Taking Stock

- What is a network made of?
- How is it shared?
- How do we evaluate a network?
- How is communication architected?

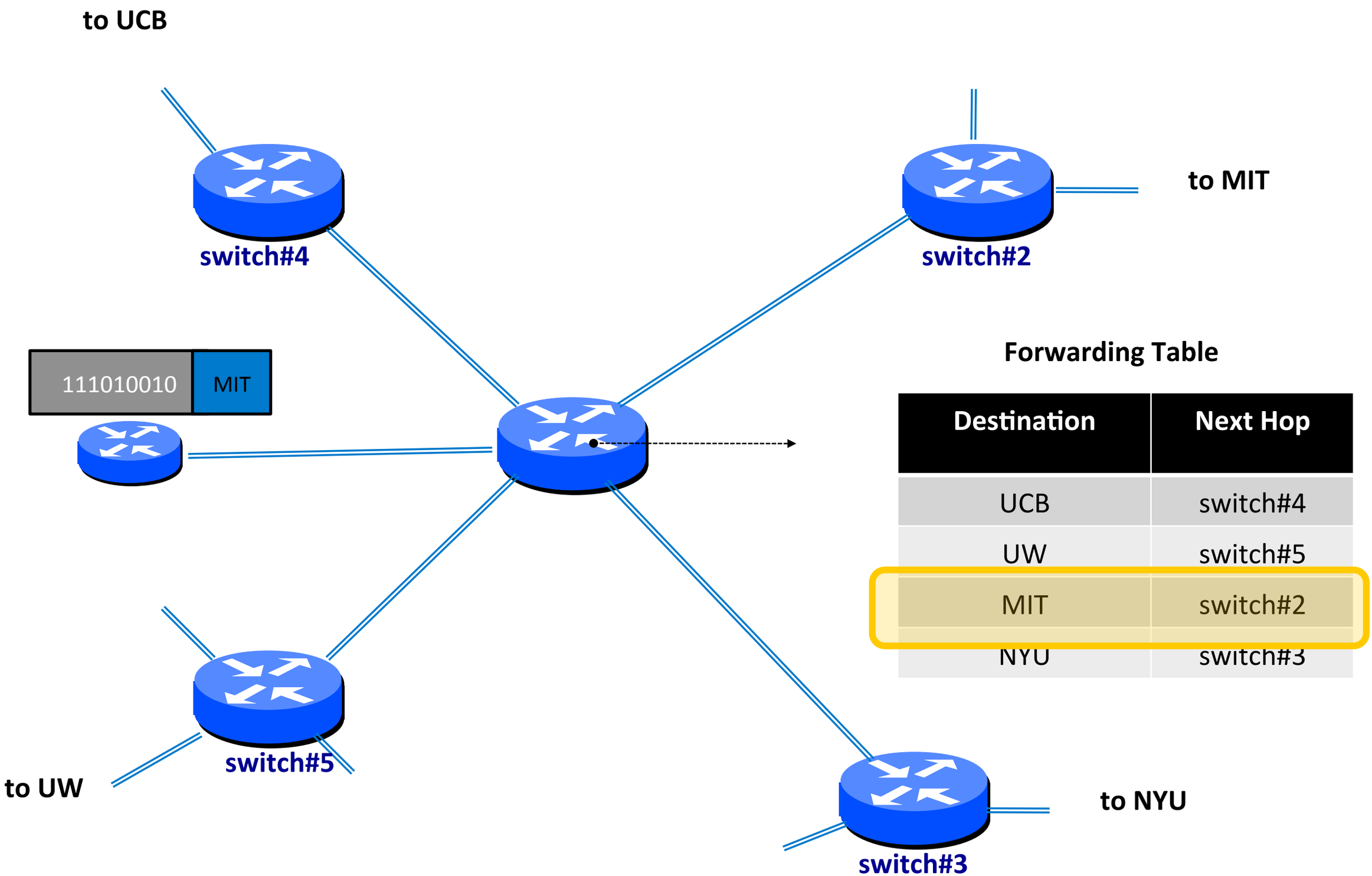
Taking Stock

- Basic concepts
  - *components: end-systems, links, switches, routers*
  - *performance: delay, throughput*
- Basic design choices
  - *packet vs. circuit switching*
  - *Best-effort vs. guaranteed service*
  - *Layering*
  - *“Dumb” network, “smart” ends*

Coming up next

- "How", in detail, by layer
  - *Network layer: next ~two weeks*
  - *Transport layer*
  - Midterm
  - *Application layer (HTTP)*
  - *A little bit of lower layers (Ethernet, Wireless)*
  - *Newer topics (e.g., datacenters, management)*

