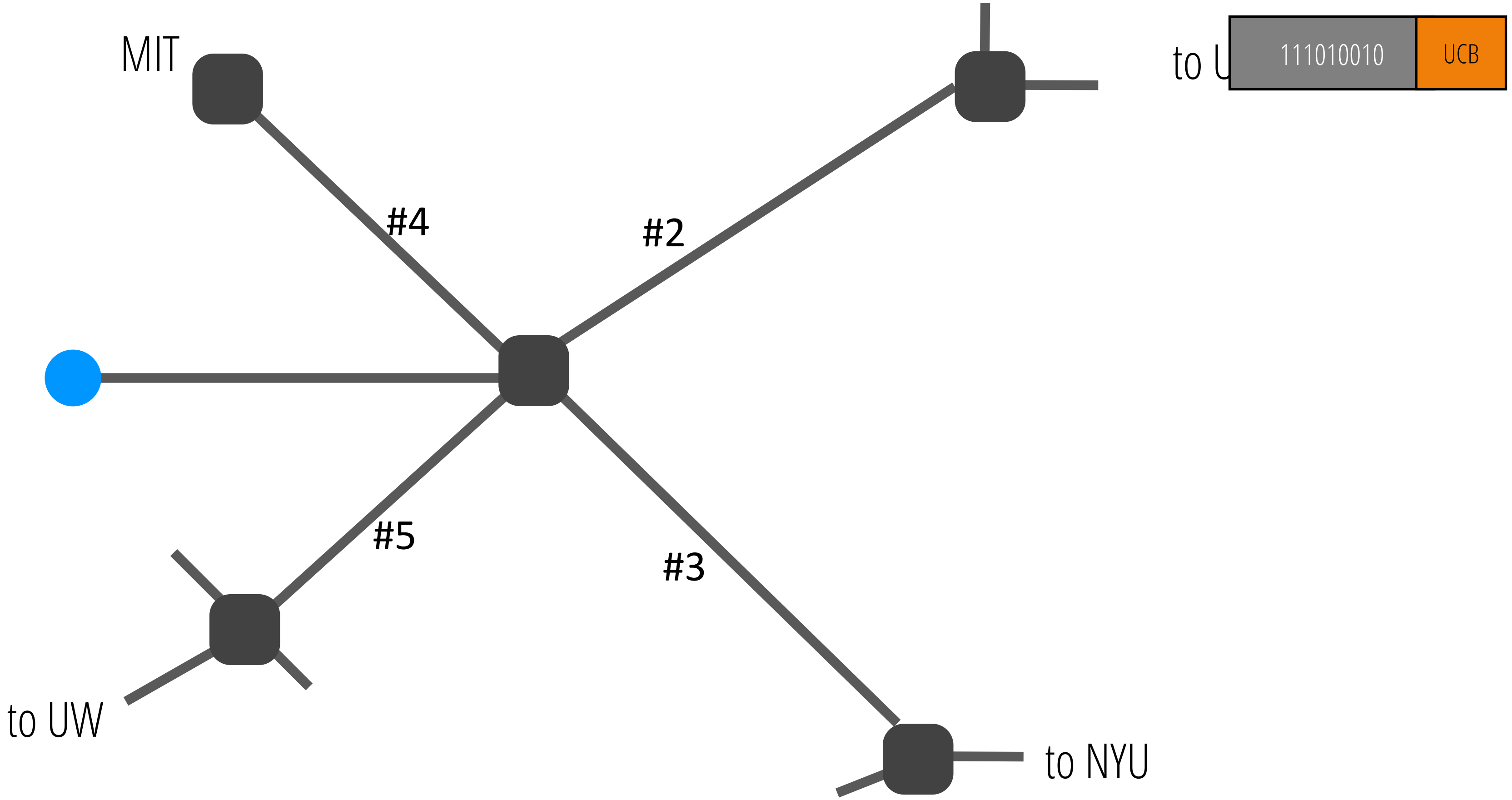
Forwarding Table

Destination	Next Hop Link
MIT	Link#4
UW	Link#5
UCB	Link#2
NYU	Link#3



MIT
#4
#2
to U
111010010
UCB
#5
#3
to UW
to NYU

Switches "forward" packets



Recap: life of a packet so far…

Recap: life of a packet so far…

- Source has some data to send to a destination

Recap: life of a packet so far…

- Source has some data to send to a destination
- Chunks it up into packets: each packet has a payload and a header

Recap: life of a packet so far…

- Source has some data to send to a destination
- Chunks it up into packets: each packet has a payload and a header
- Packet travels along a link

Recap: life of a packet so far…

- Source has some data to send to a destination
- Chunks it up into packets: each packet has a payload and a header
- Packet travels along a link
- Arrives at a switch; switch forwards the packet to its next hop

Recap: life of a packet so far…

- Source has some data to send to a destination
- Chunks it up into packets: each packet has a payload and a header
- Packet travels along a link
- Arrives at a switch; switch forwards the packet to its next hop

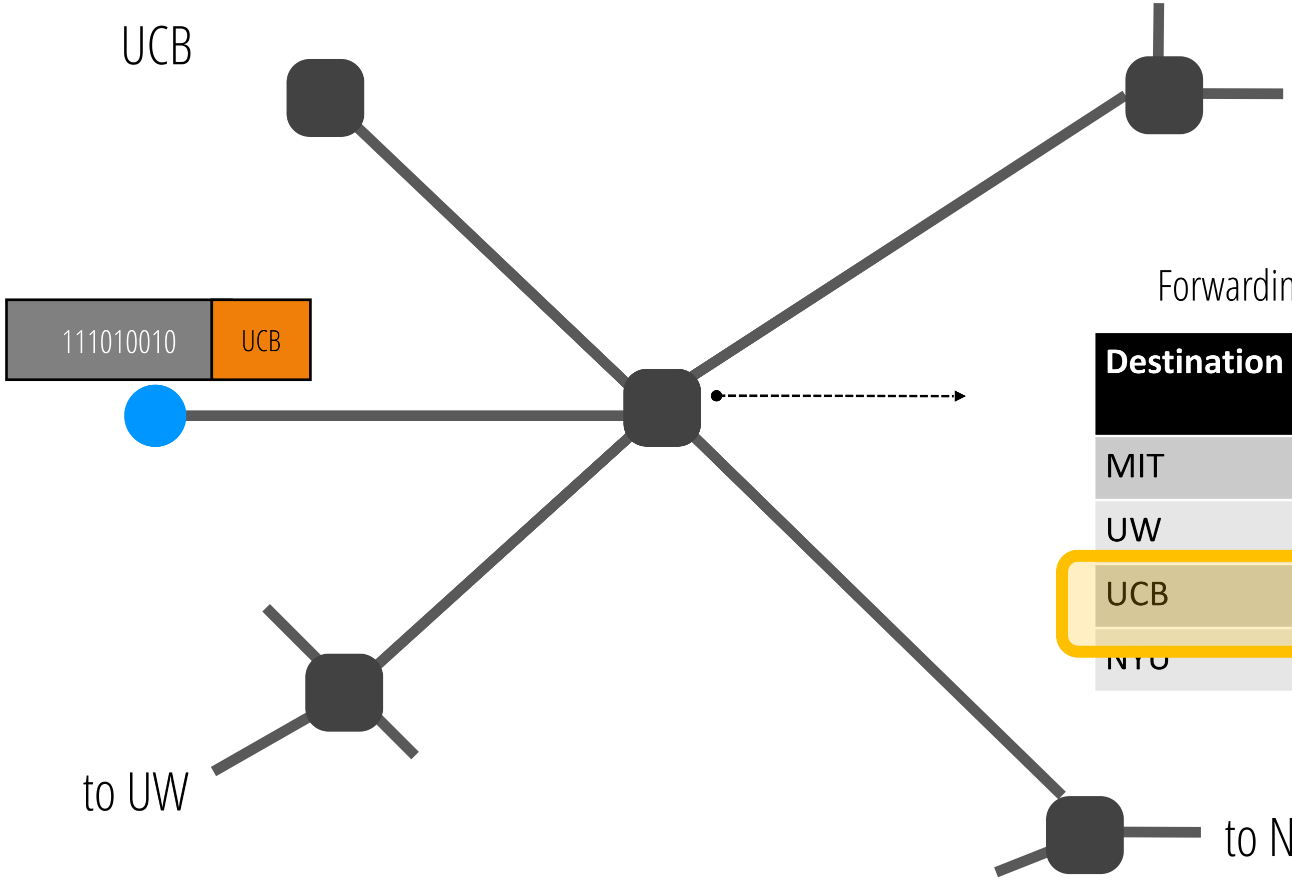
And the last two steps repeat until we reach the destination…

Recap: life of a packet so far…

- Source has some data to send to a destination
- Chunks it up into packets: each packet has a payload and a header
- Packet travels along a link
- Arrives at a switch; switch forwards the packet to its next hop

And the last two steps repeat until we reach the destination…

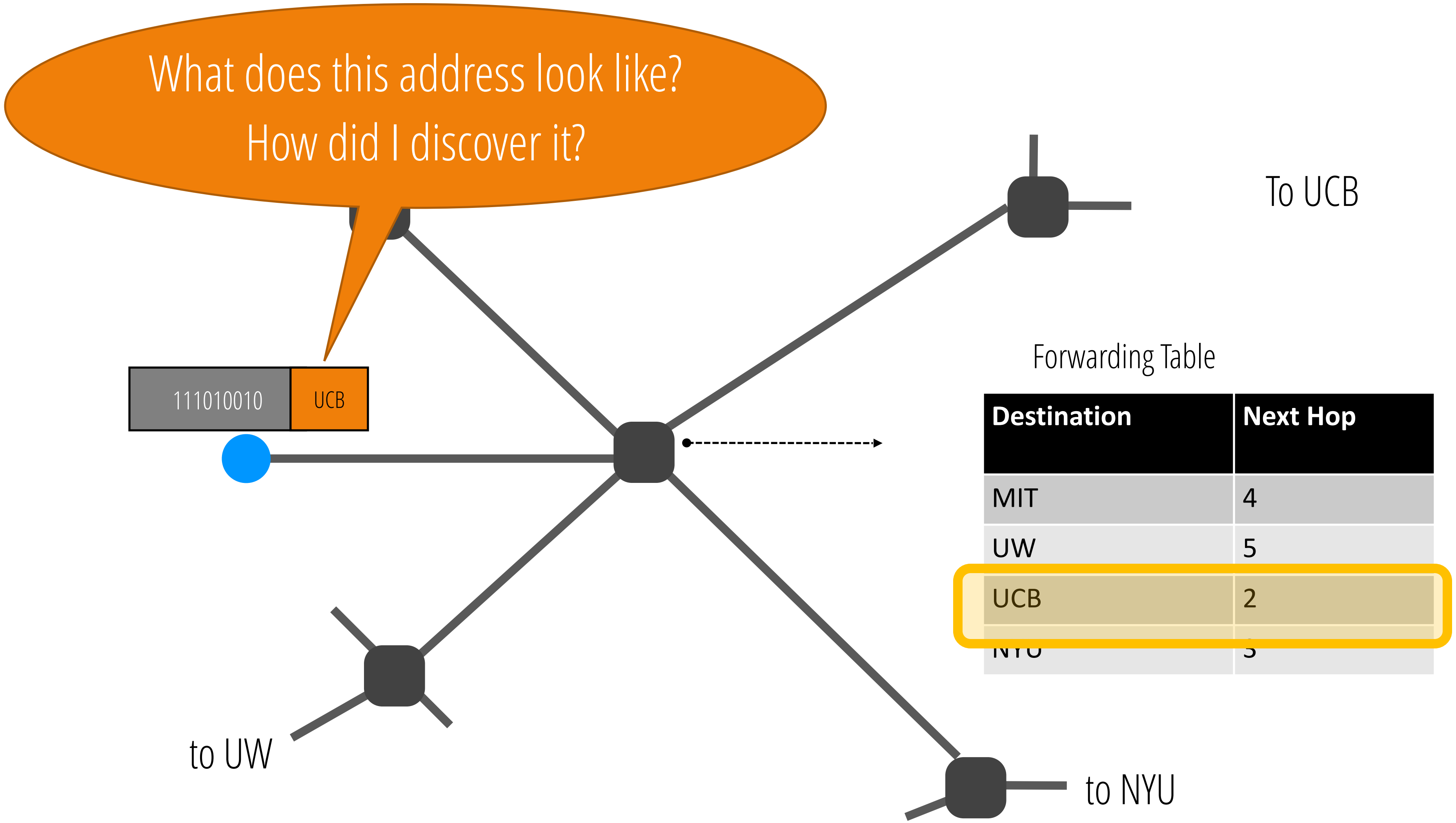
What are the fundamental challenges in this?



Forwarding Table

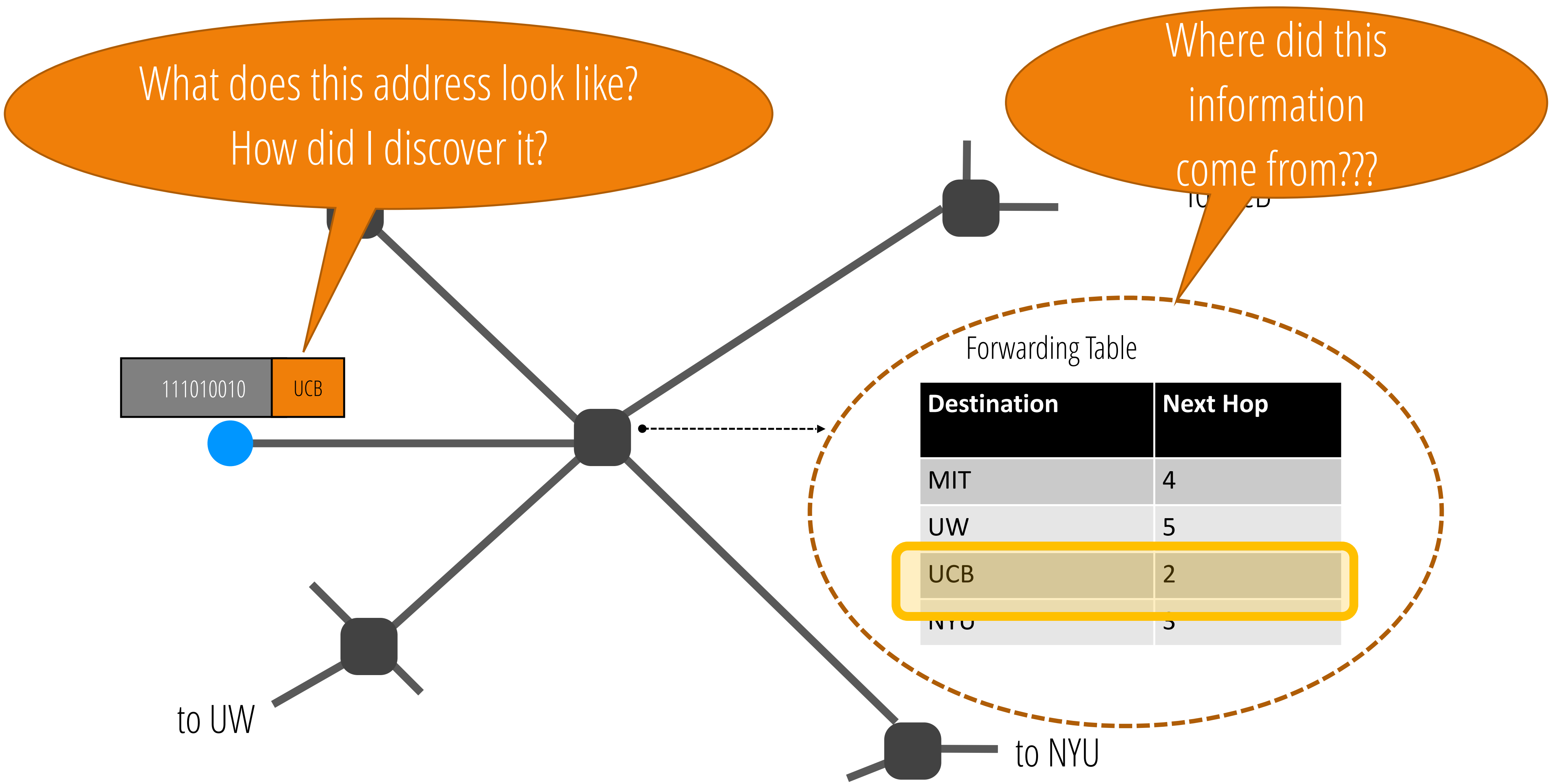
Destination	Next Hop
MIT	4
UW	5
UCB	2
NYU	3

What are the fundamental challenges in this?



