

CS 181AG  
Lecture 15

# Midterm Review

Arthi Padmanabhan

Oct 24, 2022

# Lecture Material – Quick Reference

- Lecture 1:
  - Intro, circuit switched vs. packet switched networks, intro to layered architecture
- Lecture 2:
  - Local Area Network communication over a single wire: collisions, Aloha, Slotted Aloha, p-persistence, exponential backoff, collision detection, wireless networks
- Lecture 3:
  - Moving beyond one wire: hubs, bridges, selective forwarding, Spanning Tree Protocol
- Lecture 4
  - Intro to Network Layer: IP addresses, public vs private IP, Address Resolution Protocol, routers, forwarding information base

# Lecture Material – Quick Reference

- Lecture 5
  - Routing Protocols: Distance Vector (Bellman-Ford), count-to-infinity, poison reverse
- Lecture 6
  - Routing Protocols: Review of Spanning Tree Protocol, Link State (Dijkstra's), temporary loops
- Lecture 7
  - Current topics lecture (not on midterm)
- Lecture 8
  - Intro to Prefix Lookup – unibit tries, multibit tries

# Lecture Material – Quick Reference

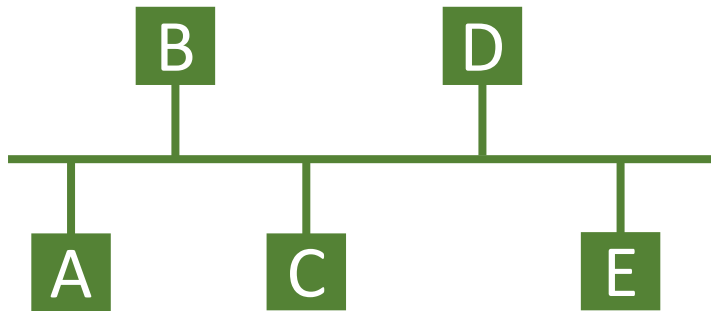
- Lecture 9
  - Prefix lookup: Variable stride tries, Lulea compressed tries
- Lecture 10
  - Non trie approach to prefix lookup: binary search on prefix lengths, binary search on prefix ranges
- Lecture 11
  - Intro to Packet Classification: firewalls, trie of tries (or grid of tries) with backtracking, grid of tries with keeping all possible sources in source trie
- Lecture 12
  - Packet Classification: Grid of tries with switch pointers, geometric view, divide and conquer using bitmaps

# Lecture Material – Quick Reference

- Lecture 13
  - Packet Classification: using bitmaps (cont.), not on midterm: decision trees + neural packet classification
- Lecture 14
  - Vehicular Networking to Intro to cyber security (not on midterm)

# LANs: sending signals across a single wire

- We first learned about local networks

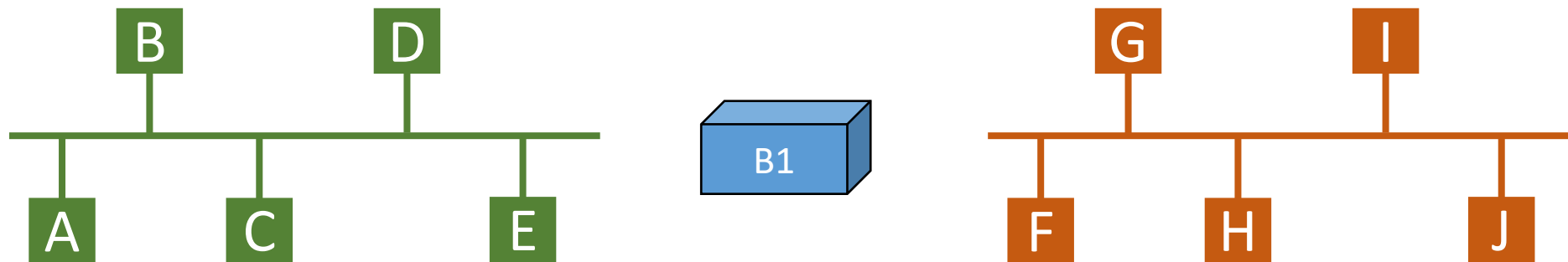


# Ethernet

- Differences between channel partitioning protocols, random access protocols, and controlled access protocols
- Random access protocols: Aloha, Slotted Aloha, CSMA, CSMA/CD
- How/why does collision detection work in CSMA/CD – why do we have a minimum packet size? How is a collision detected?
- Understand ways of dealing with a collision – wait random time, exponential backoff
- Wireless protocols: why doesn't CSMA work in wireless protocols?

