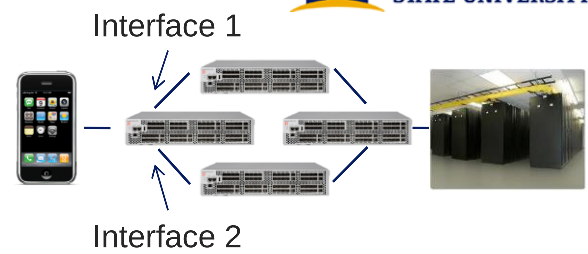
Datagram networks

- Datagram network provides network-layer *connectionless* service
  - No call setup at network layer
  - Routers: no state about end-to-end connections
- Packets forwarded using destination host address
  - How to keep routing tables small?
  - Longest prefix matching



Address range	Interface
128.11.52.0 – 128.11.52.255	1
153.90.2.0 – 153.90.2.255	2
153.90.2.87 – 153.90.2.89	1

Longest Prefix Matching

Routing Table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Match + Action

Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

Longest Prefix Matching

Routing Table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Match + Action

Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

Translate the routing table from binary to CIDR representation

```
200.23.16.0 to 200.23.23.255   → 0
200.23.24.0 to 200.23.24.255   → 1
200.23.25.0 to 200.23.31.255   → 2
else                           → 3
```

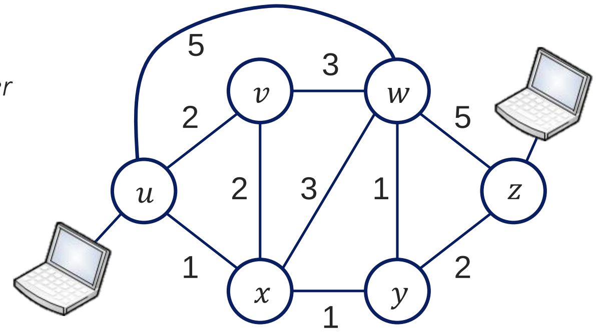
```
200.23.16.0/21                 → 0
200.23.24.0/24                 → 1
200.23.24.0/21                 → 2
else                           → 3
```


Chapter 5: Network Layer – Control Plane

Introduction to Routing

MONTANA STATE UNIVERSITY

- Routing:
  - Discovery of paths between *first-hop router* and the *destination router*
- Path *routing metric*
  - Shortest path
    - Number of links
    - Sum of link delay
  - Lowest cost path
    - Sum of link costs
  - Highest throughput path
    - Maximum link capacity
  - Highest path reliability
    - Prob. of contact between path nodes in an ad-hoc mobile network



- Network model
  - Graph $G = (V, E)$
    - $V = \{u, v, w, x, y, z\}$
    - $E = \{(u, v), (u, x), (v, x), \dots\}$
  - Link cost $c(x, y)$
    - $c(x, y) = c(y, x)$
    - If $(x, y) \notin E$ then $c(x, y) = \infty$

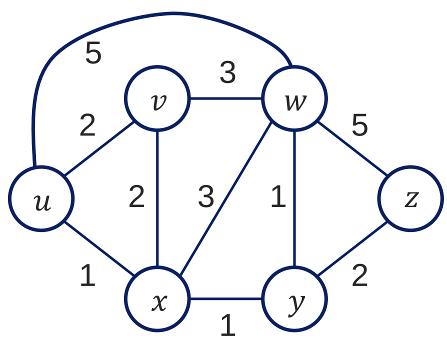
Introduction to Routing

MONTANA STATE UNIVERSITY

What is the least cost path between u and z?

- Come up with a routing algorithm based on global knowledge of link costs
  - What happens wh… ery large?
  - What is the effect … uent changes in network topology?

Link State (LS)



- Network model
  - Graph $G = (V, E)$
    - $V = \{u, v, w, x, y, z\}$
    - $E = \{(u, v), (u, x), (v, x), \dots\}$
  - Link cost $c(x, y)$
    - $c(x, y) = c(y, x)$
    - If $(x, y) \notin E$ then $c(x, y) = \infty$

Dijkstra’s Algorithm

MONTANA STATE UNIVERSITY

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

 Loop
   find w not in N' such that D(w) is a minimum
   add w to N'
   update D(v) for all v adjacent to w and not in N' :
      D(v) = min( D(v), D(w) + c(w,v) )
   /* new cost to v is either old cost to v or known
    shortest path cost to w plus cost from w to v */
 until all nodes in N'
```

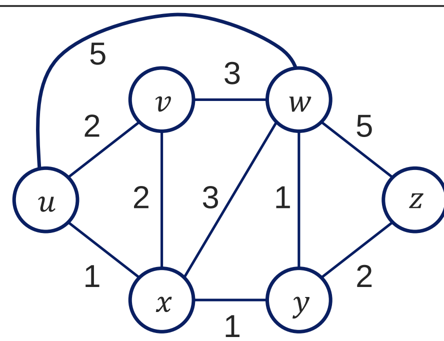
Update shortest path if transitive cost through **added node** is lower

Dijkstra’s algorithm example

MONTANA STATE UNIVERSITY

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

So Dijkstra gives you the **cost** of the shortest path to each node. How would you find the **actual path** of nodes?



Introduction to Routing

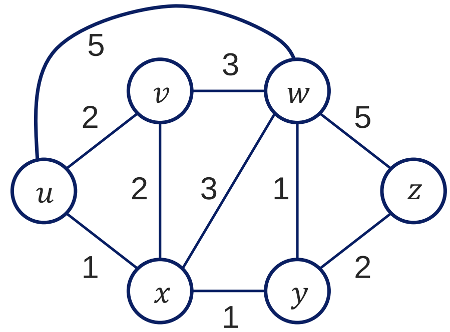
What is the least cost path between u and z?

- Come up with a routing algorithm based on global knowledge of link costs
  - What happens when the network ... rge?
  - What is the effect of ... network topology?

Link State (LS)

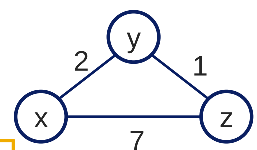
- Come up with a routing algorithm based on local knowledge of link costs?
  - Node x knows $c(x,v)$ for all its one hop ... s
  - Node x can advertise its ... $d_x(y)$
  - Belman-Ford equation:
    $d_x(y) = \min_{v \in V}[c(x,v) + d_v(y)]$
  - What is the effect of frequent **cost** changes in network topology?

Distance Vector (DV)



- Network model
  - Graph $G = (V, E)$
    - $V = \{u, v, w, x, y, z\}$
    - $E = \{(u,v), (u,x), (v,x), \dots\}$
  - Link cost $c(x, y)$
    - $c(x,y) = c(y,x)$
    - If $(x,y) \notin E$ then $c(x,y) = \infty$

Distance Vector (DV) Algorithm



Node	Cost to		
x	x	y	z
From x	0	2	7
From y	∞	∞	∞
From z	∞	∞	∞