Destination	Next Hop
MIT	4
UW	5
UCB	2
NYU	3

What are the fundamental challenges in this?



What are the fundamental challenges in this?

Challenge: addressing and naming

Challenge: addressing and naming

- In the real world, we have *names* and *addresses*
  - E.g., my name is Sylvia; my address is 413 Soda Hall
  - When I move to a new building: my name doesn't change but my address does

Challenge: addressing and naming

- In the real world, we have *names* and *addresses*
  - E.g., my name is Sylvia; my address is 413 Soda Hall
  - When I move to a new building: my name doesn't change but my address does
- Network address: where host is located

Challenge: addressing and naming

- In the real world, we have *names* and *addresses*
  - E.g., my name is Sylvia; my address is 413 Soda Hall
  - When I move to a new building: my name doesn't change but my address does

- Network address: where host is located

- Network name: which host it is

- Need an addressing and naming scheme that works at Internet scale!

Challenge: addressing and naming

- In the real world, we have *names* and *addresses*
  - E.g., my name is Sylvia; my address is 413 Soda Hall
  - When I move to a new building: my name doesn't change but my address does

- Network address: where host is located

- Network name: which host it is

- Need an addressing and naming scheme that works at Internet scale!

Will discuss IP addressing a few lectures from now

Challenge: mapping names to addresses

Challenge: mapping names to addresses

- Consider when you access a web page
  - Insert URL into browser (e.g., cnn.com)
  - You want to communicate with the server hosting cnn.com content

Challenge: mapping names to addresses

- Consider when you access a web page
  - Insert URL into browser (e.g., cnn.com)
  - You want to communicate with the server hosting cnn.com content

- How do you get to the server?
  - URL is user-level *name* (e.g., cnn.com)
  - Network needs address (e.g., where is cnn.com?)

Challenge: mapping names to addresses

- Consider when you access a web page
  - Insert URL into browser (e.g., cnn.com)
  - You want to communicate with the server hosting cnn.com content

- How do you get to the server?
  - URL is user-level *name* (e.g., cnn.com)
  - Network needs address (e.g., where is cnn.com?)

- Must map – or “resolve” -- host names to addresses

- Done by the Domain Name System (DNS)

Challenge: mapping names to addresses

- Consider when you access a web page
  - Insert URL into browser (e.g., cnn.com)
  - You want to communicate with the server hosting cnn.com content

- How do you get to the server?
  - URL is user-level *name* (e.g., cnn.com)
  - Network needs address (e.g., where is cnn.com?)

- Must map – or “resolve” -- host names to addresses

- Done by the Domain Name System (DNS)

Will cover DNS in a later lecture (second half of semester)

Challenge: Routing

Challenge: Routing

- When a packet arrives at a router, the forwarding table determines which outgoing link the packet is sent on

Challenge: Routing

- When a packet arrives at a router, the forwarding table determines which outgoing link the packet is sent on

- How do you compute the forwarding tables necessary to deliver packets?

Challenge: Routing

- When a packet arrives at a router, the forwarding table determines which outgoing link the packet is sent on

- How do you compute the forwarding tables necessary to deliver packets?

Will devote multiple lectures (and one project) to this question!

Routing (Conceptually)

Routing (Conceptually)

- Distributed routing algorithm run between switches/routers
- Gather information about the network topology

Routing (Conceptually)

- Distributed routing algorithm run between switches/routers
- Gather information about the network topology
- Compute paths through that topology
- Store forwarding information in each router:
  - If packet is destined for X, send it on this link
  - If packet is destined for Y, send it on that link
  - …

Routing (Conceptually)

- Distributed routing algorithm run between switches/routers
- Gather information about the network topology
- Compute paths through that topology
- Store forwarding information in each router:
  - If packet is destined for X, send it on this link
  - If packet is destined for Y, send it on that link
  - …
- This is the forwarding table

Control Plane *vs* Data Plane

Control Plane *vs* Data Plane

- Control plane: mechanisms used to compute forwarding tables
  - Inherently global: must know topology to compute
  - *Routing algorithm is part of the control plane*
  - Time scale: per network event

Control Plane *vs* Data Plane

- Control plane: mechanisms used to compute forwarding tables
  - Inherently global: must know topology to compute
  - *Routing algorithm is part of the control plane*
  - Time scale: per network event

Control Plane *vs* Data Plane

- Control plane: mechanisms used to compute forwarding tables
  - Inherently global: must know topology to compute
  - *Routing algorithm is part of the control plane*
  - Time scale: per network event

- Data plane: using those tables to actually forward packets
  - Inherently local: depends only on arriving packet and local routing table
  - *Forwarding mechanism ("lookup" algorithm) is part of data plane*
  - Time scale: per packet arrival

Control Plane: Challenge

Control Plane: Challenge

- Computing routes at scale

Control Plane: Challenge

- Computing routes at scale
- In the face of network failures and topology changes

Control Plane: Challenge

- Computing routes at scale
- In the face of network failures and topology changes

(Will study routing algorithms starting week#3)

Control Plane: Challenge

- Computing routes at scale
- In the face of network failures and topology changes

(Will study routing algorithms starting week#3)

- While respecting ISPs' need for autonomy

Control Plane: Challenge

- Computing routes at scale
- In the face of network failures and topology changes

(Will study routing algorithms starting week#3)

- While respecting ISPs' need for autonomy
  - Each ISP gets to choose how to do routing *within* its networks

Control Plane: Challenge

- Computing routes at scale
- In the face of network failures and topology changes

(Will study routing algorithms starting week#3)

- While respecting ISPs' need for autonomy
  - Each ISP gets to choose how to do routing *within* its networks
  - And they typically do not want to reveal the internals of this decision making

Control Plane: Challenge

- Computing routes at scale
- In the face of network failures and topology changes

(Will study routing algorithms starting week#3)

- While respecting ISPs' need for autonomy
  - Each ISP gets to choose how to do routing *within* its networks
  - And they typically do not want to reveal the internals of this decision making
  - Can we ensure that ISPs' independent decisions result in usable end-to-end routes?

Control Plane: Challenge

- Computing routes at scale
- In the face of network failures and topology changes

(Will study routing algorithms starting week#3)

- While respecting ISPs' need for autonomy
  - Each ISP gets to choose how to do routing *within* its networks
  - And they typically do not want to reveal the internals of this decision making
  - Can we ensure that ISPs' independent decisions result in usable end-to-end routes?

  (Will study BGP in depth later in the semester)

Data Plane: Challenge

Data Plane: Challenge

- Consider a 1 Tbps link (10^{12}) receiving 10,000 bit packets
  - New packet arrives every 10 nanoseconds (10^{-8})

Data Plane: Challenge

- Consider a 1 Tbps link (10^{12}) receiving 10,000 bit packets
  - New packet arrives every 10 nanoseconds (10^{-8})

- The following operations must be done after packet arrives (in ~10 nanoseconds or less)
  - Parse packet (extract address, etc.)
  - Look up address in forwarding table
  - Update other fields in packet header (if needed)
  - Update relevant internal counters, etc.
  - Send packet to appropriate output link

(Will study router designs and IP forwarding lookup algorithms.)

Hence, our important topics (so far)

Hence, our important topics (so far)

- How do we name endhosts on the Internet? (naming)

Hence, our important topics (so far)

- How do we name endhosts on the Internet? (naming)
- How do we address endhosts? (addressing)

Hence, our important topics (so far)

- How do we name endhosts on the Internet? (naming)
- How do we address endhosts? (addressing)
- How do we map names to addresses? (mapping names to addresses)

Hence, our important topics (so far)

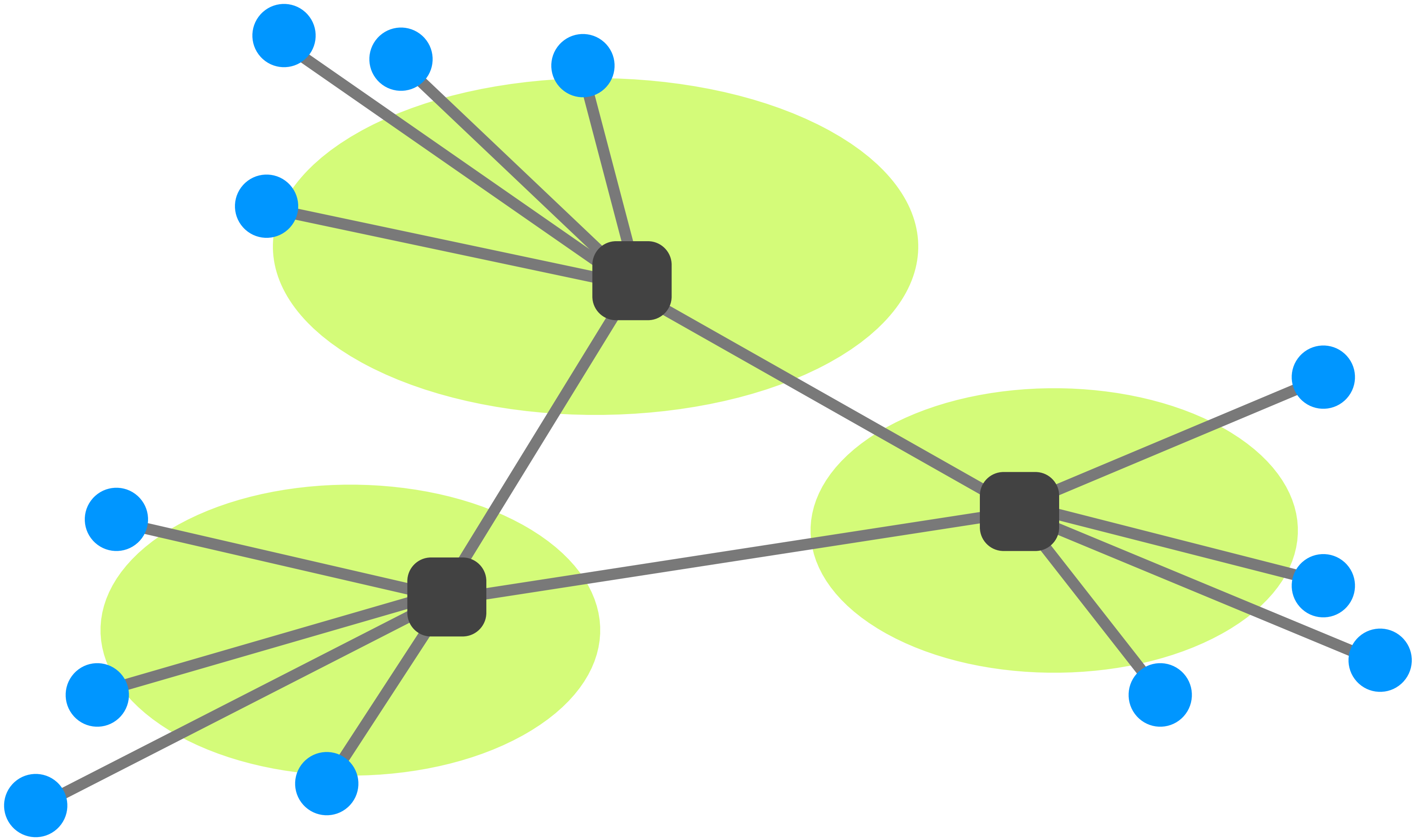
- How do we name endhosts on the Internet? (naming)
- How do we address endhosts? (addressing)
- How do we map names to addresses? (mapping names to addresses)
- How do we compute forwarding tables? (routing control plane → project 1)

Hence, our important topics (so far)