# Bridges

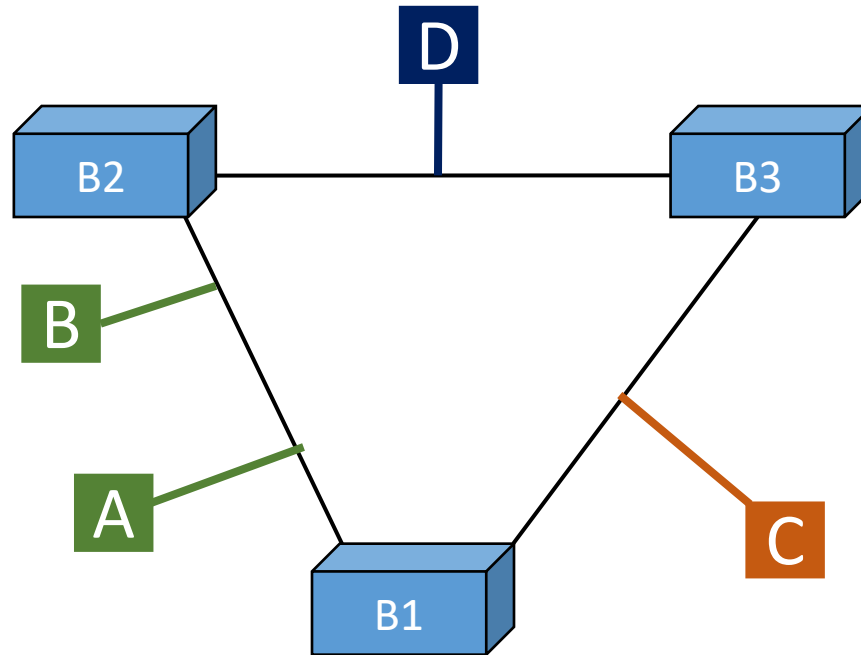
- We then learned about how local networks can be expanded by bridges





# Bridges

- We then learned about how local networks can be expanded by bridges and must deal with cycle (Spanning Tree Protocol)

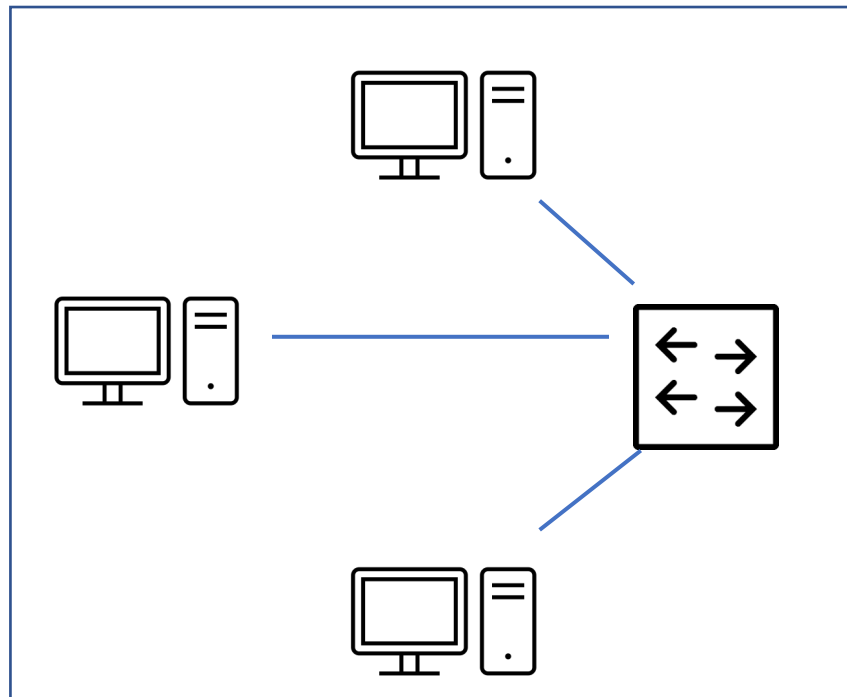


# Bridges and Spanning Tree

- Differences between hubs, bridges, and switches
- Selective forwarding – be comfortable with process of building out forwarding tables
- Spanning Tree Protocol:
  - Messages sent between bridges during protocol
  - Which ports will shut off at the end
  - Understand implications of ports being shut off (i.e., this is essentially breaking the connection between the LAN and bridge at the port)