The Network Layer



Forwarding Table

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

Three topics

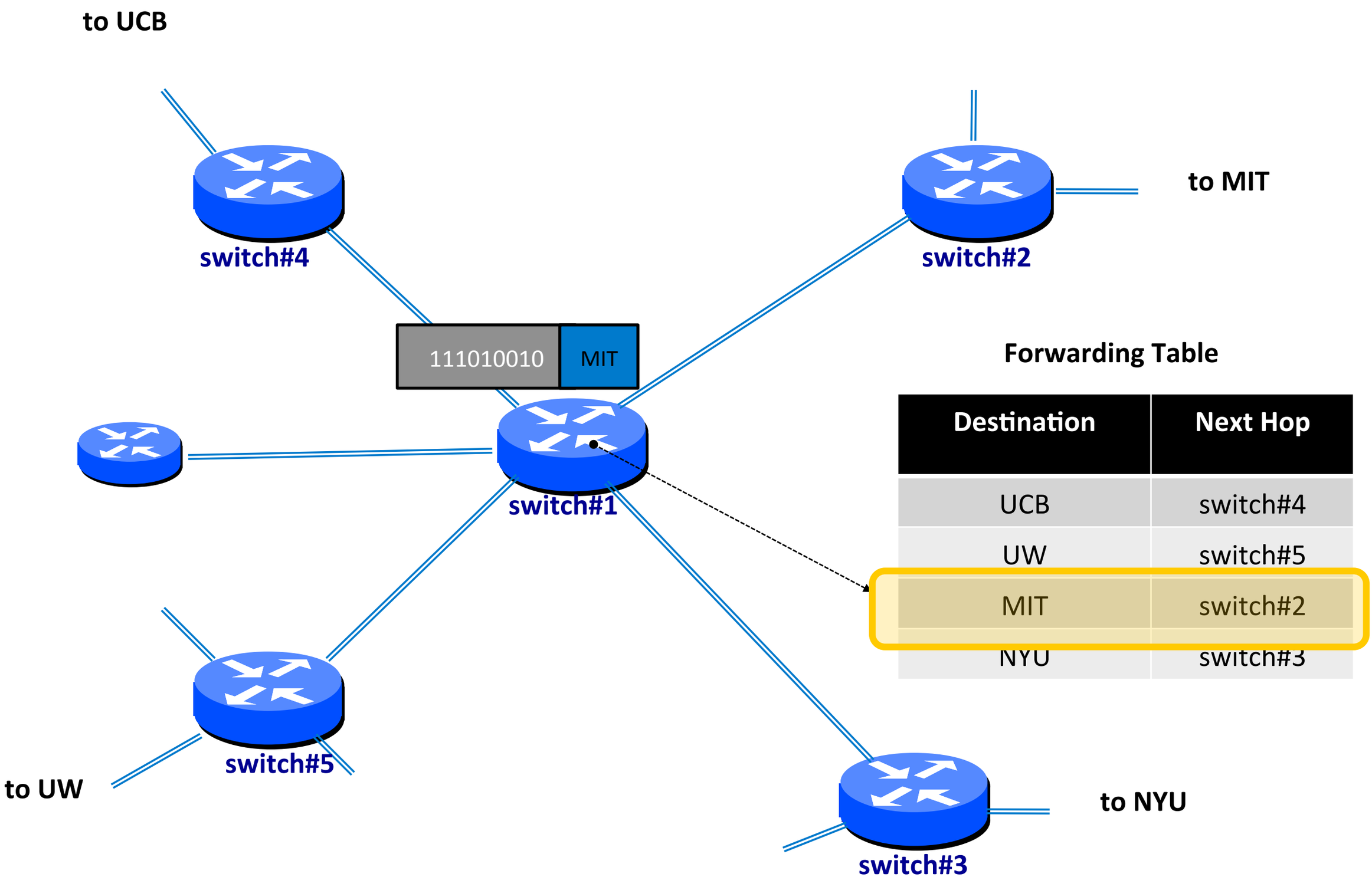
- Addressing
- Forwarding
- Routing

Addressing (for now)

- Assume each end-system has a unique address
- No particular structure to these addresses
- Will cover IP addresses later in the course

Forwarding

- **Local** router process that determines the output link (a.k.a “next hop”) for each packet
- How
  - *read address from packet’s network layer header*
  - *search forwarding table*