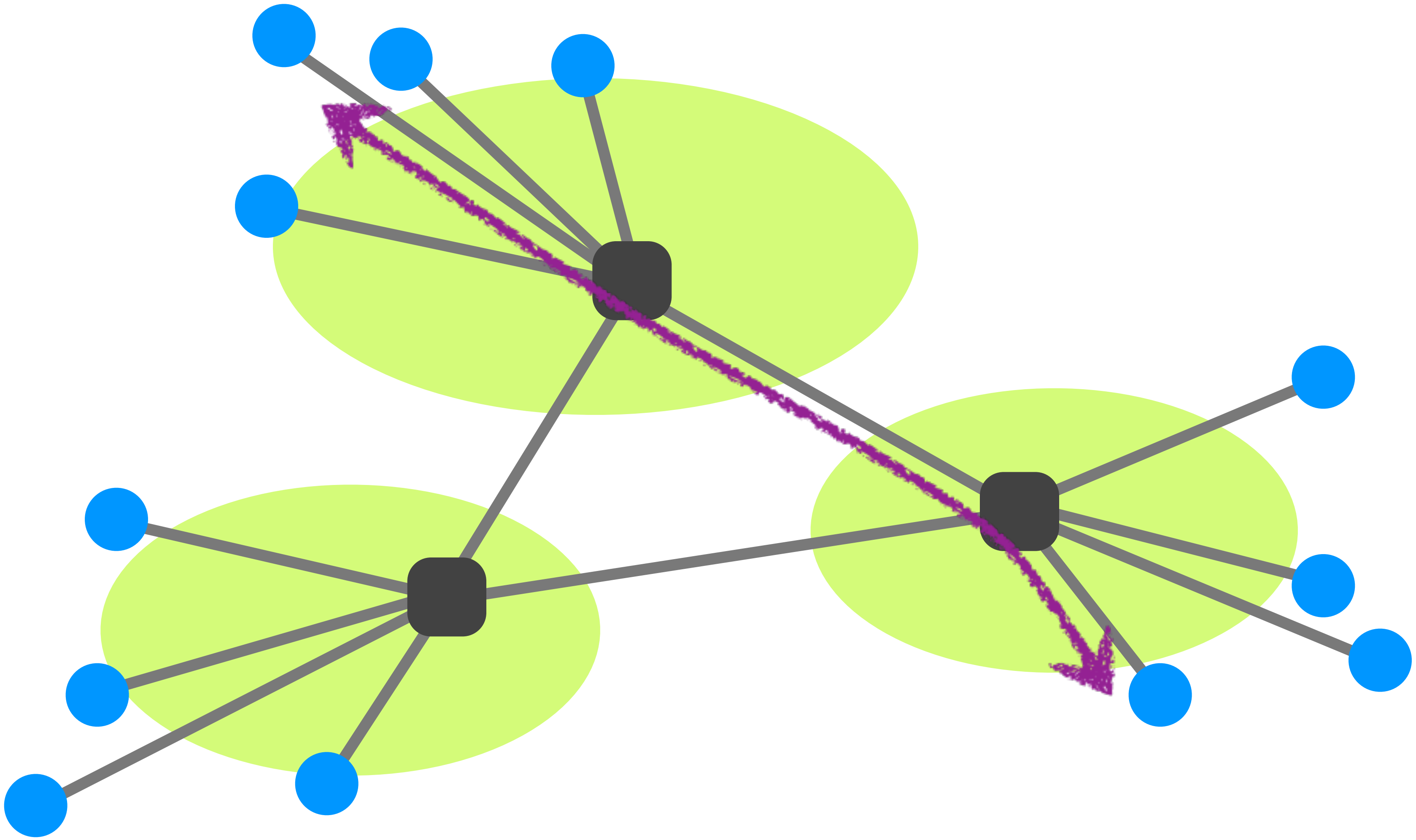
- How do we name endhosts on the Internet? (naming)
- How do we address endhosts? (addressing)
- How do we map names to addresses? (mapping names to addresses)
- How do we compute forwarding tables? (routing control plane → project 1)
- How do we forward packets? (routing data plane)

Questions??

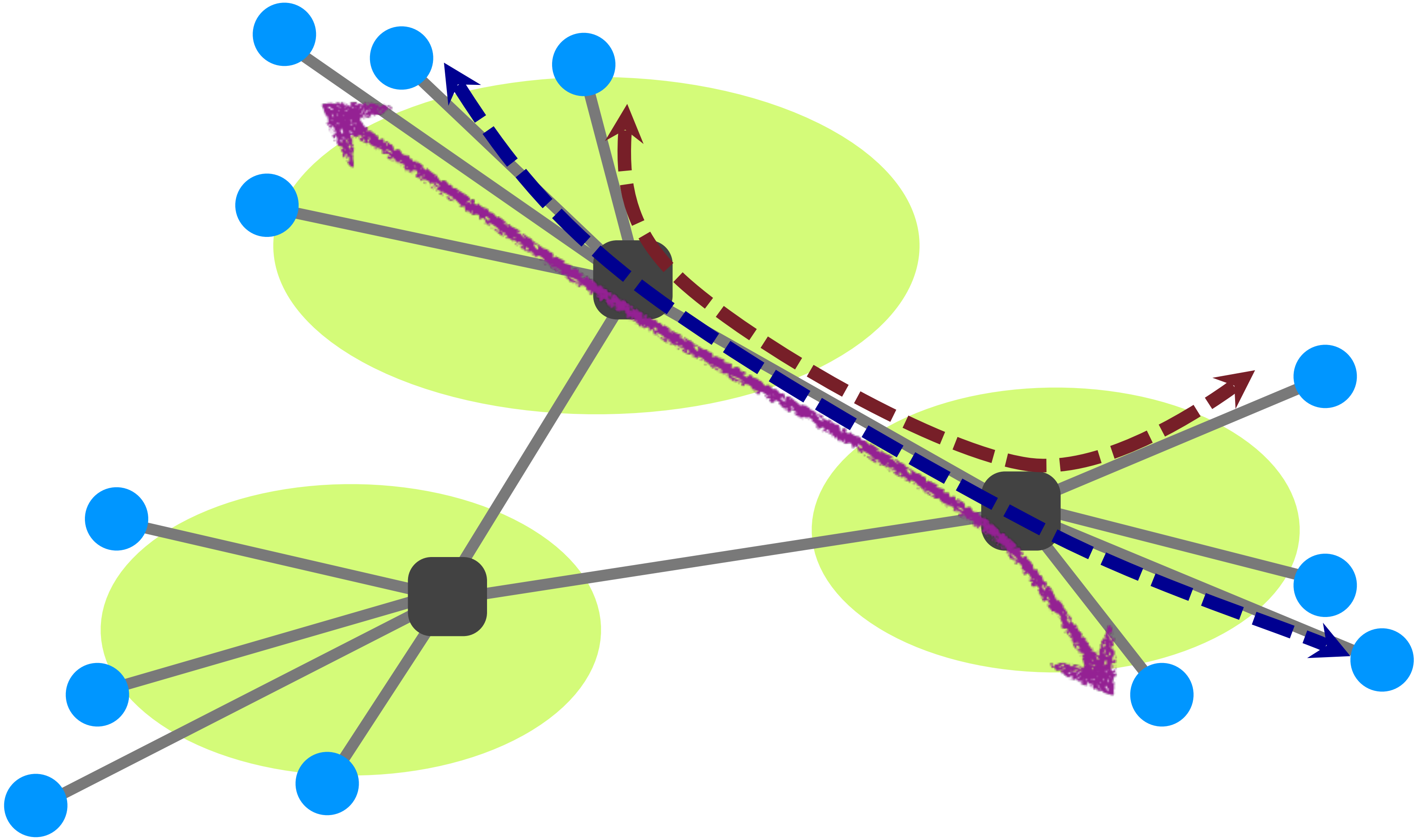
Let’s back up a level...



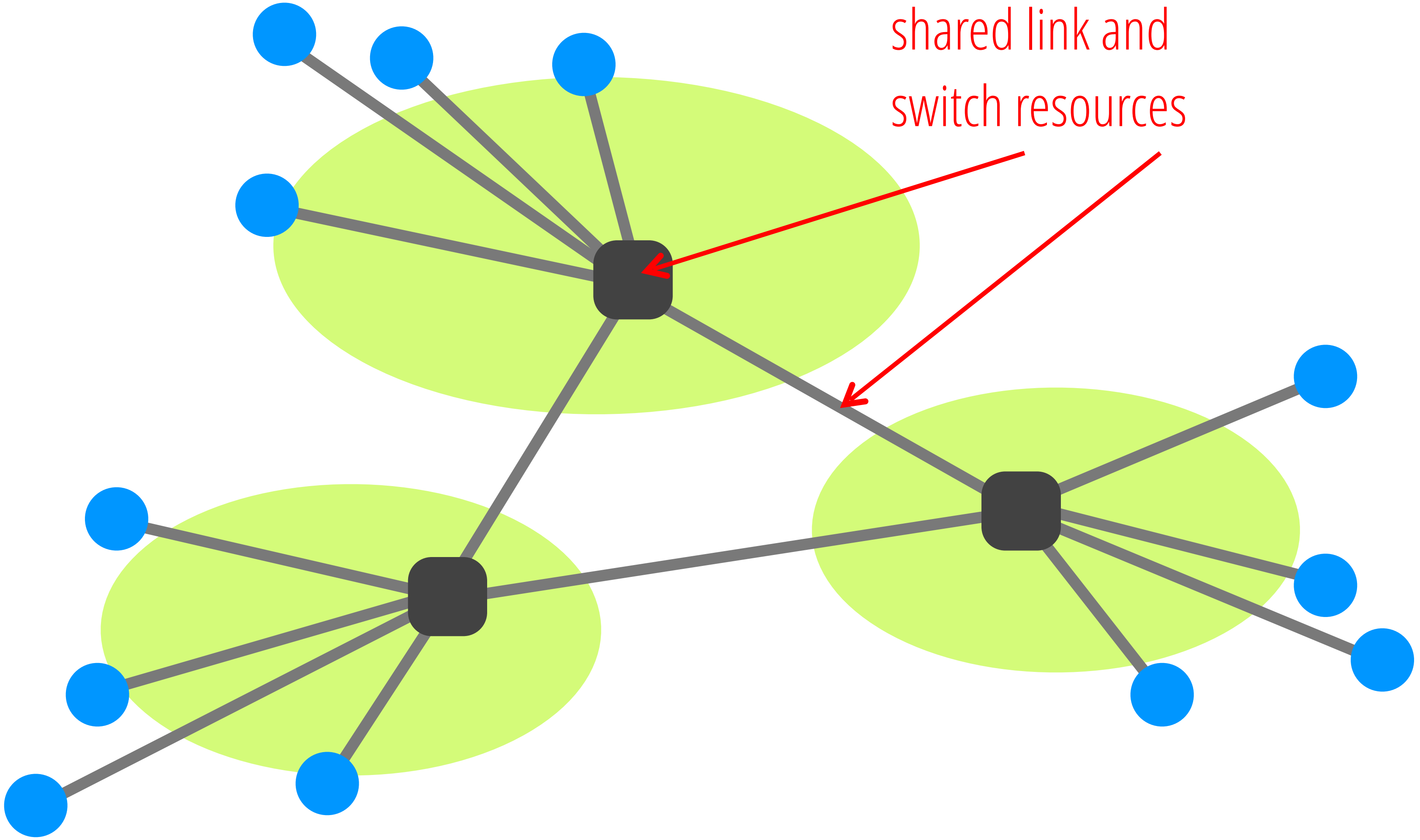
Let’s back up a level…



Let’s back up a level…



Let's back up a level...



Fundamental Fact About Networks

- Network must support many simultaneous flows at the same time
  - Recall, flow = stream of packets sent between two end hosts

- Which means network resources (links and switches) are shared between end hosts

Network resources (i.e., bandwidth) are statistically multiplexed

Statistical Multiplexing

Statistical Multiplexing

- Combining demands to share resources efficiently
  - vs. statically partitioning resources

Statistical Multiplexing

- Combining demands to share resources efficiently
  - vs. statically partitioning resources

- Long history in computer science
  - Processes on an OS (*vs.* every process has own core)
  - Cloud computing (*vs.* everyone has own datacenter)

Statistical Multiplexing

- Combining demands to share resources efficiently
  - vs. statically partitioning resources

- Long history in computer science
  - Processes on an OS (*vs.* every process has own core)
  - Cloud computing (*vs.* everyone has own datacenter)

- Based on premise: peak of aggregate demand is << aggregate of peak demands

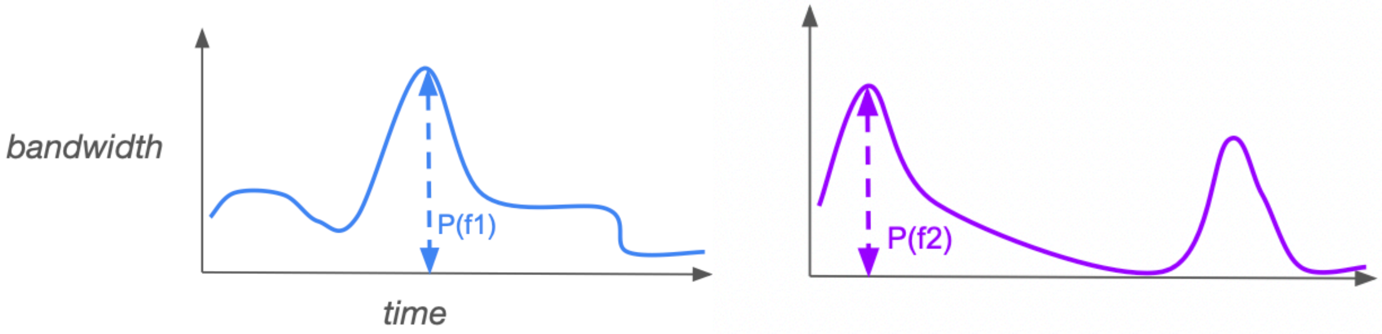
Aggregates, Peaks, etc....

Aggregates, Peaks, etc….

- Peak rate of flow f: P(f)

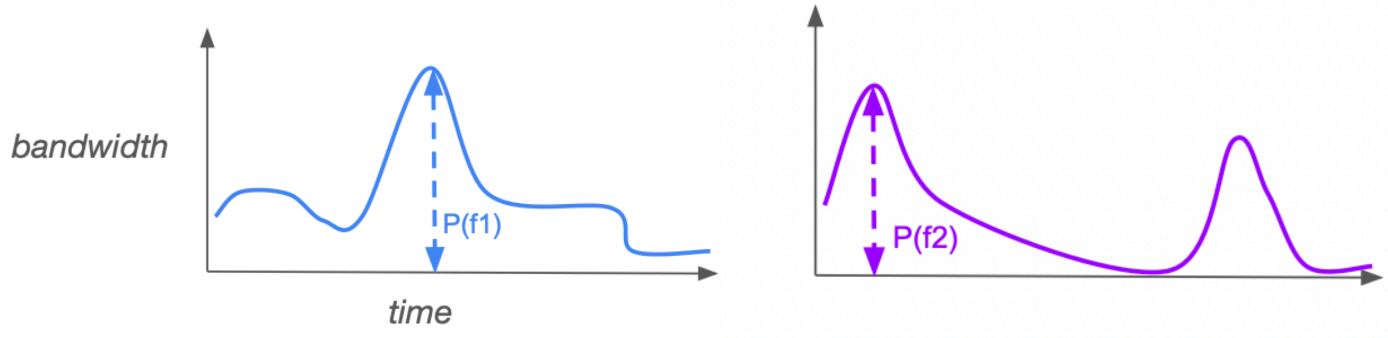
Aggregates, Peaks, etc….

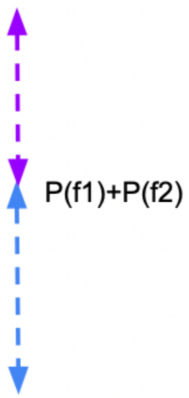
- Peak rate of flow f: P(f)
- Aggregate of peaks: $\Sigma_{\{f\}}$[P(f)]



Aggregates, Peaks, etc….