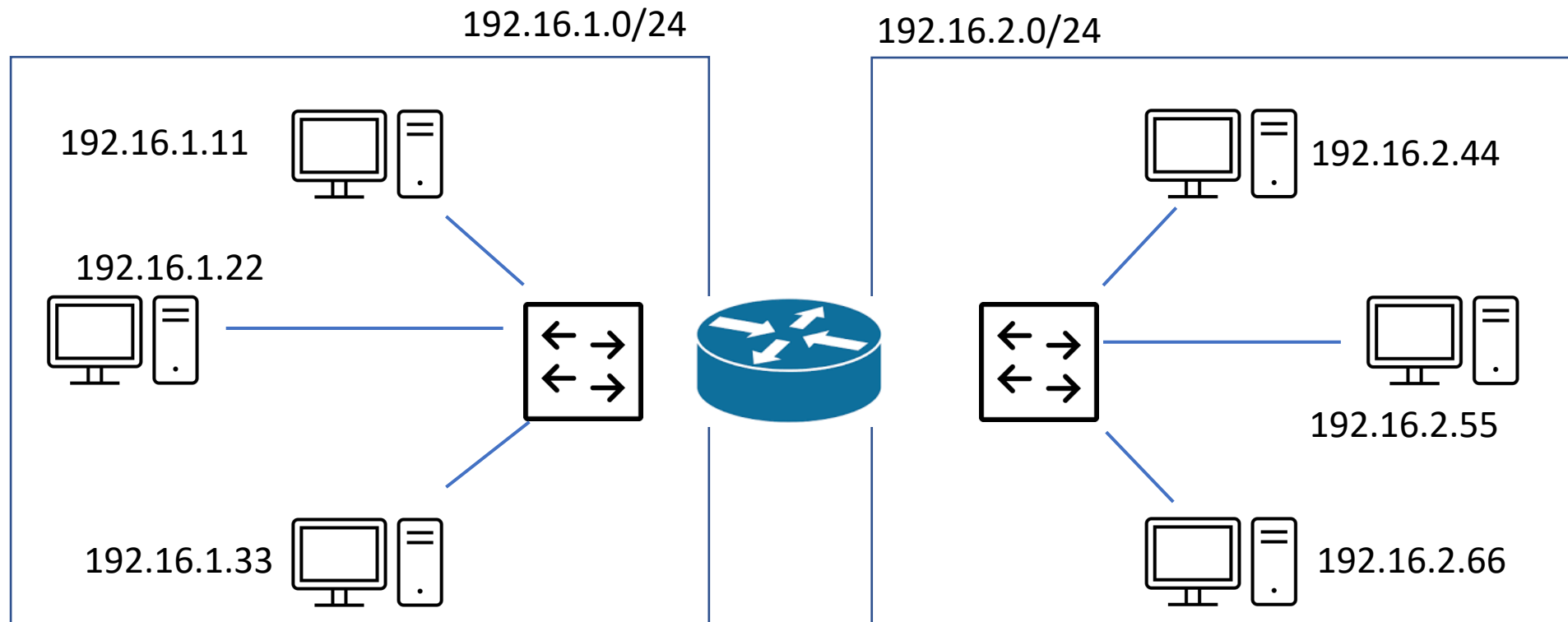
# Switching

- Switches are now the common device to forward traffic within a LAN

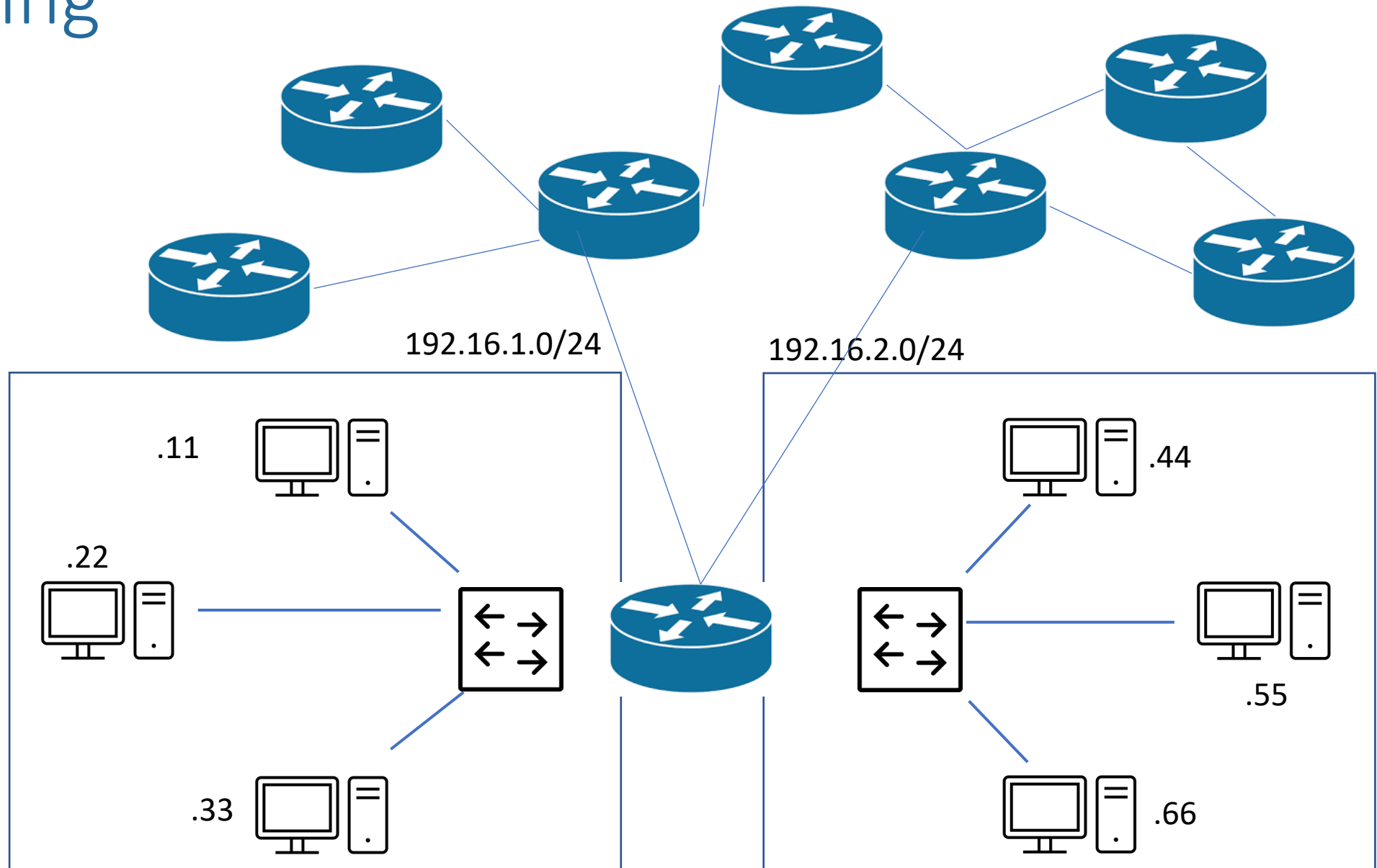


# Routing

- To connect different local networks, we need **routers**, which forward based on **IP addresses**



# Routing



# Routing Protocols

- IP addresses – hierarchical nature, how networks can be characterized by an IP prefix
- Given a network, know network/broadcast/host addresses
- Private vs. public IP addresses
- Address Resolution Protocol (ARP)
- Difference between routing and switching