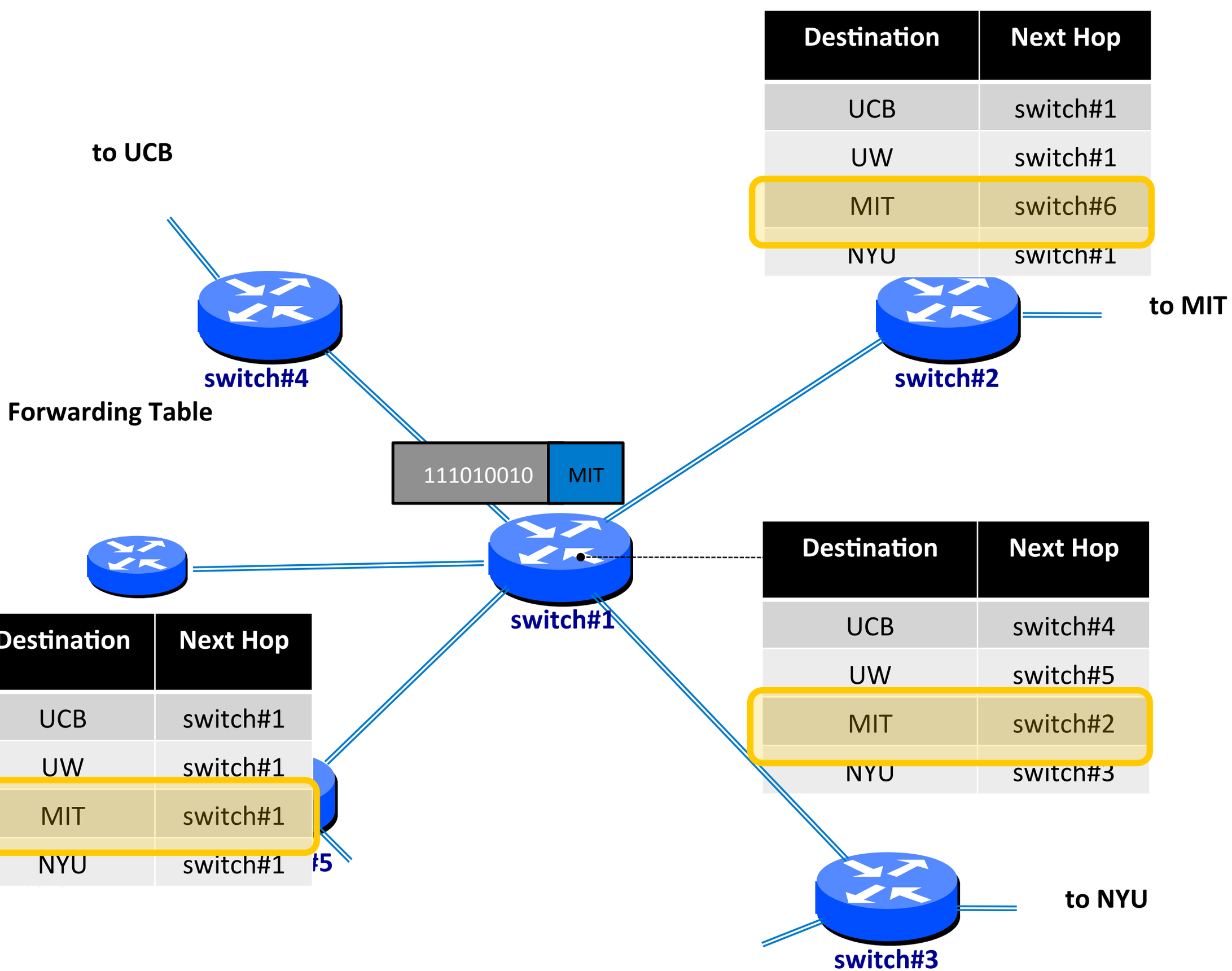
Forwarding Table

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

Routing

- **Network-wide** process that determines the content of forwarding tables

  → *determines the end-to-end path for each destination*



Destination	Next Hop
UCB	switch#1
UW	switch#1
MIT	switch#6
NYU	switch#1

Destination	Next Hop
UCB	switch#1
UW	switch#1
MIT	switch#1
NYU	switch#1

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

Routing

- **Network-wide** process that determines the content of forwarding tables

  → *determines the end-to-end path for each destination*

- How

  - *coming up soon*

Forwarding vs. Routing

- Forwarding: "data plane"
  - *Directing one data packet*
  - *Each router using local routing state*

- Routing: "control plane"
  - *Computing the forwarding tables that guide packets*
  - *Jointly computed by routers using a distributed algorithm*

- Very different timescales!

Routing Fundamentals

- Validity of routing state

Goal (v1)

- Find a path to a given destination
- How do we know that the state contained in forwarding tables meets our goal?
  - *this is what "validity" of routing state tells us*
  - *[this is non-standard terminology]*

Local *vs.* Global View of State

- *Local* routing state is the forwarding table in a single router
  - By itself, the state in a single router can't be evaluated
  - It must be evaluated in terms of the global context

Destination	Next Hop
UCB	switch#1
UW	switch#1
MIT	switch#1
NYU	switch#1

to UCB

switch#4

Forwarding Table

111010010 | MIT

switch#1

switch#2

to MIT

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

Destination	Next Hop
UCB	switch#1
UW	switch#1
MIT	switch#1
NYU	switch#1

switch#3

to NYU

Local *vs.* Global View of State

- *Local* routing state is the forwarding table in a single router
  - By itself, the state in a single router can't be evaluated
  - It must be evaluated in terms of the global context

- *Global* state refers to the collection of forwarding tables in each of the routers
  - Global state determines which paths packets take

(Will discuss later where this routing state comes from)

"Valid" Routing State

- Global state is "valid" if it produces forwarding decisions that always deliver packets to their destinations

- Goal of routing protocols: compute valid state
  - But how can you tell if routing state if valid?

- Need a succinct correctness condition for routing
  - Suggestions?

Necessary and Sufficient Condition

- Global routing state is valid ***if and only if***:
  - There are no dead ends (other than destination)
  - There are no loops

- A dead end is when there is no outgoing link (next-hop)
  - A packet arrives, but the forwarding decision does not yield any outgoing link

- A loop is when a packet cycles around the same set of nodes forever

Loop!

to UCB

Destination	Next Hop
UCB	switch#1
UW	switch#1
MIT	switch#1
NYU	switch#1

to MIT

switch#4

switch#2

Forwarding Table

111010010	MIT

switch#1

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

Destination	Next Hop
UCB	switch#1
UW	switch#1
MIT	switch#1
NYU	switch#1

switch#5

switch#3

to NYU

Deadend (to MIT)

Destination	Next Hop
UCB	switch#1
UW	switch#1
MIT	switch#1
NYU	switch#1

to UCB

switch#4

Forwarding Table

111010010	MIT

to MIT

switch#2

switch#1

Destination	Next Hop
UCB	switch#4
UW	switch#5
NYU	switch#3

Destination	Next Hop
UCB	switch#1
UW	switch#1
MIT	switch#1
NYU	switch#1

#5

switch#3

to NYU

Necessary and Sufficient Condition

- Global routing state is valid ***if and only if***:
  - There are no dead ends (other than destination)
  - There are no loops

Necessary (“only if”): Easy

- If you run into a deadend before hitting destination, you’ll never reach the destination
- If you run into a loop, you’ll never reach destination

Sufficient ("if"):

- Assume no deadends, no loops
- Packet must keep wandering, but without repeating
  - If ever enter same switch from same link, will loop
- Only a finite number of possible links for it to visit
  - It cannot keep wandering forever without looping
  - Must eventually hit destination