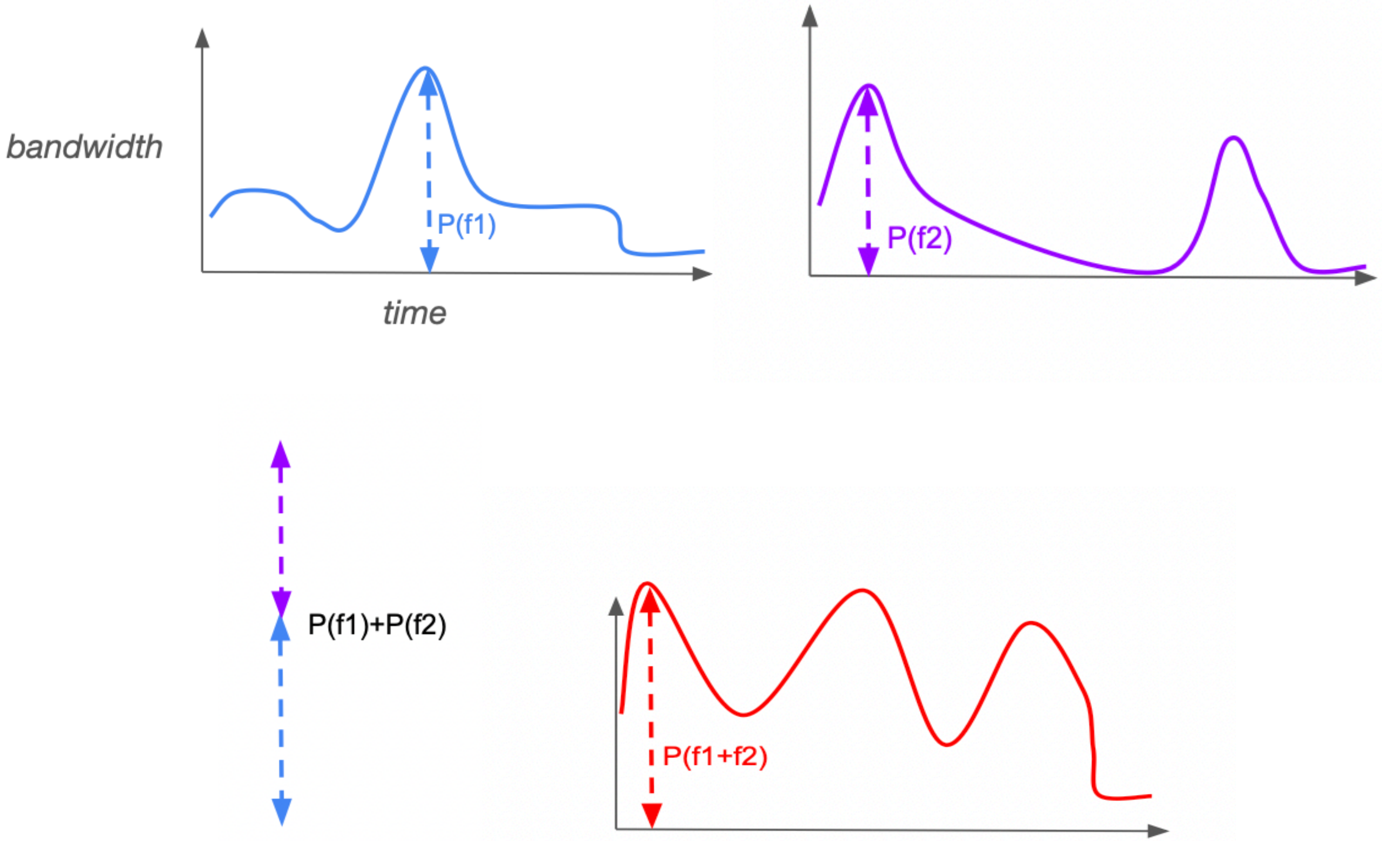
- Peak rate of flow f: P(f)
- Aggregate of peaks: $\Sigma_{\{f\}}[P(f)]$
- Peak of aggregate: $P(\Sigma_{\{f\}} f)$
- Typically: $\Sigma_{\{f\}}[P(f)] >> P(\Sigma_{\{f\}} f)$





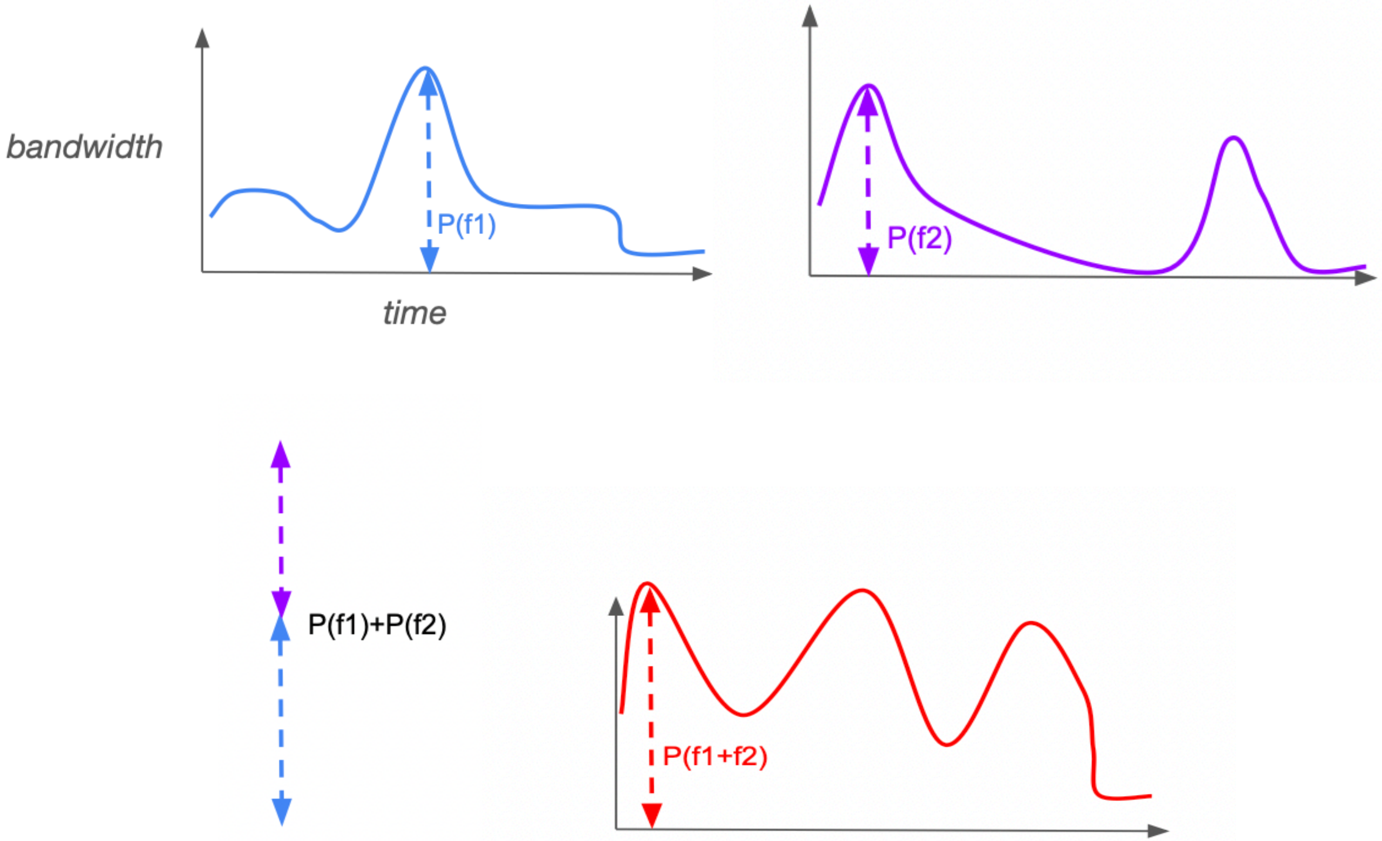
Aggregates, Peaks, etc….

- Peak rate of flow f: P(f)
- Aggregate of peaks: $\Sigma_{\{f\}}$[P(f)]
- Peak of aggregate: P($\Sigma_{\{f\}}$f)
- Typically: $\Sigma_{\{f\}}$[P(f)] >> P($\Sigma_{\{f\}}$f)



Aggregates, Peaks, etc….

- Peak rate of flow f: P(f)
- Aggregate of peaks: $\Sigma_{\{f\}}$[P(f)]
- Peak of aggregate: P($\Sigma_{\{f\}}$f)
- Typically: $\Sigma_{\{f\}}$[P(f)] >> P($\Sigma_{\{f\}}$f)
- Typically: P($\Sigma_{\{f\}}$f) ~ $\Sigma_{\{f\}}$A(f)
  - Where A(f) is the average rate of flow f



Statistical Multiplexing

- Statistical multiplexing merely means that you don’t provision for absolute worst case
  - When everything peaks at the same time

Statistical Multiplexing

- Statistical multiplexing merely means that you don’t provision for absolute worst case
  - When everything peaks at the same time
- Instead, you share resources and hope that peak rates don’t occur at same time

How would you share network resources?

Two approaches to sharing

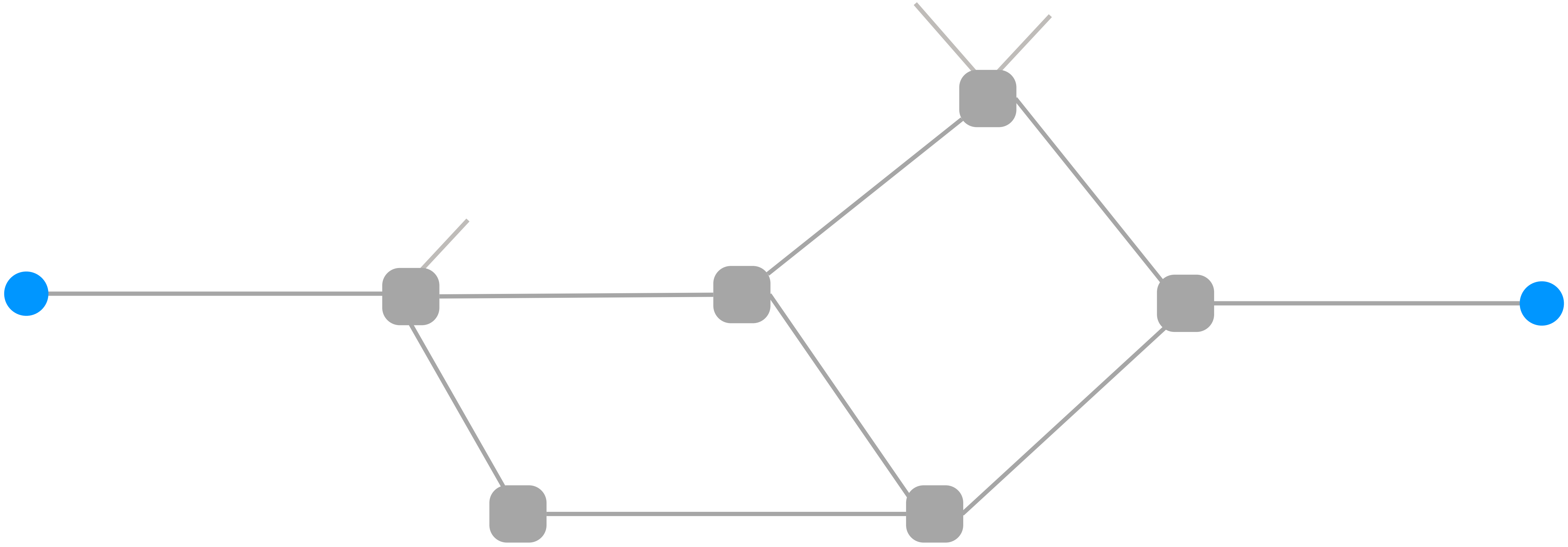
- Reservations: end-hosts explicitly reserve BW when needed (e.g., at the start of a flow)
  - Request/reserve resources
  - Send data
  - Release resources
- Best-effort: just send data packets when you have them and hope for the best ...

Implementing reservations / best-effort sharing

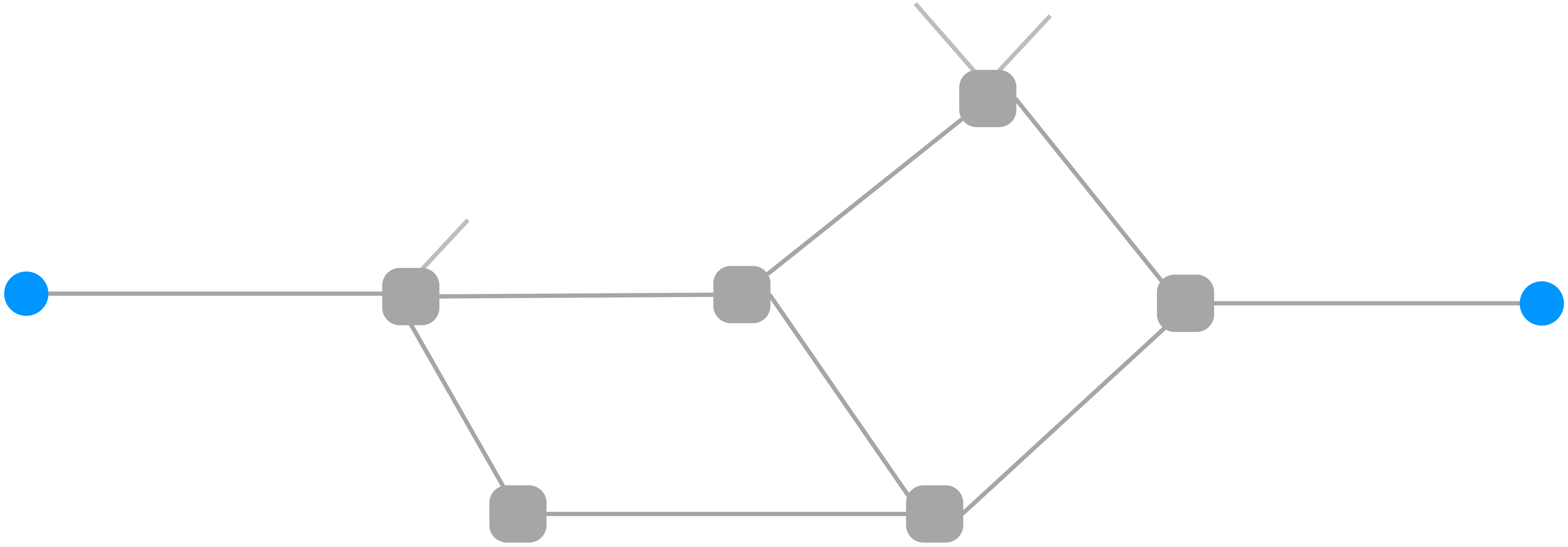
- Many possible approaches!