# Routing Protocols Cont.

- How the router populates the FIB:
  - Distance Vector:
    - How each router updates its tables in each step (Bellman-Ford)
    - Link changes: count to infinity problem, poison reverse
    - BGP hijacking: be able to explain at a high-level what it is
  - Link state:
    - How each router calculates its shortest paths, given the full topology (Dijkstra)
    - Link changes: temporary loops are possible

# Looking Up the FIB

- Routers look up the FIB by finding the **longest matching prefix**

Destination	Next Hop
192.168.74.0/24	Router 1
192.168.74.192/28	Router 2
192.168.74.204/30	Router 3
10.1.120.0/21	Router 4
0.0.0.0/0	Router 5



# Longest Matching Prefix

- Decide which interface a packet should be sent to, given a forwarding information base
- Unibit trie, multibit trie, Lulea compressed trie: what they are, how to form them from a forwarding information base, and pros/cons of each
- Binary search on prefix lengths: how it works (array of tables), why we have markers, why best matching prefix is stored with each marker, i.e., to avoid backtracking
- Binary search on prefix ranges: given a list of prefixes, be able to show ranges and form tree of range endpoints with  $>$  and  $=$  values for each node

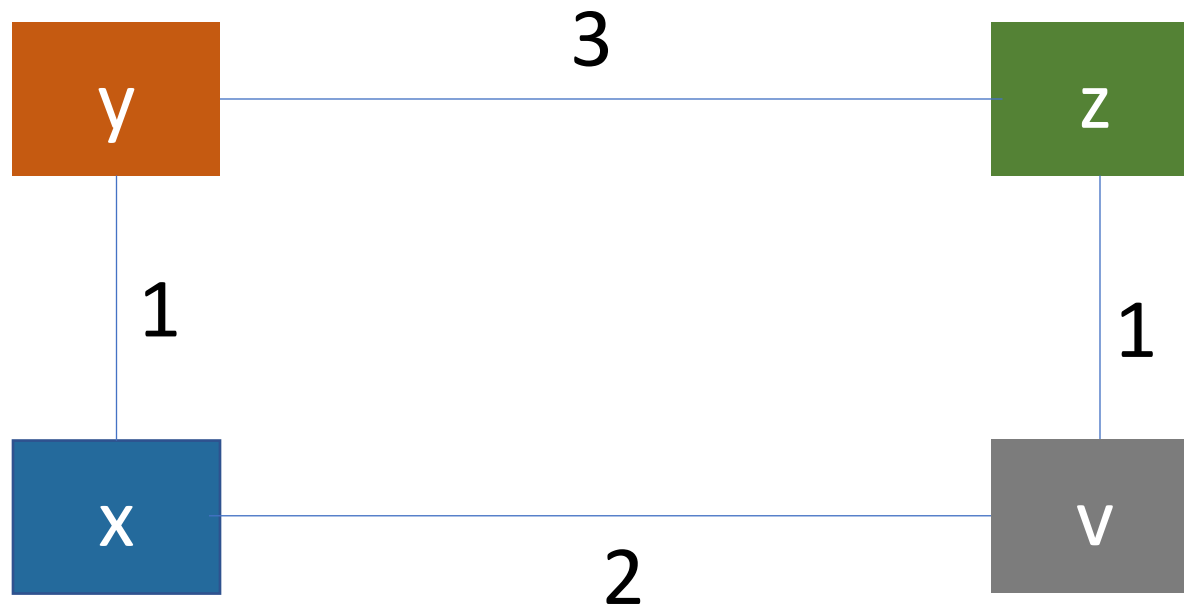
# Packet Classification

- Converting firewall rules (in plain English) to a rule base (similar to hw)
- Two dimensional schemes: grid of tries -> understand the three methods from class:
  - Form the grid of tries adding each source prefix exactly once, and use a backtracking algorithm to find the best matching rule
  - Form the grid of tries, adding each source prefix to every source trie corresponding to a matching destination prefix and do not backtrack
  - Form the grid of tries, adding each source prefix exactly once, and use switch pointers to move between the different source tries
- Beyond two dimensions: using bitmaps to find the least cost rule