Checking Validity of Routing State

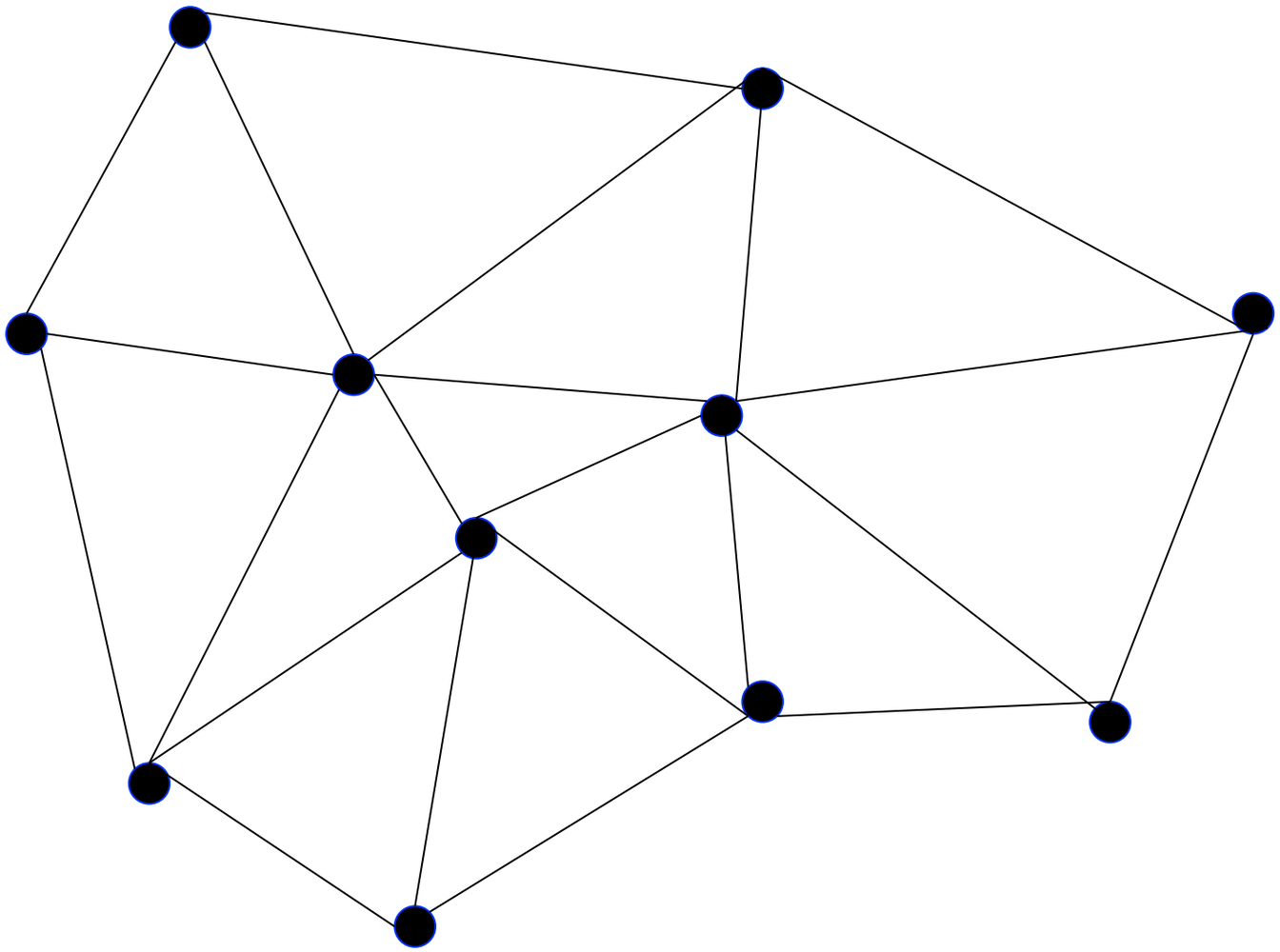
- Focus only on a single destination
  - Ignore all other routing state