- Two canonical designs explored in research and industry
  - Reservations via circuit switching
  - Best-effort via packet switching

Reservations: e.g., circuit switching



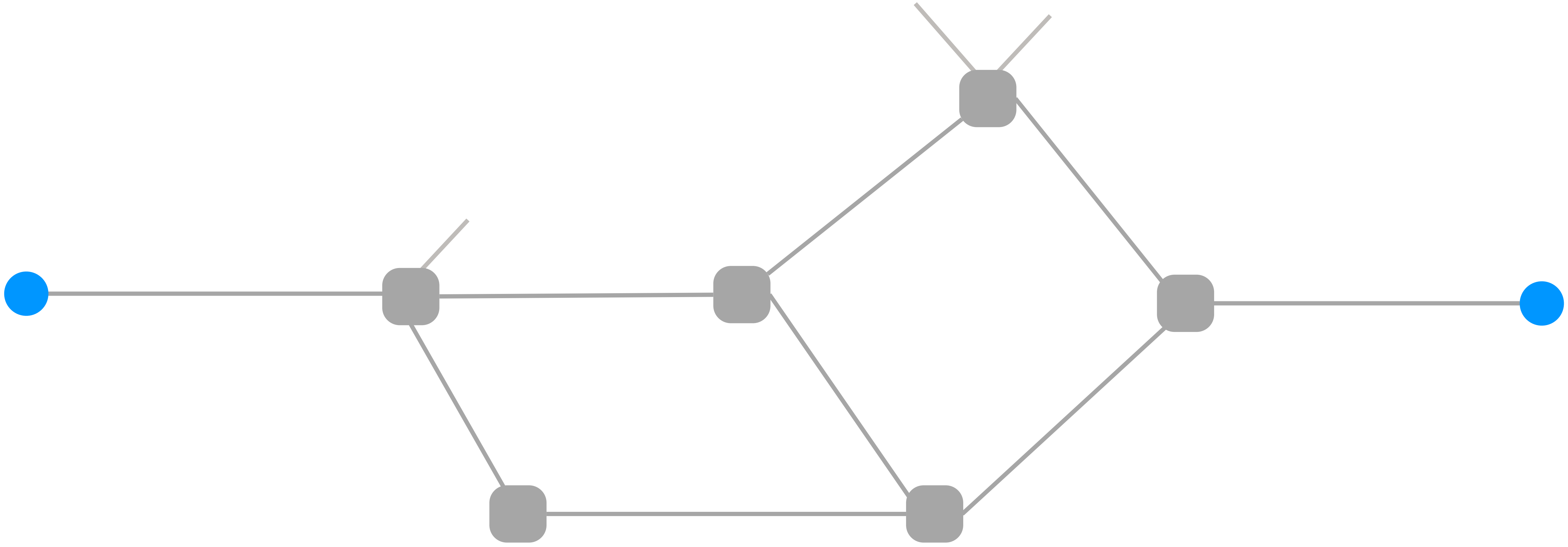
Reservations: e.g., circuit switching



Idea: Reserve network capacity for all packets in a flow

Reservations: e.g., circuit switching

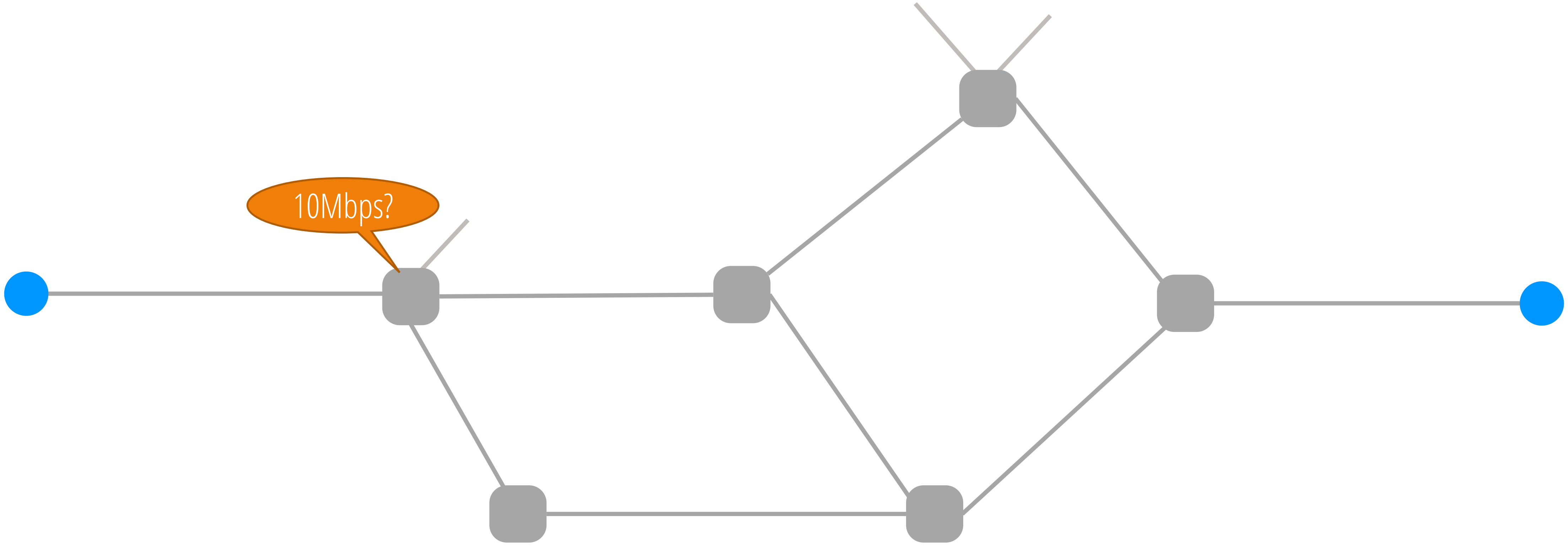
(1) source sends a reservation request to the destination



Idea: Reserve network capacity for all packets in a flow

Reservations: e.g., circuit switching

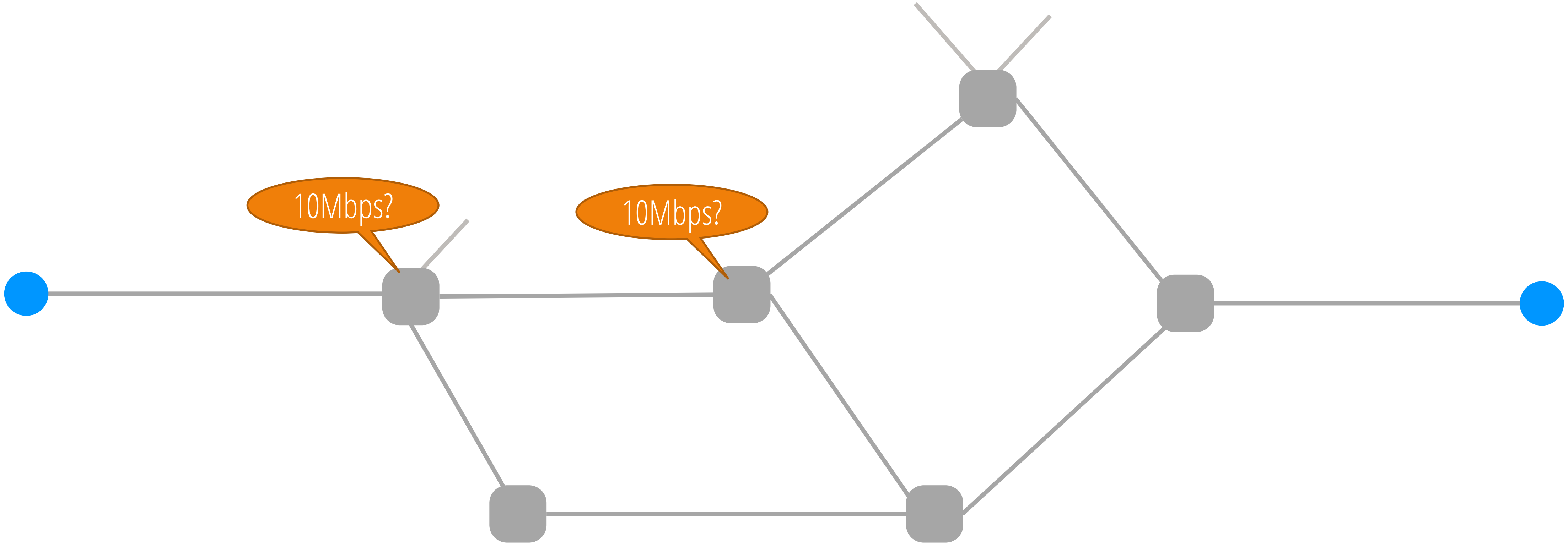
(1) source sends a reservation request to the destination



Idea: Reserve network capacity for all packets in a flow

Reservations: e.g., circuit switching

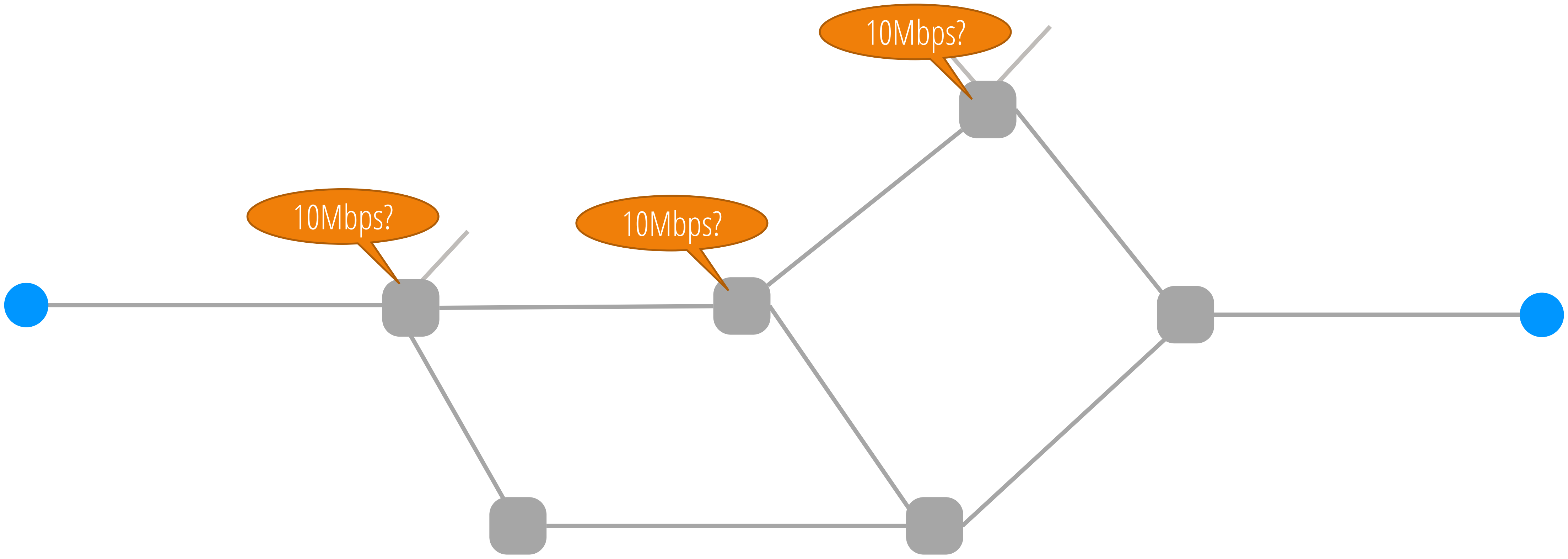
(1) source sends a reservation request to the destination



Idea: Reserve network capacity for all packets in a flow

Reservations: e.g., circuit switching

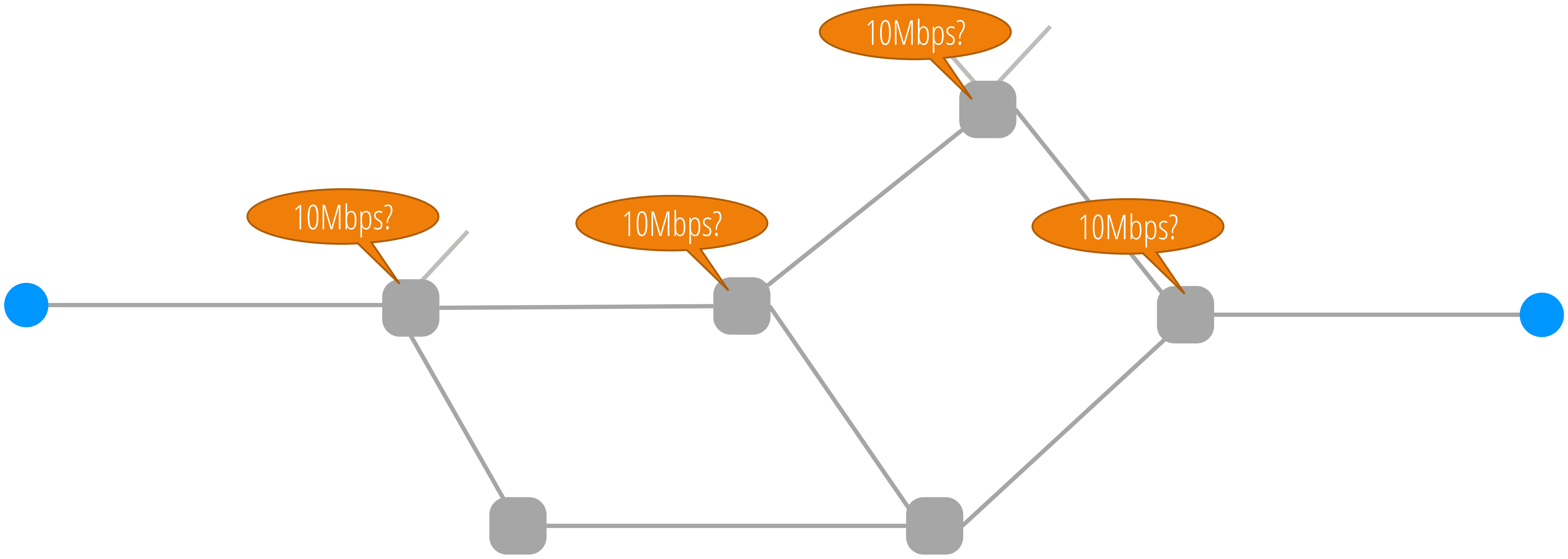
(1) source sends a reservation request to the destination