# Round 1

Dst	Cost	Next hop
x	1	x
y	0	-
z	3	z
v	$\infty$	-

Dst	Cost	Next hop
x	0	-
y	1	y
z	$\infty$	-
v	2	v

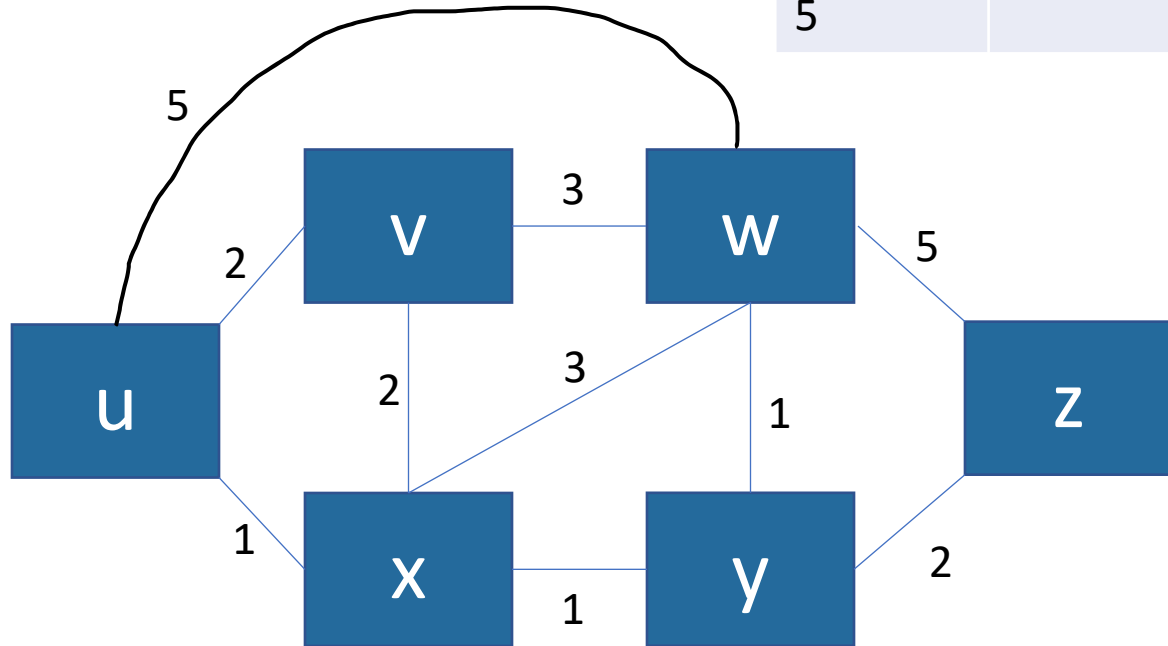


Dst	Cost	Next hop
x	$\infty$	-
y	3	y
z	0	-
v	1	v

Dst	Cost	Next hop
x	2	x
y	$\infty$	-
z	1	z
v	0	-

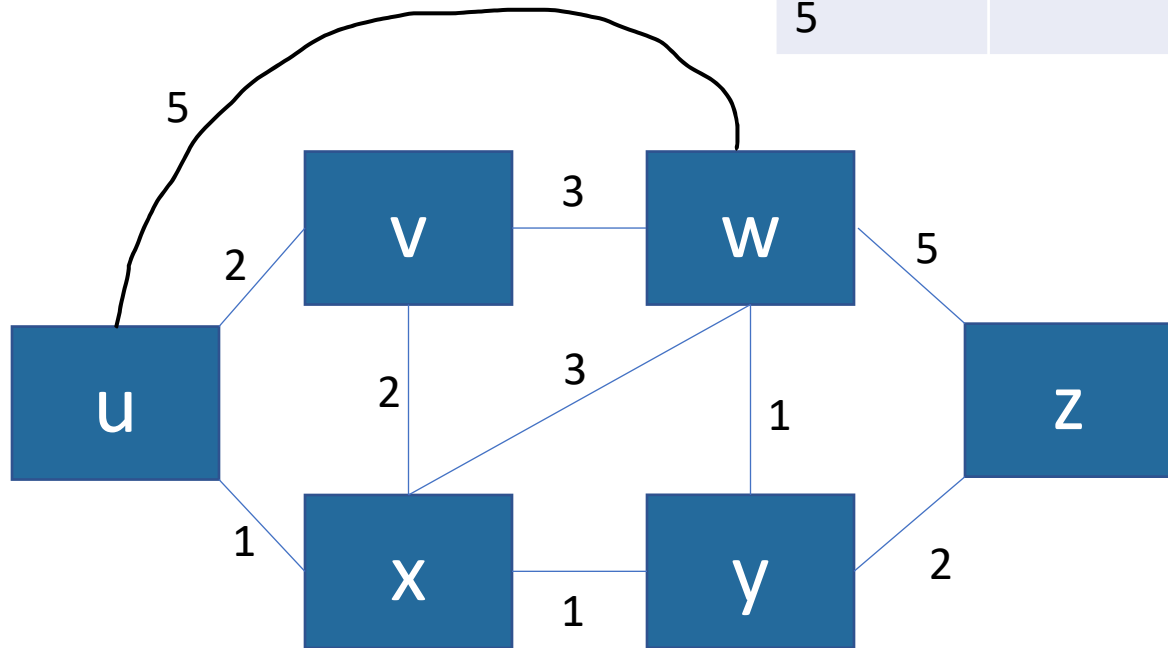
# Dijkstra's: Example

Step	N	Q	v	w	x	y	z
0	{u}	{vwxyz}	2, u	5, u	1, u	$\infty$	$\infty$
1							
2							
3							
4							
5							