- Mark outgoing link ("next hop") with arrow
  - There is only one at each node

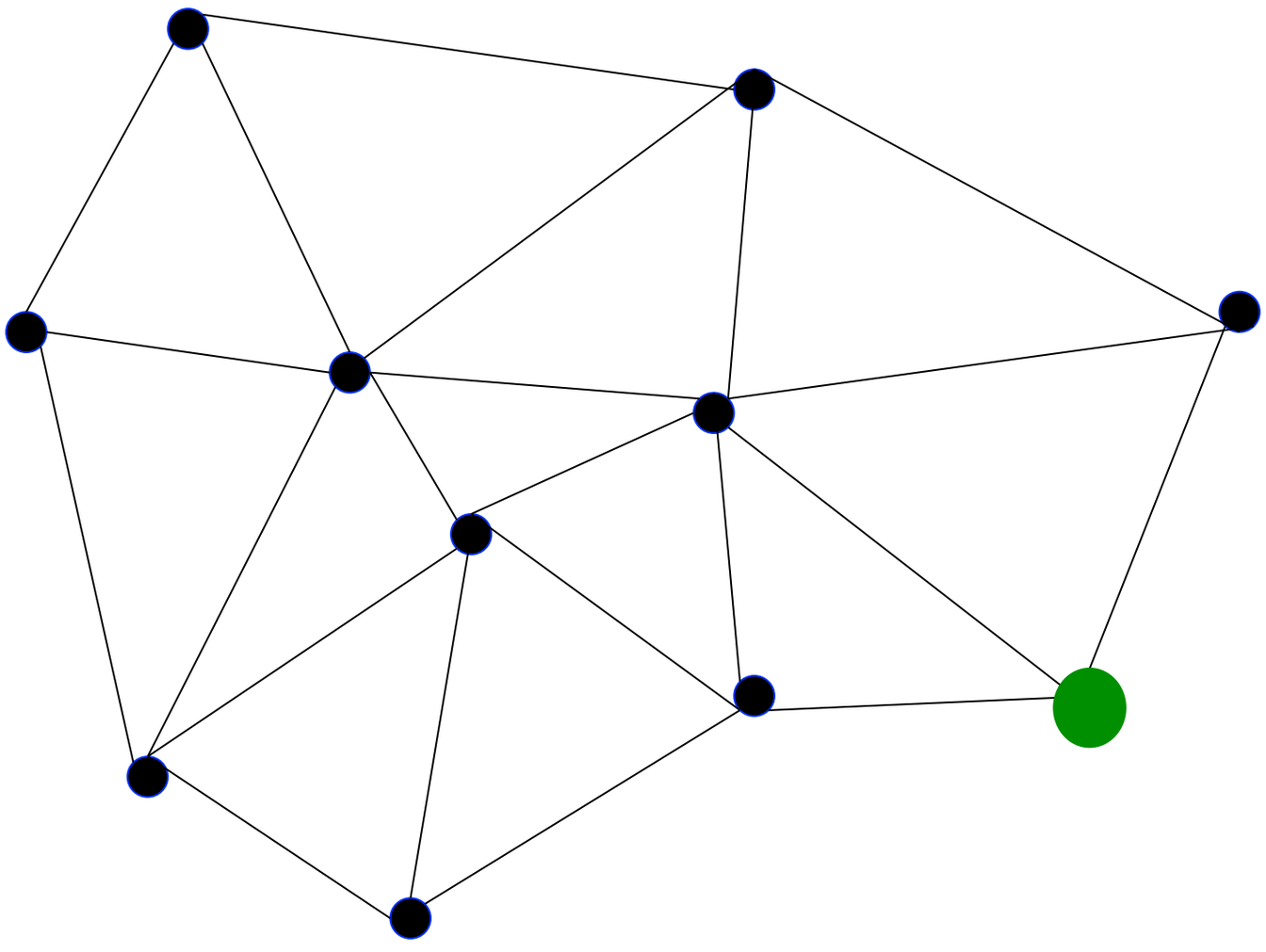
- Eliminate all links with no arrows

- Look at what's left….