Idea: Reserve network capacity for all packets in a flow

Reservations: e.g., circuit switching

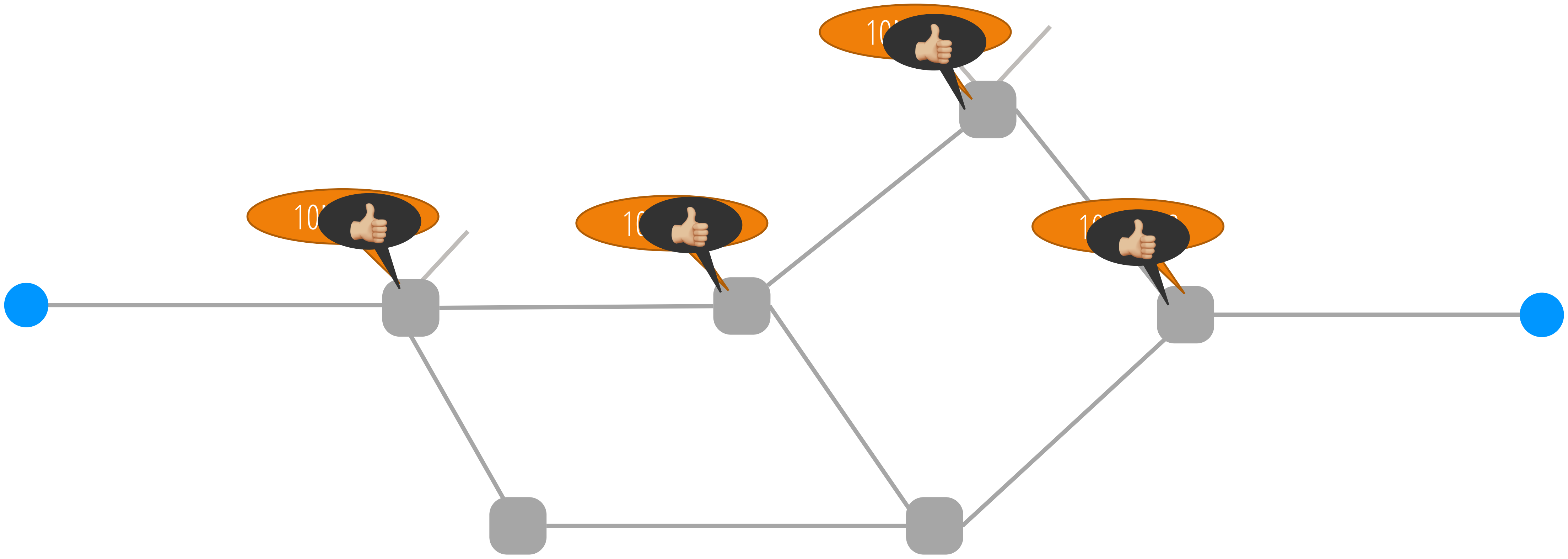
(1) source sends a reservation request to the destination



Idea: Reserve network capacity for all packets in a flow

Reservations: e.g., circuit switching

(1) source sends a reservation request to the destination

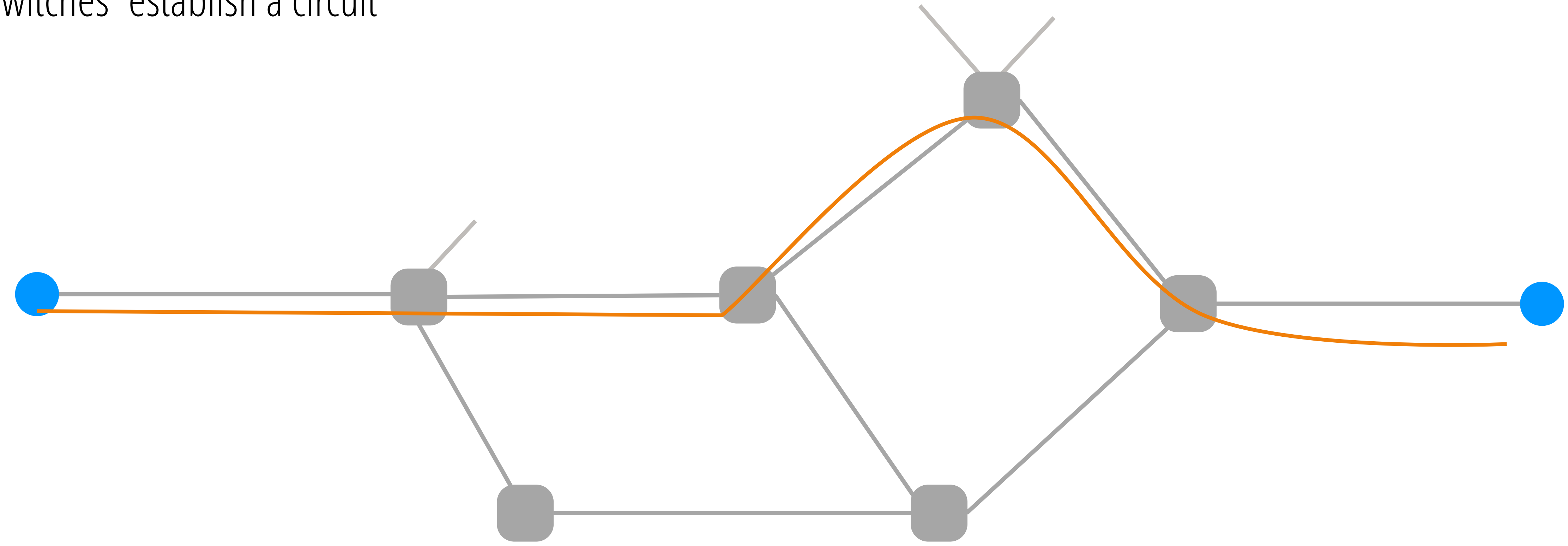


Idea: Reserve network capacity for all packets in a flow

Reservations: e.g., circuit switching

(1) source sends a reservation request to the destination

(2) switches “establish a circuit”



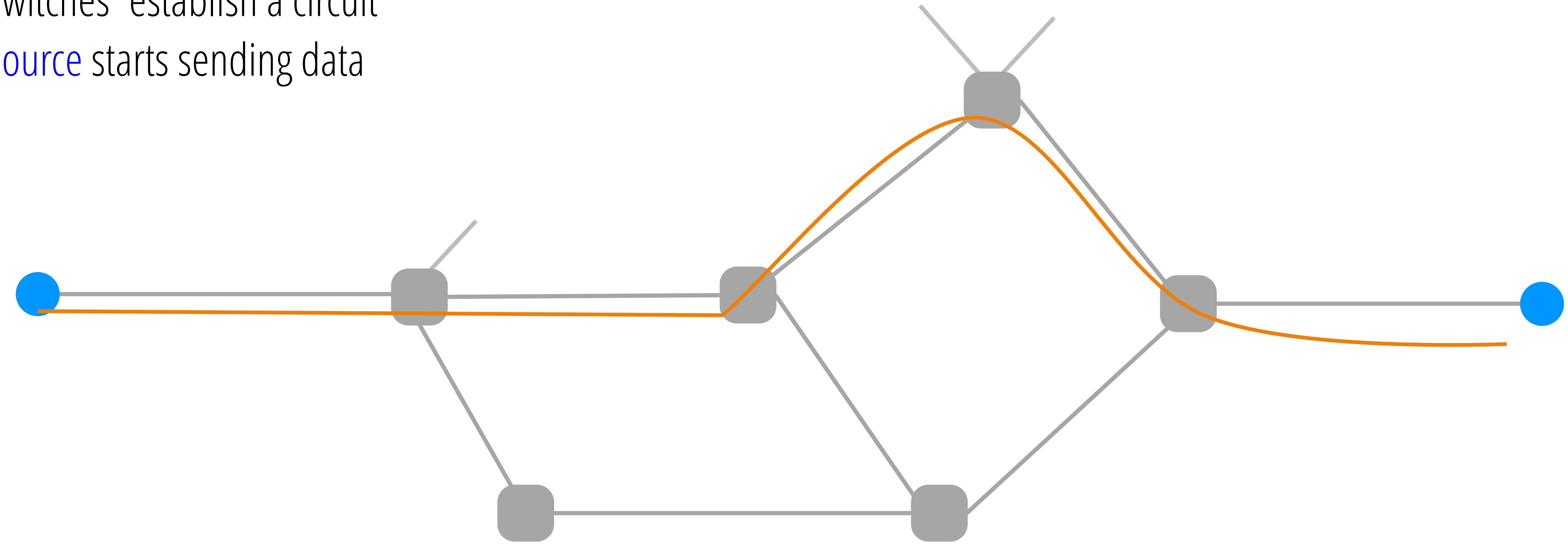
Idea: Reserve network capacity for all packets in a flow

Reservations: e.g., circuit switching

(1) source sends a reservation request to the destination

(2) switches "establish a circuit"

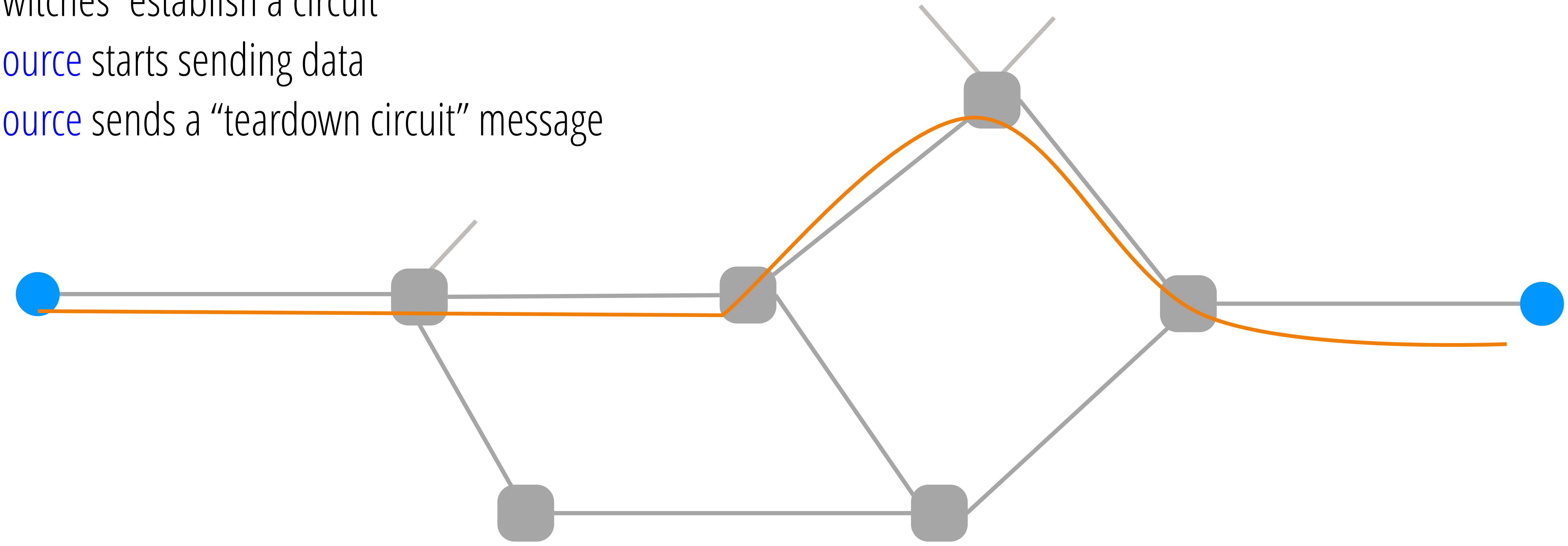
(3) source starts sending data



Idea: Reserve network capacity for all packets in a flow

Reservations: e.g., circuit switching

(1) source sends a reservation request to the destination
(2) switches "establish a circuit"
(3) source starts sending data
(4) source sends a "teardown circuit" message



Idea: Reserve network capacity for all packets in a flow

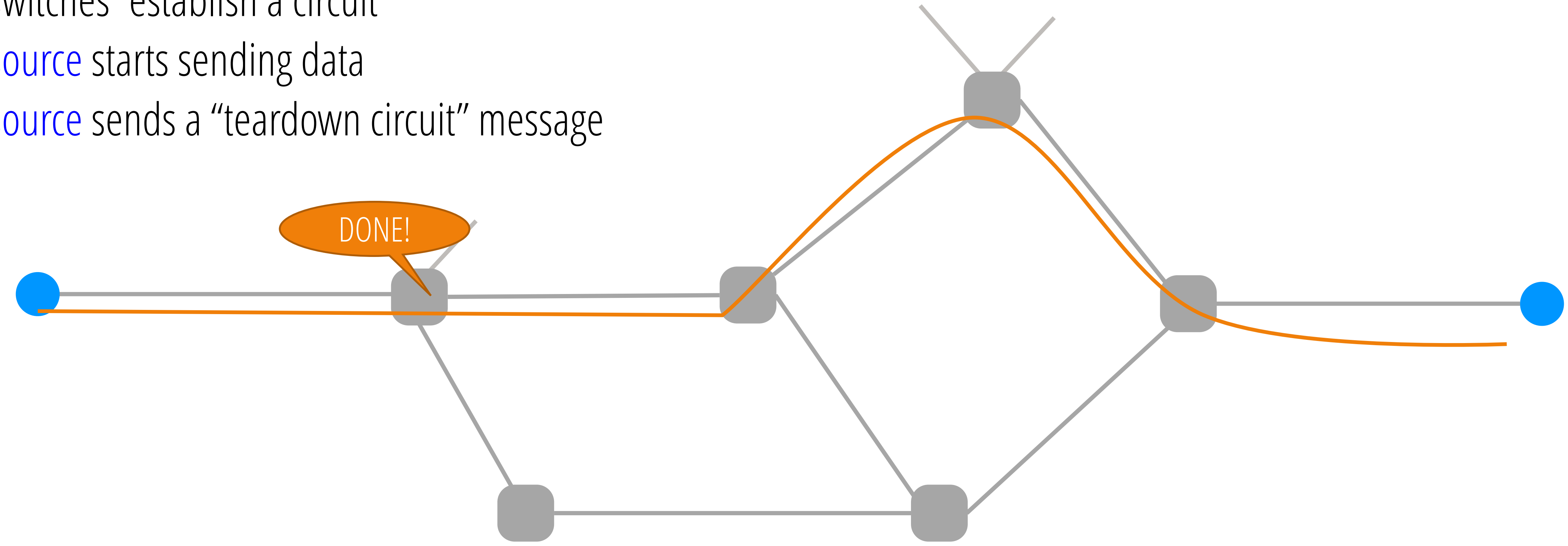
Reservations: e.g., circuit switching

(1) source sends a reservation request to the destination

(2) switches “establish a circuit”