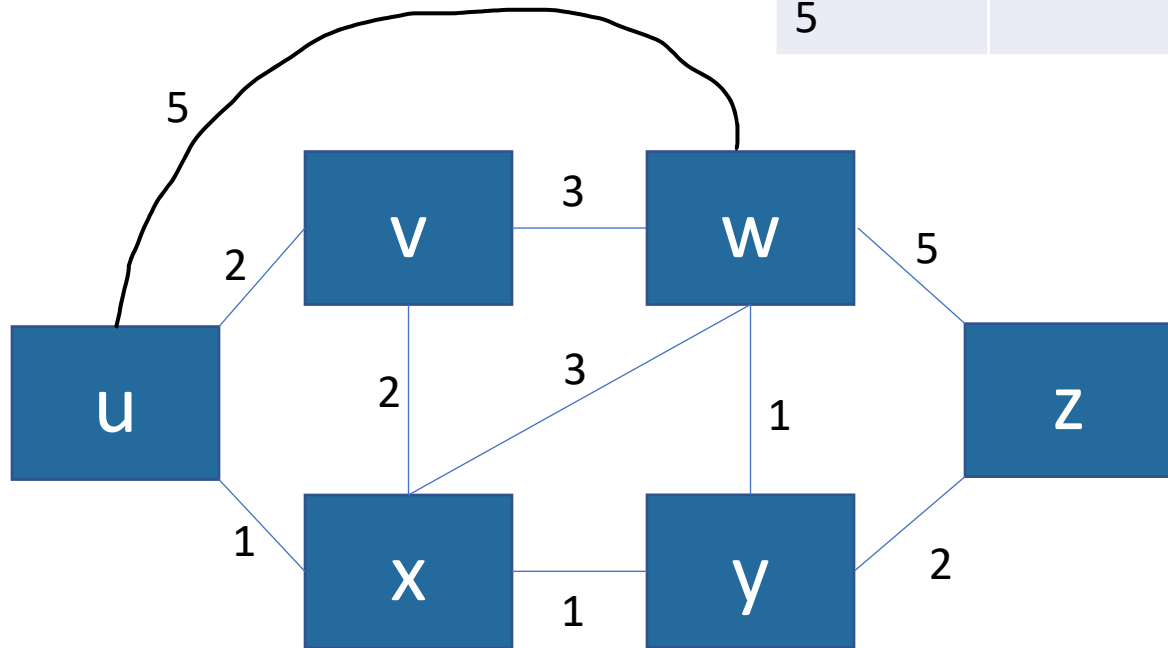
# Dijkstra's: Example

Step	N	Q	v	w	x	y	z
0	{u}	{vwxyz}	2, u	5, u	1, u	$\infty$	$\infty$
1	{ux}	{vwyz}	2, u	4, x		2, x	$\infty$
2							
3							
4							
5							



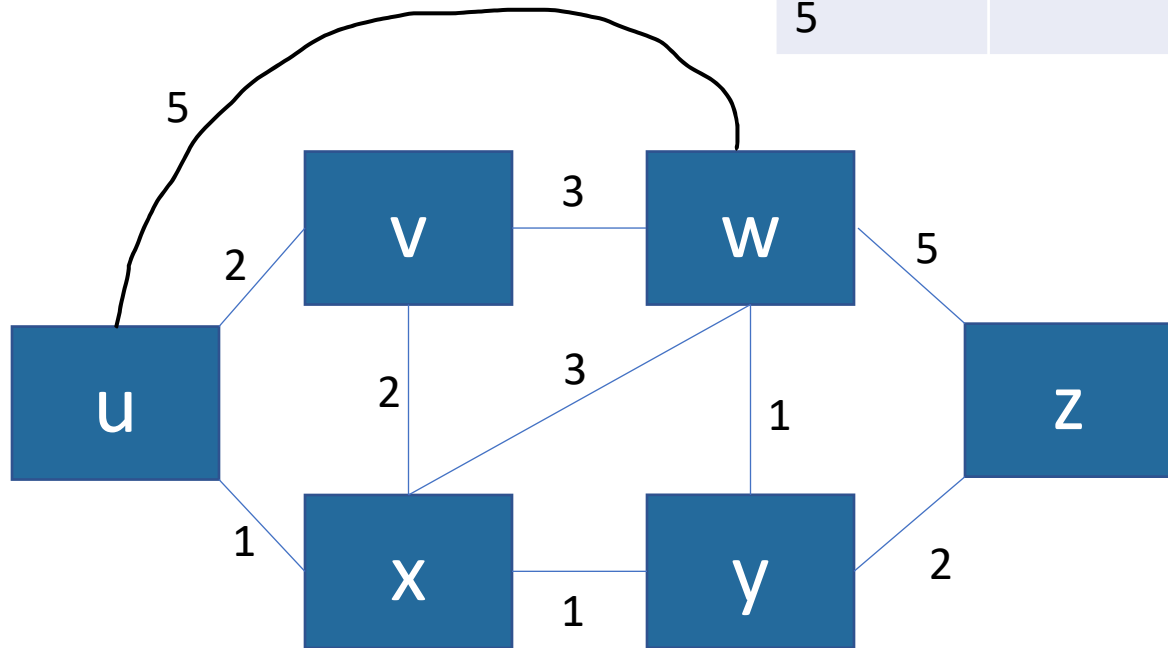
# Dijkstra's: Example

Step	N	Q	v	w	x	y	z
0	{u}	{vwxyz}	2, u	5, u	1, u	$\infty$	$\infty$
1	{ux}	{vwyz}	2, u	4, x		2, x	$\infty$
2	{uxy}	{vwz}	2, u	3, y			4, y
3							
4							
5							