```
Initialization:
   for each destination y in N:
      D_w(y) = ∞
   for all neighbors y in N:
      D_x(y) = c(x,y)
   for all destination y in N:
      send distance vector D_x = [D_x(y): y in N] to neighbors

 while True:
   wait for new distance vector from v
   for each y in N:
      D_x(y) = min_v{c(x,v) + D_v(y)}
      if D_x(y) change for any destination y:
         send D_x = [D_x(y): y in N] to all neighbors
```

Forwarding table num. rows equal to num. neighbors

Update shortest path if transitive cost through **announcing node** is lower

DV Example

```
D_x(z) = min_v{c(x,v) + D_v(y)}
       = min{c(x,x) + D_x(z),
             c(x,y) + D_y(z)}
       = min{0+7, 2+1} = 3
```

Node		Cost to		
x		x	y	z
From	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

Node		Cost to		
y		x	y	z
From	x	∞	∞	∞
	y	2	0	1
	z	∞	∞	∞

Node		Cost to		
z		x	y	z
From	x	∞	∞	∞
	y	∞	∞	∞
	z	7	1	0

Node		Cost to		
x		x	y	z
From	x	0	2	3
	y	2	0	1
	z	7	1	0

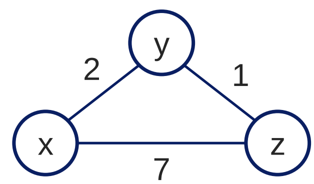
Node		Cost to		
y		x	y	z
From	x	0	2	7
	y	2	0	1
	z	7	1	0

Node		Cost to		
z		x	y	z
From	x	0	2	7
	y	2	0	1
	z	3	1	0

Node		Cost to		
x		x	y	z
From	x	0	2	3
	y	2	0	1
	z	3	1	0

Node		Cost to		
y		x	y	z
From	x	0	2	3
	y	2	0	1
	z	3	1	0

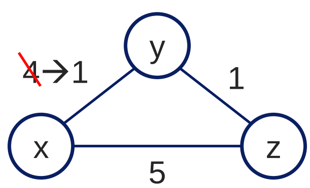
Node		Cost to		
z		x	y	z
From	x	0	2	3
	y	2	0	1
	z	3	1	0



DV Examples

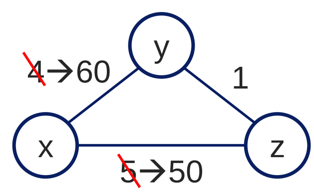
MONTANA STATE UNIVERSITY

Link cost decrease



- At time instant t_1 node y notices cost change and updates its D vector
- At t_2 z receives y's D vector and updates its $D_z(x)$ to 2
- At t_3 y receives z's D vector, makes no change to own D and the algorithm come to a *quiescent* state

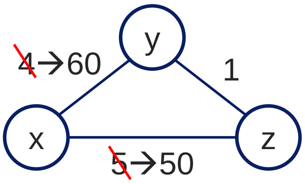
Link cost increase



- At t_1 y notices cost change and

DV Poisoned Reverse

Link cost increase



- At t_1 y notices cost change and updates its $D_y(x)$ to 6
- At t_2 z receives y's D vector and updates its $D_z(x)$ to 7
  - Routing loop!
- At t_3 y receives z's D vector and updates its $D_y(x)$ to 8
  - *Count to infinity* problem!

- If z routes through y to x,
  z advertises *poisoned* $D_z(x) = \infty$ to y
- At t_1 y notices cost change and updates its $D_y(x)$ to **60**
- At t_2 z receives y's D vector and **keeps** its $D_z(x)$ at 50
- At t_3 y receives z's D vector (no longer *poisoned*) and updates its $D_y(x)$ to 51

- Count to infinity problem not solved for routing loops with 3+ nodes…

Link State vs. Distance Vector

- Mechanism
  - LS: centralized, synchronous
  - DV: distributed, asynchronous
- Message complexity
  - LS: $O(VE)$ messages sent
  - DV: exchange between neighbors only
- Speed of convergence
  - LS: $O(n \log n)$ algorithm
  - DV: convergence time varies
    - May get routing loops
    - Count-to-infinity problem

- Robustness: what happens if router malfunctions?
  - LS:
    - Node can advertise incorrect link cost
    - Each node computes only its own table
  - DV:
    - DV node can advertise incorrect path cost
    - Each node's table used by others
      - Errors propagate through the network