(3) source starts sending data

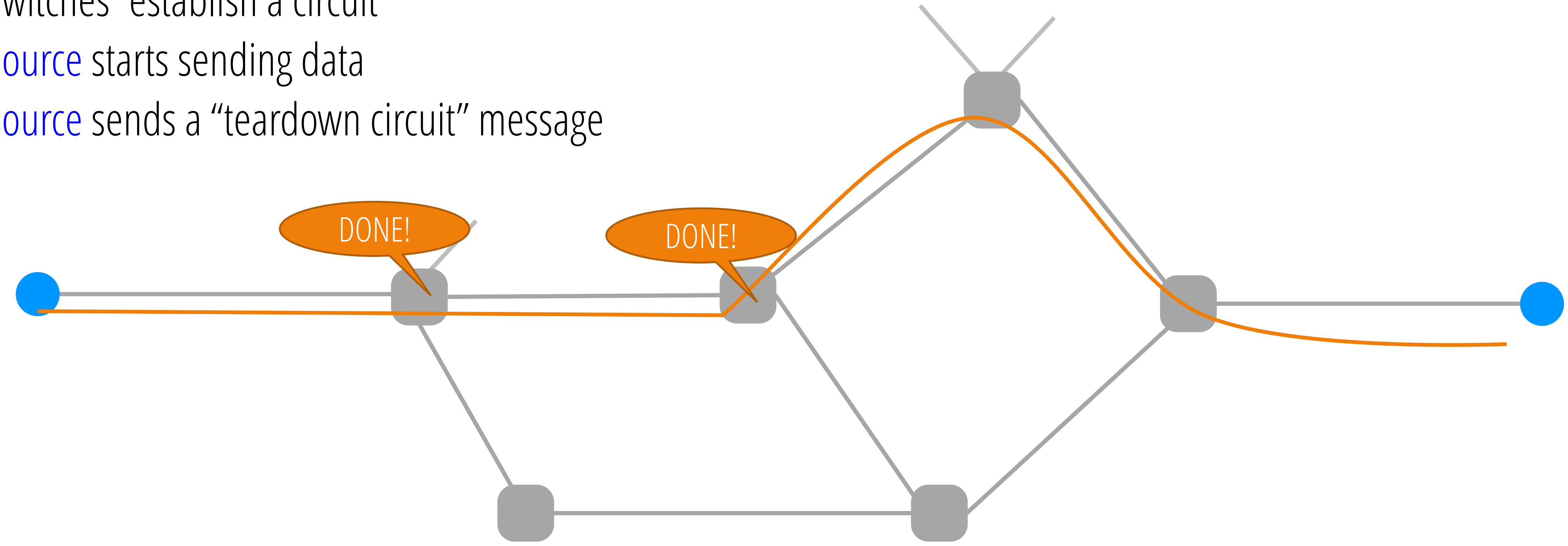
(4) source sends a “teardown circuit” message



Idea: Reserve network capacity for all packets in a flow

Reservations: e.g., circuit switching

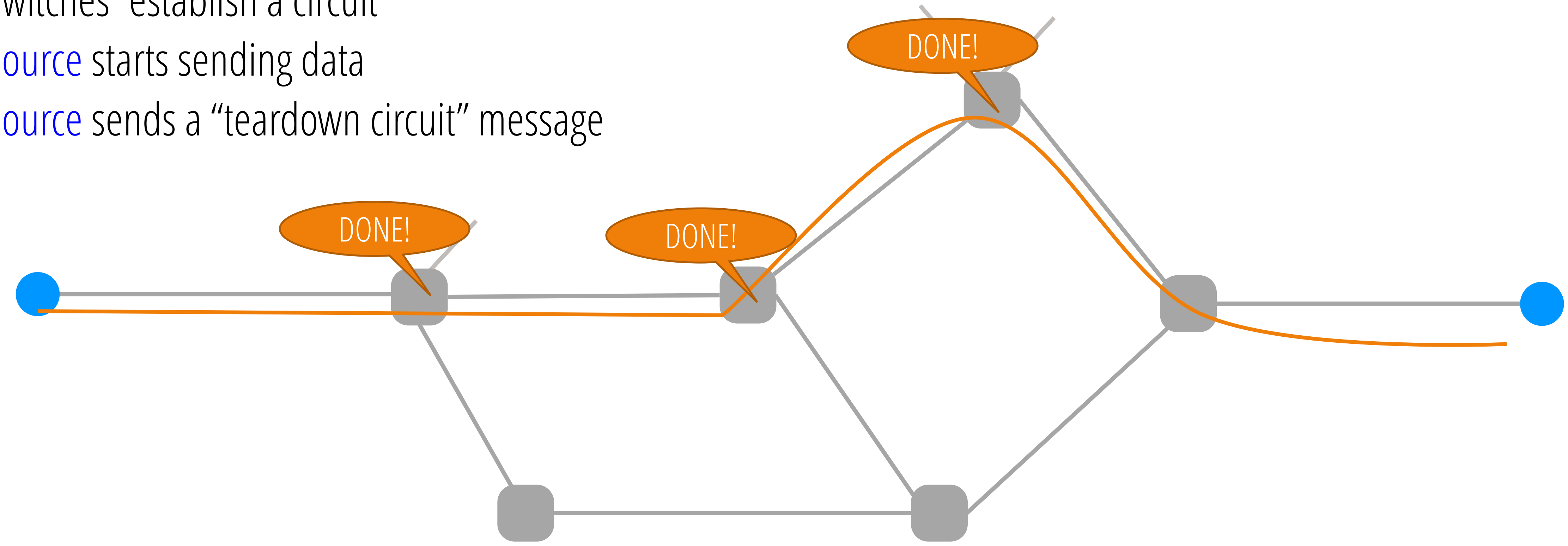
(1) source sends a reservation request to the destination
(2) switches "establish a circuit"
(3) source starts sending data
(4) source sends a "teardown circuit" message



Idea: Reserve network capacity for all packets in a flow

Reservations: e.g., circuit switching

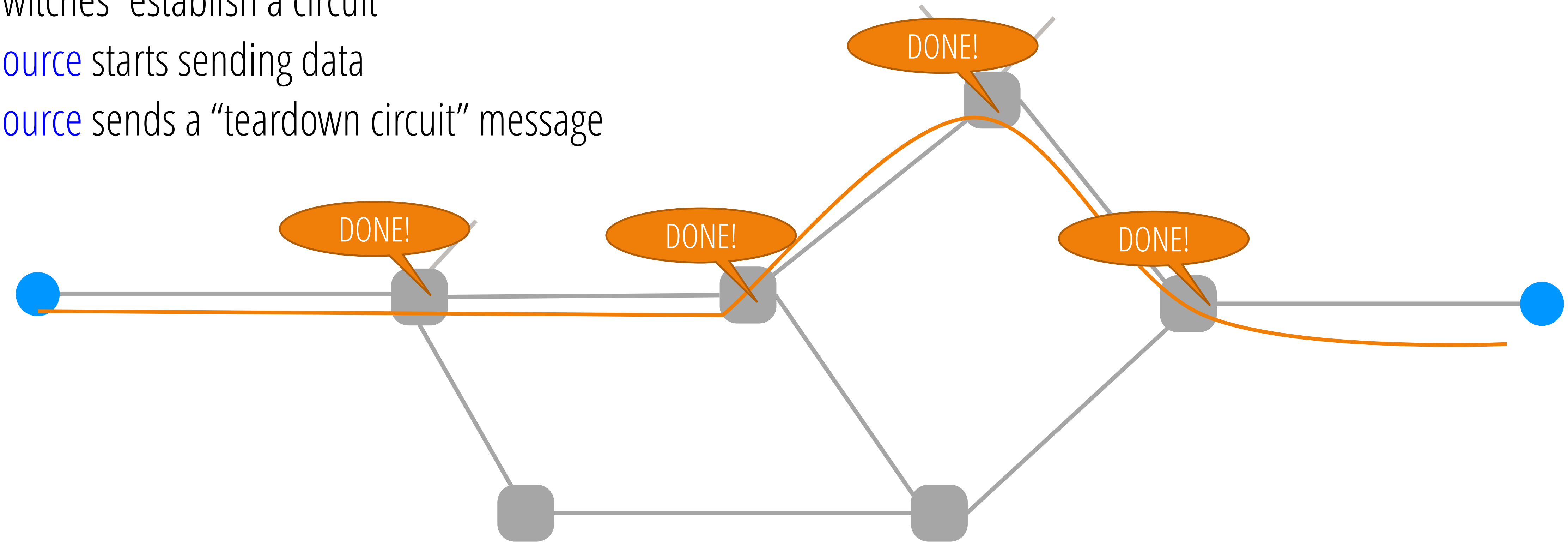
(1) source sends a reservation request to the destination
(2) switches "establish a circuit"
(3) source starts sending data
(4) source sends a "teardown circuit" message



Idea: Reserve network capacity for all packets in a flow

Reservations: e.g., circuit switching

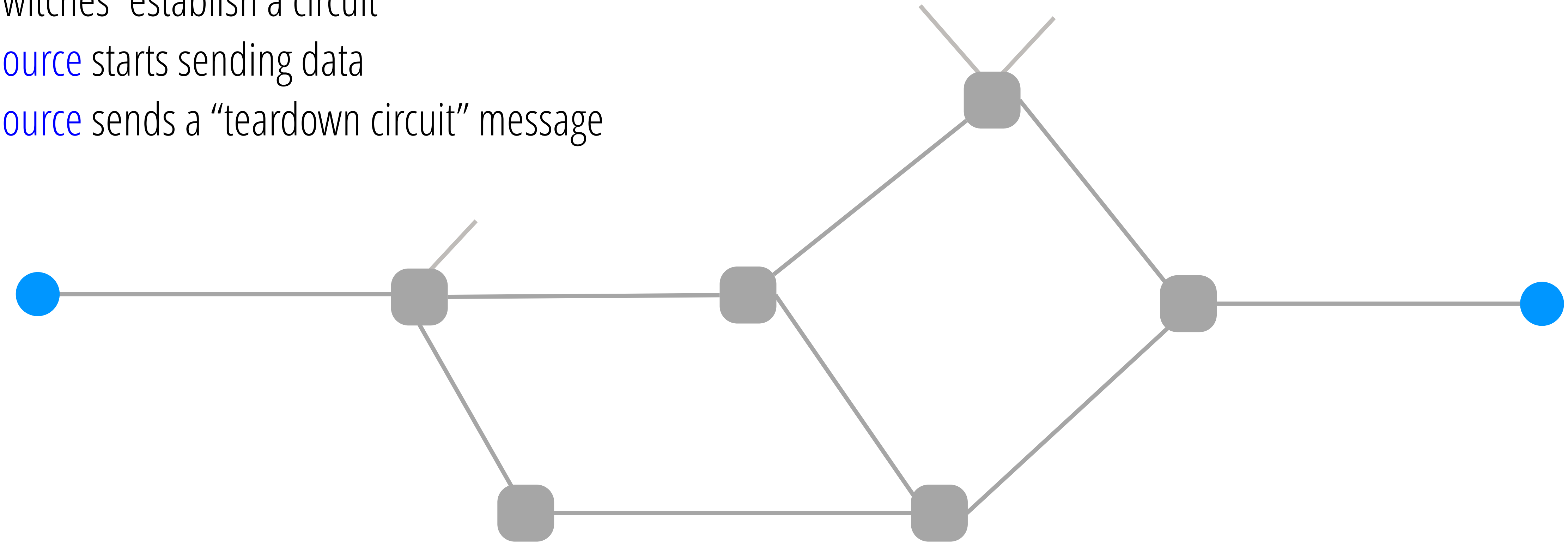
(1) source sends a reservation request to the destination
(2) switches "establish a circuit"
(3) source starts sending data
(4) source sends a "teardown circuit" message



Idea: Reserve network capacity for all packets in a flow

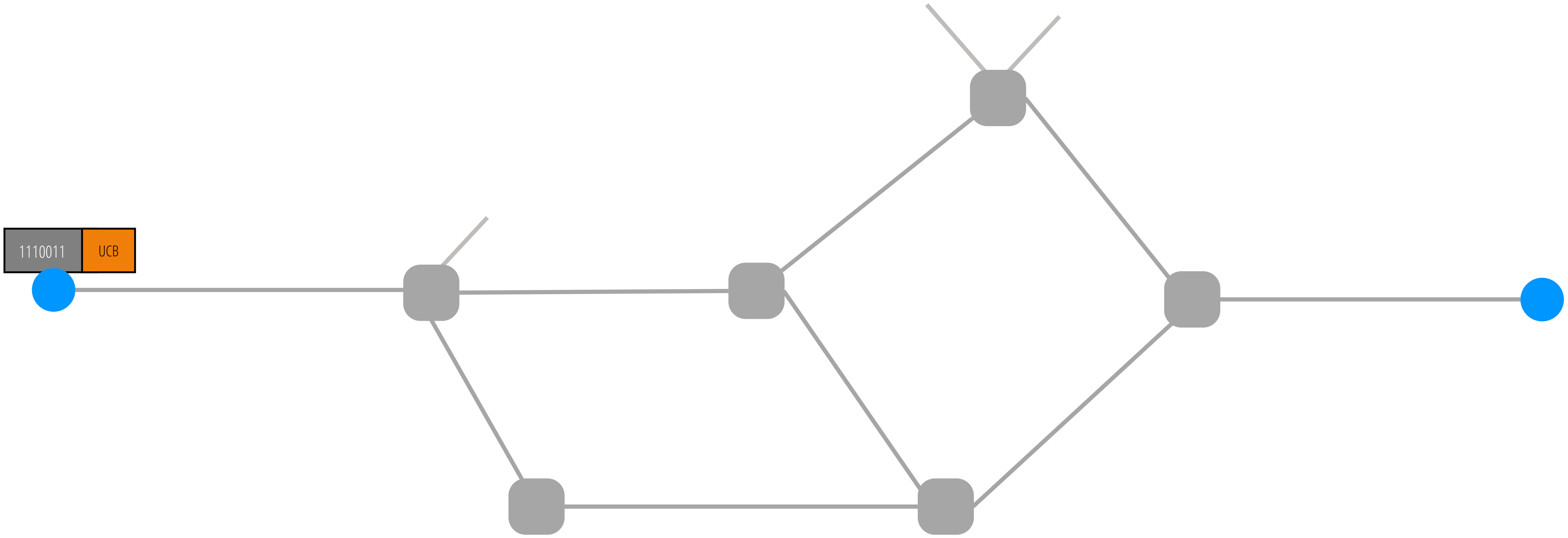
Reservations: e.g., circuit switching

(1) source sends a reservation request to the destination
(2) switches "establish a circuit"
(3) source starts sending data
(4) source sends a "teardown circuit" message