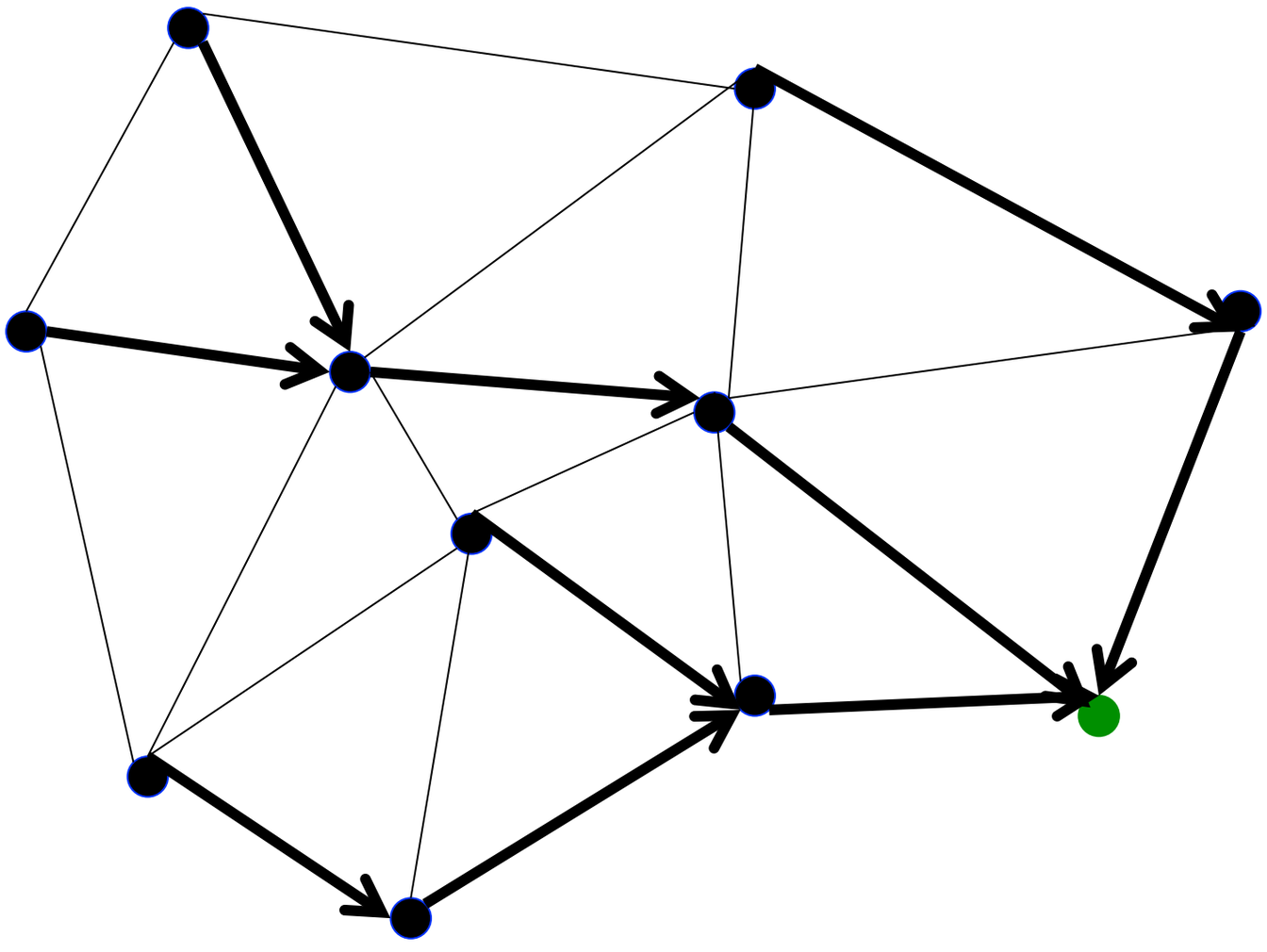
Example 1



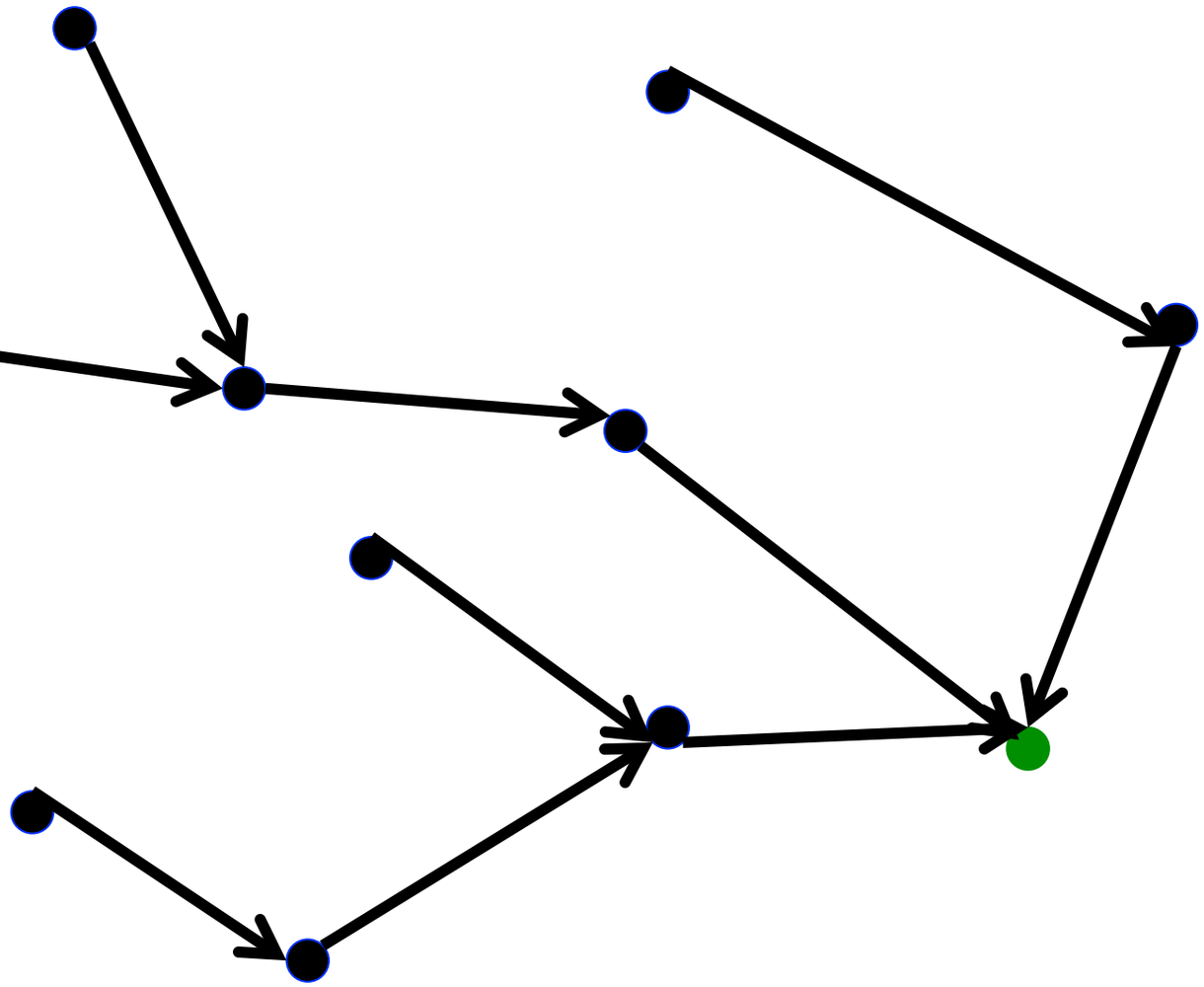
Pick Destination



Put arrows on outgoing links (to green dot)