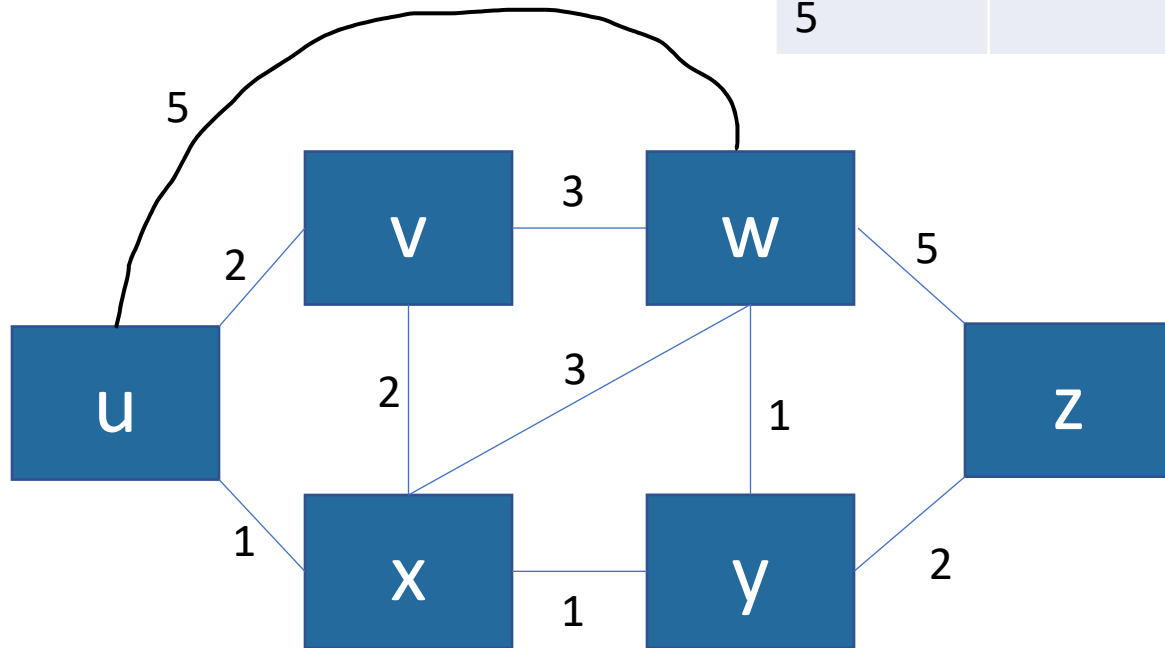
# Dijkstra's: Example

Step	N	Q	v	w	x	y	z
0	{u}	{vwxyz}	2, u	5, u	1, u	$\infty$	$\infty$
1	{ux}	{vwyz}	2, u	4, x		2, x	$\infty$
2	{uxy}	{vwz}	2, u	3, y			4, y
3	{uxyv}	{wz}		3, y			4, y
4							
5							



# Dijkstra's: Example

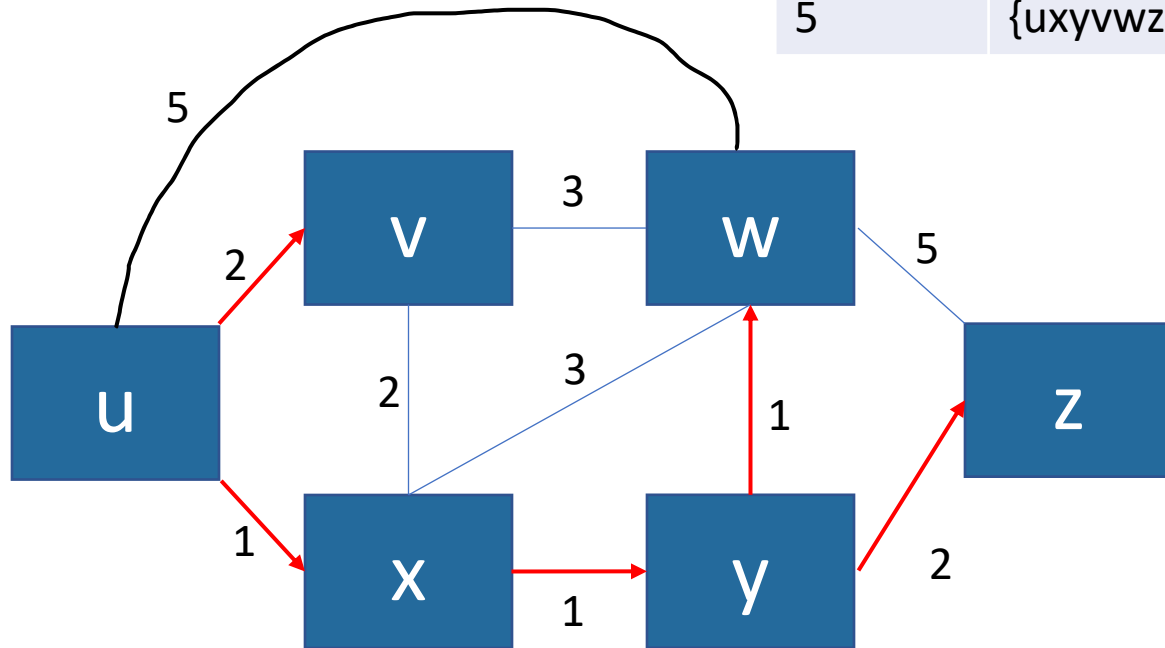
Step	N	Q	v	w	x	y	z
0	{u}	{vwxyz}	2, u	5, u	1, u	$\infty$	$\infty$
1	{ux}	{vwyz}	2, u	4, x		2, x	$\infty$
2	{uxy}	{vwz}	2, u	3, y			4, y
3	{uxyv}	{wz}		3, y			4, y
4	{uxyvw}	{z}					4, y
5							