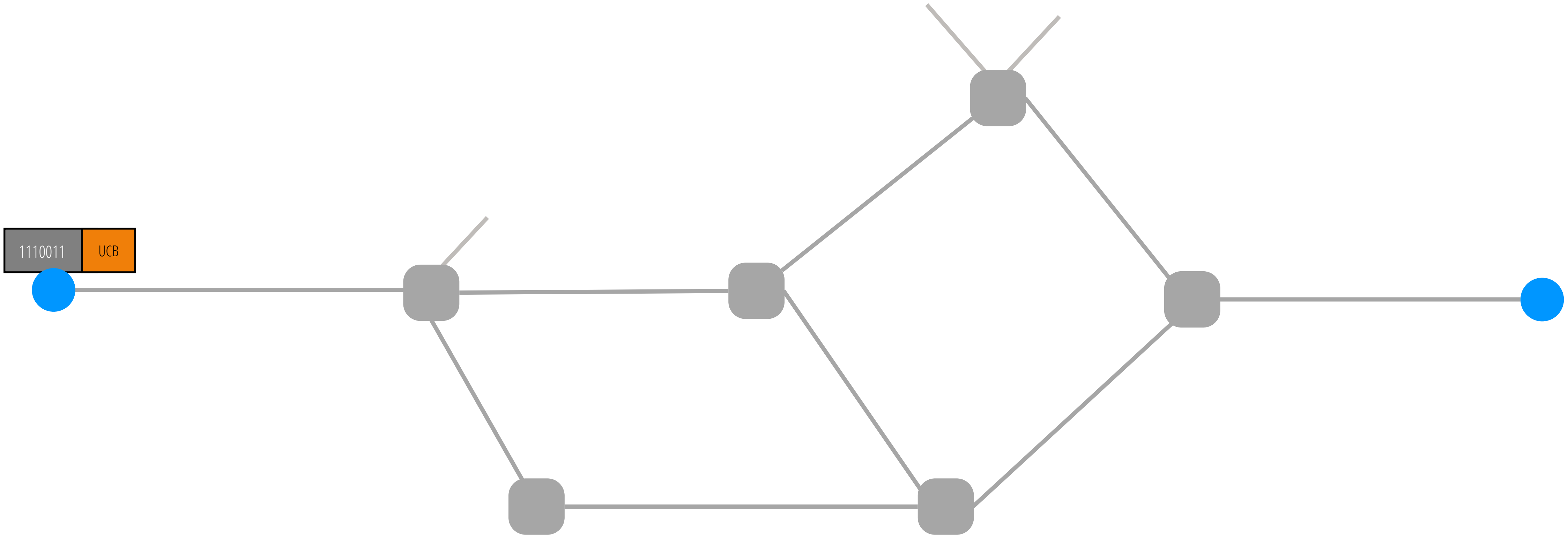


Idea: Reserve network capacity for all packets in a flow

Best-effort: e.g., packet switching



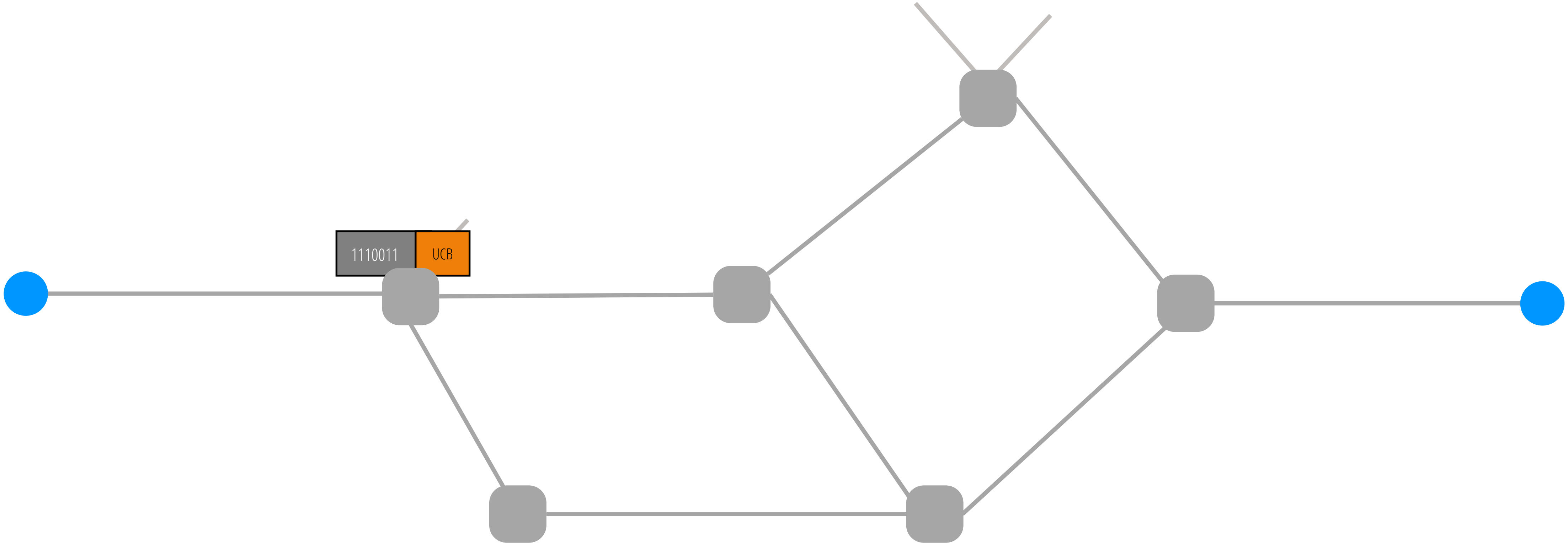
Best-effort: e.g., packet switching



Allocate resources to each packet independently

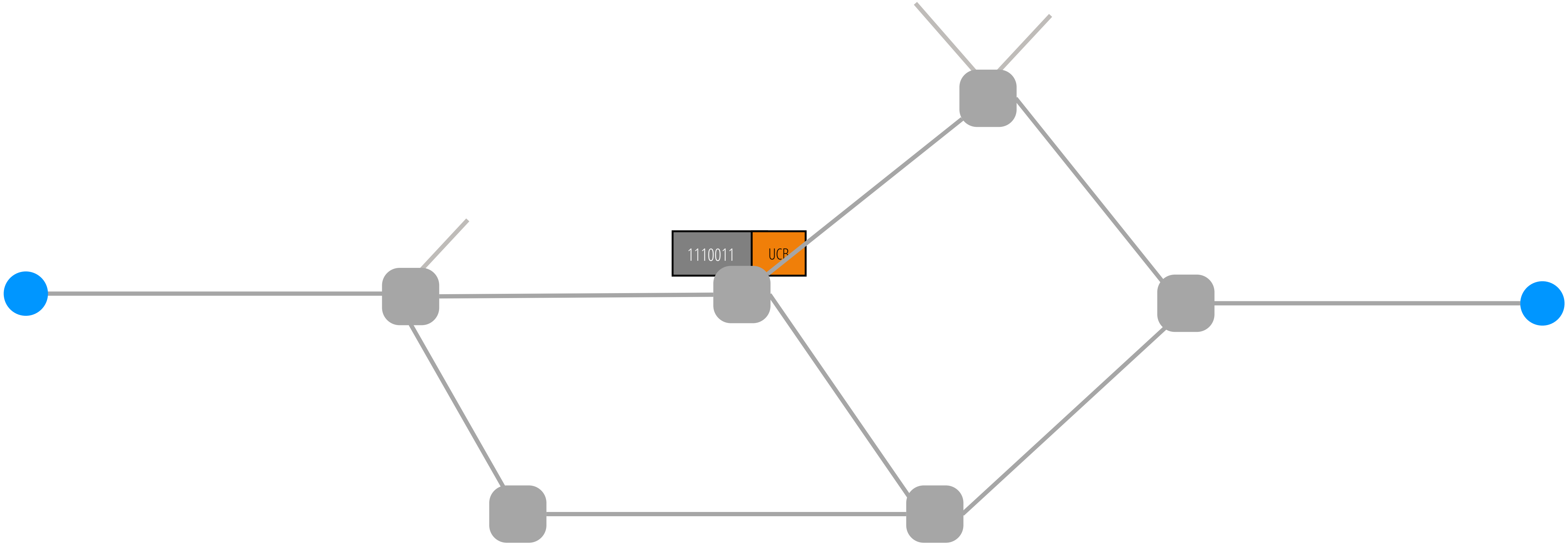
(independent across switches and across packets)

Best-effort: e.g., packet switching



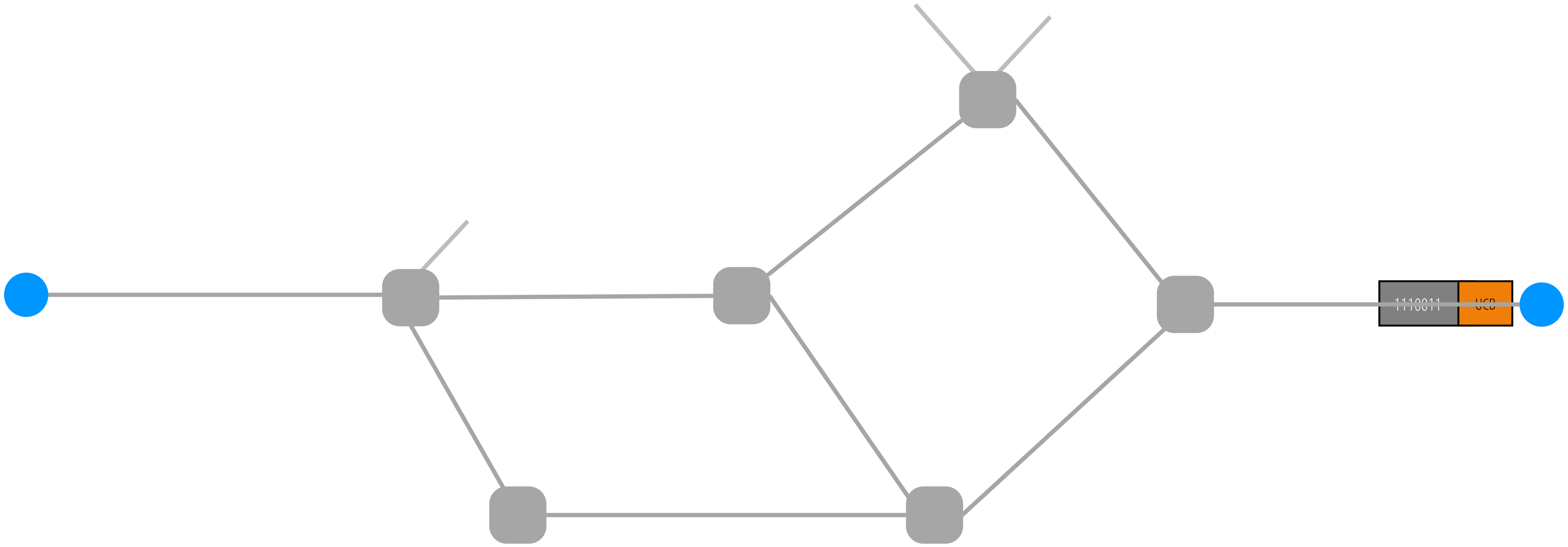
Allocate resources to each packet independently
(independent across switches and across packets)

Best-effort: e.g., packet switching



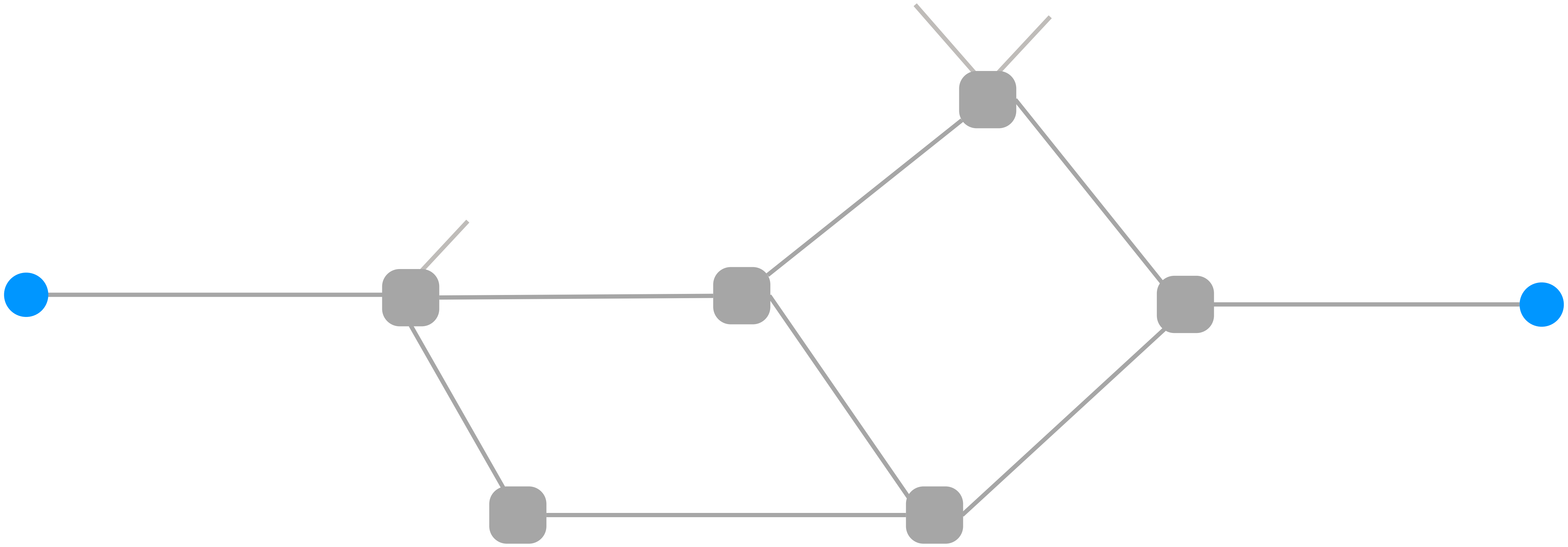
Allocate resources to each packet independently
(independent across switches and across packets)

Best-effort: e.g., packet switching



Allocate resources to each packet independently
(independent across switches and across packets)

Best-effort: e.g., packet switching



Allocate resources to each packet independently
(independent across switches and across packets)

Both approaches embody statistical multiplexing!

Both approaches embody statistical multiplexing!

- Circuit switching: resources shared between *flows* currently in system
  - Reserve the peak demand for a flow
  - But don't reserve for all flows that might ever exist

Both approaches embody statistical multiplexing!

- Circuit switching: resources shared between *flows* currently in system
  - Reserve the peak demand for a flow
  - But don’t reserve for all flows that might ever exist

Both approaches embody statistical multiplexing!

- Circuit switching: resources shared between *flows* currently in system
  - Reserve the peak demand for a flow
  - But don’t reserve for all flows that might ever exist

- Packet switching: resources shared between *packets* currently in system
  - Resources given out on packet-by-packet basis
  - Never reserve resources

Circuit *vs.* Packet switching: which is better?

Circuit *vs.* Packet switching: which is better?

- What are the dimensions along which we should compare?

Circuit *vs.* Packet switching: which is better?

- What are the dimensions along which we should compare?
  - As an abstraction to applications
  - Efficiency (at scale)
  - Handling failures (at scale)
  - Complexity of implementation (at scale)