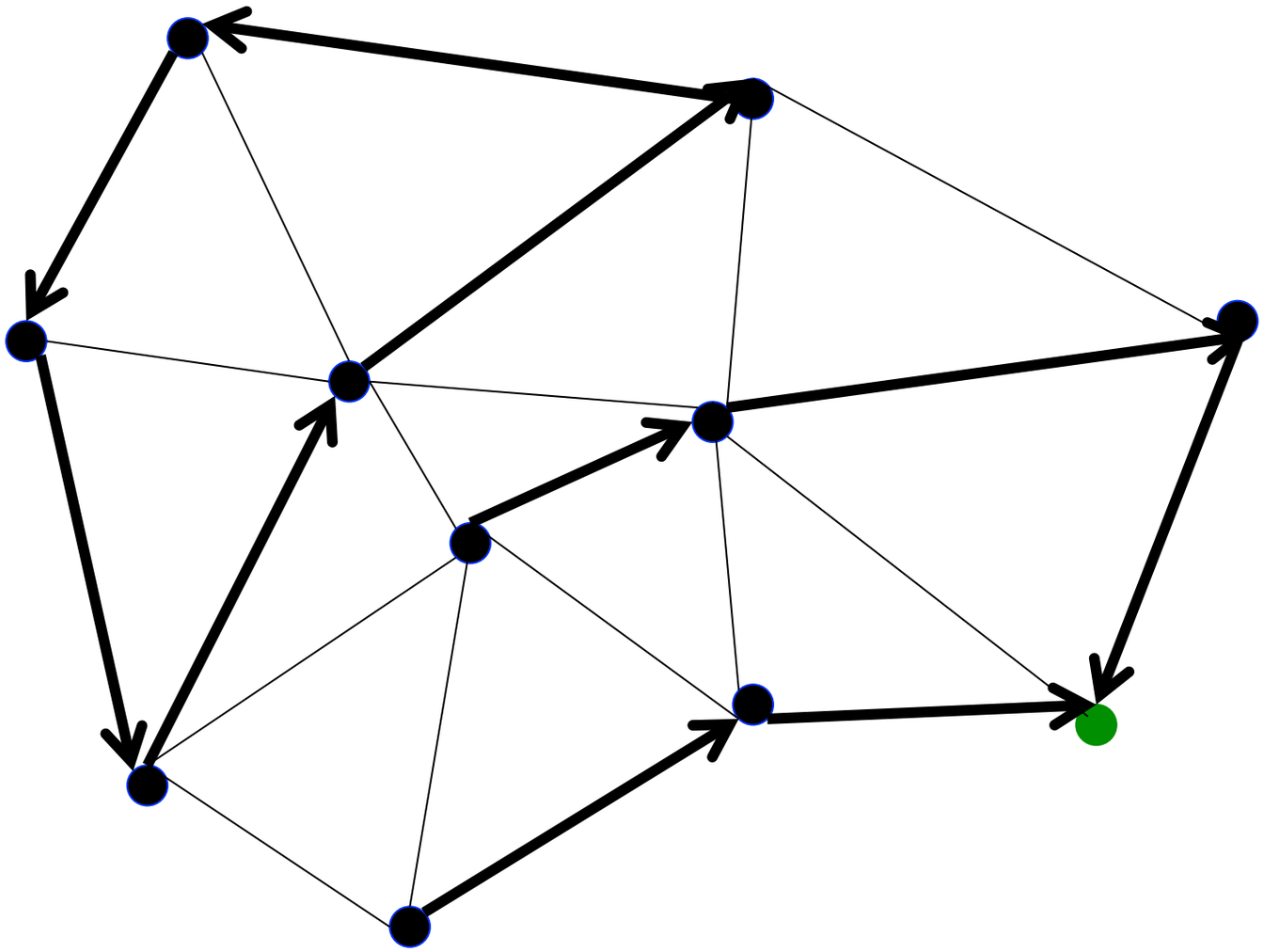


Remove Unused Links

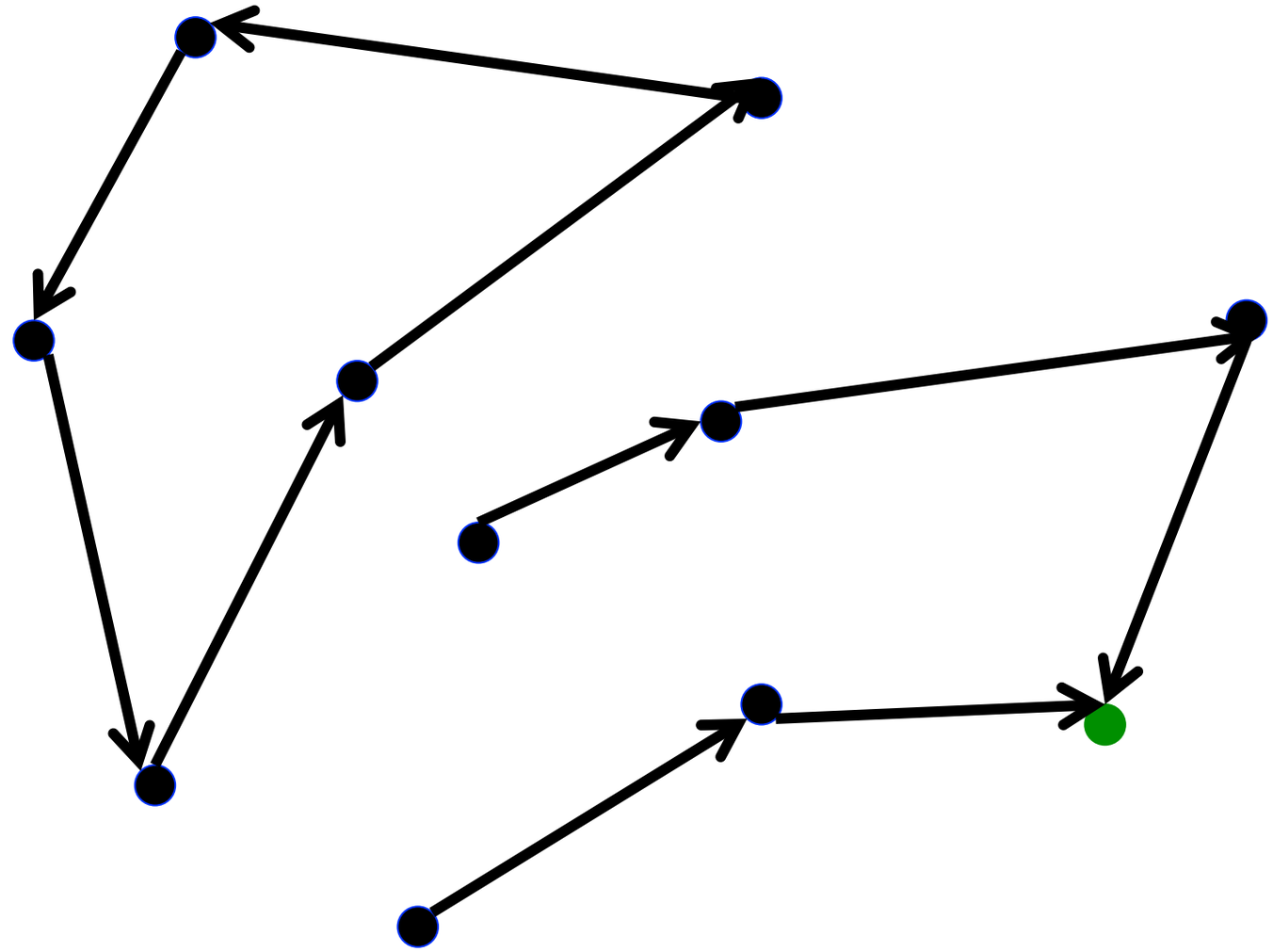


Leaves Spanning Tree: Valid

Second Example



Second Example



Is this valid?

Lesson….