# Dijkstra's: Example

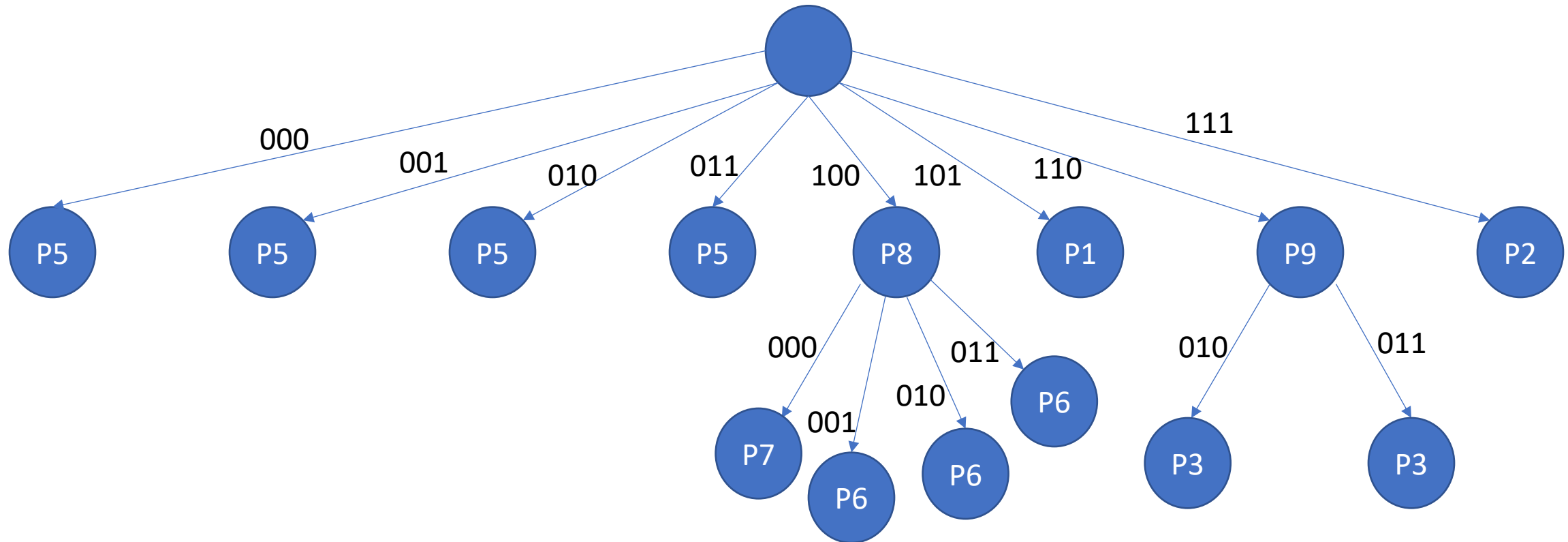
Step	N	Q	v	w	x	y	z
0	{u}	{vwxyz}	2, u	5, u	1, u	$\infty$	$\infty$
1	{ux}	{vwyz}	2, u	4, x		2, x	$\infty$
2	{uxy}	{vwz}	2, u	3, y			4, y
3	{uxyv}	{wz}		3, y			4, y
4	{uxyvw}	{z}					4, y
5	{uxyvwz}	{}					



# Prefix Expansion

Interface	Prefix	
P1	101*	→ 101*
P2	111*	→ 111*
P3	11001*	→ 110010*, 110011*
P4	1*	→ <del>100*</del> , <del>101*</del> , <del>110*</del> , <del>111*</del>
P5	0*	→ 000*, 001*, 010*, 011*
P6	1000*	→ <del>10000*</del> , 100001*, 100010*, 100011*
P7	100000*	→ 100000*
P8	100*	→ 100*
P9	110*	→ 110*

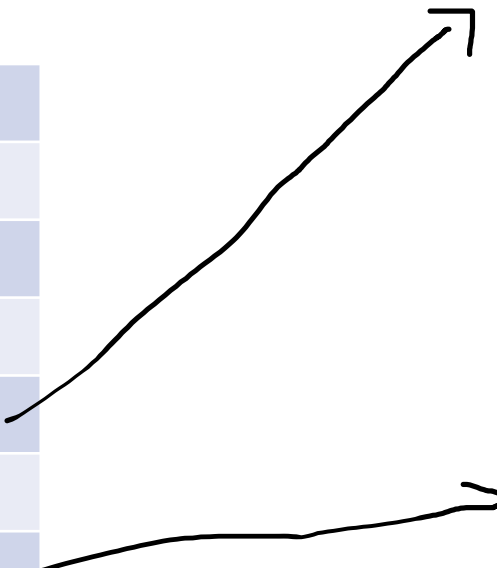
# Trie formed from expanded prefixes



# Multibit Tries in Memory

- Each node contains a list of pairings:  
<Prefix, Pointer>

000	P5	null
001	P5	null
010	P5	null
011	P5	null
100	P8	
101	P1	null
110	P9	
111	P2	null



000	P7	null
001	P6	null
010	P6	null
011	P6	null
100		null
101		null
110		null
111		null

000		null
001		null
010	P3	null
011	P3	null
100		null
101		null
110		null
111		null

# One More Example

- Draw all switch pointers for correctness, but remember: it is very important that when looking up the source trie, we keep track of the best rule we've seen so far
  - i.e., it is possible for switch pointers to point us to a worse rule, but we need to remember the best we've seen so far

Rule	Destination	Source
R1	D1 = 101*	S1 = 0*
R2	D2 = 10*	S2 = 1*
R3	D3 = 0*	S3 = 10*
R4	D4 = 0*	S4 = 00*
R5	D5 = *	S5 = 00*
R6	D6 = *	S6 = 01*
R7	D7 = *	S7 = *