- Very easy to check validity of routing state for a particular destination

- Deadends are obvious
  - Node without outgoing arrow

- Loops are obvious
  - Disconnected from rest of graph

Goal (for routing)

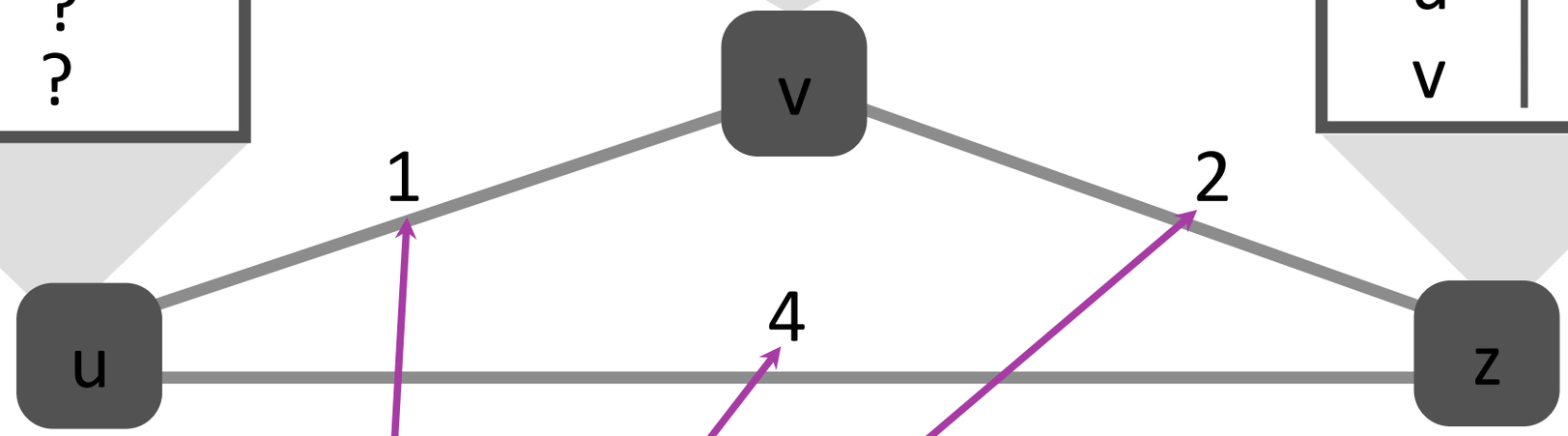
v1: Find a path to a given destination

v2: Find a *least cost path* to a given destination

dest.	next hop
z	?
u	?

dest.	next hop
z	?
v	?

dest.	next hop
u	?
v	?



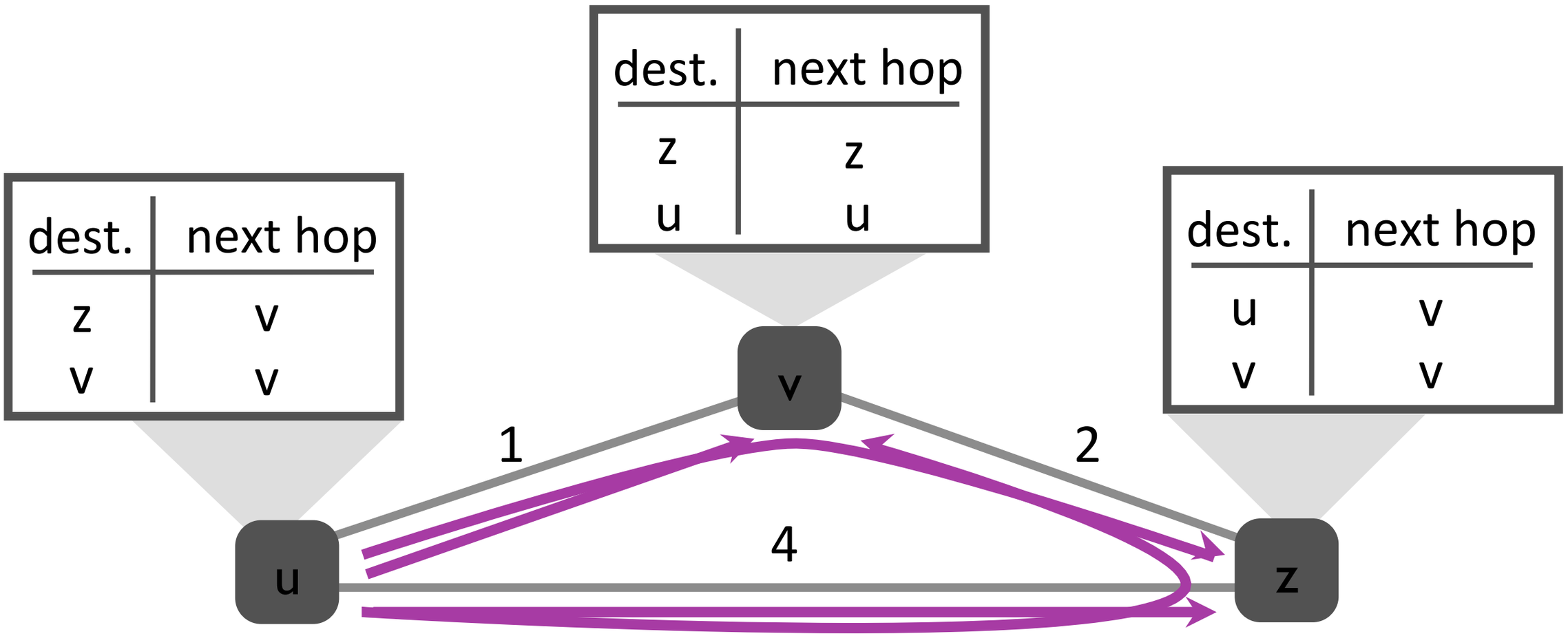
link costs

could represent:

-- propagation delay

-- load

-- cost



least-cost path from u to z: u v z

least cost path from u to v: u v

Least-cost path routing

- Given: router graph & link costs
- Goal: find least-cost path
  from each source router
  to each destination router

Next time: how we compute these