CSC4200/5200 – COMPUTER NETWORKING

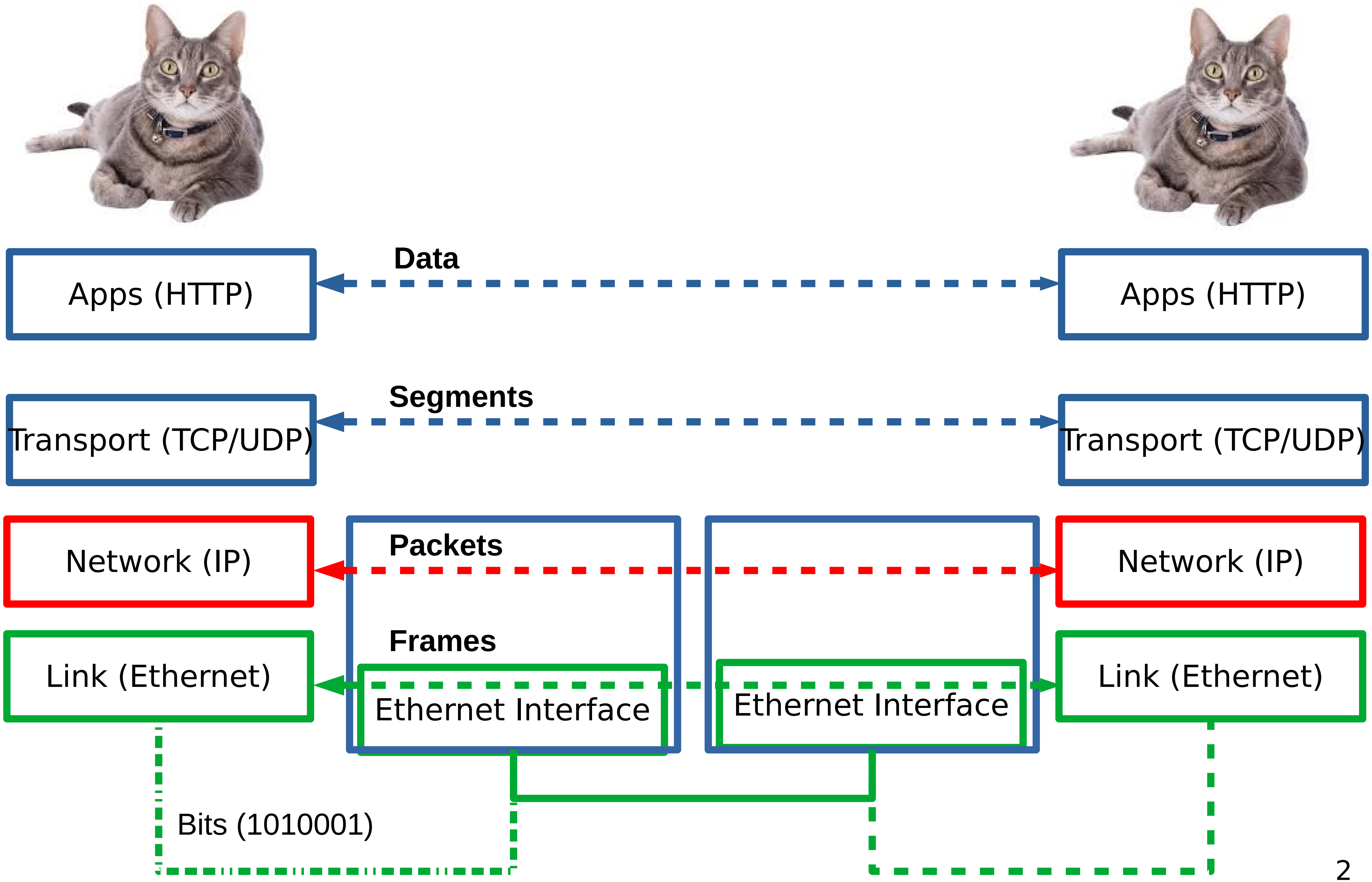
Instructor: Susmit Shannigrahi

ROUTING - CONTINUED

sshannigrahi@tntech.edu

GTA: dereddick42@students.tntech.edu

Tennessee TECH


Data
Apps (HTTP)
Apps (HTTP)
Segments
Transport (TCP/UDP)
Transport (TCP/UDP)
Packets
Network (IP)
Network (IP)
Frames
Link (Ethernet)
Ethernet Interface
Ethernet Interface
Link (Ethernet)
Bits (1010001)

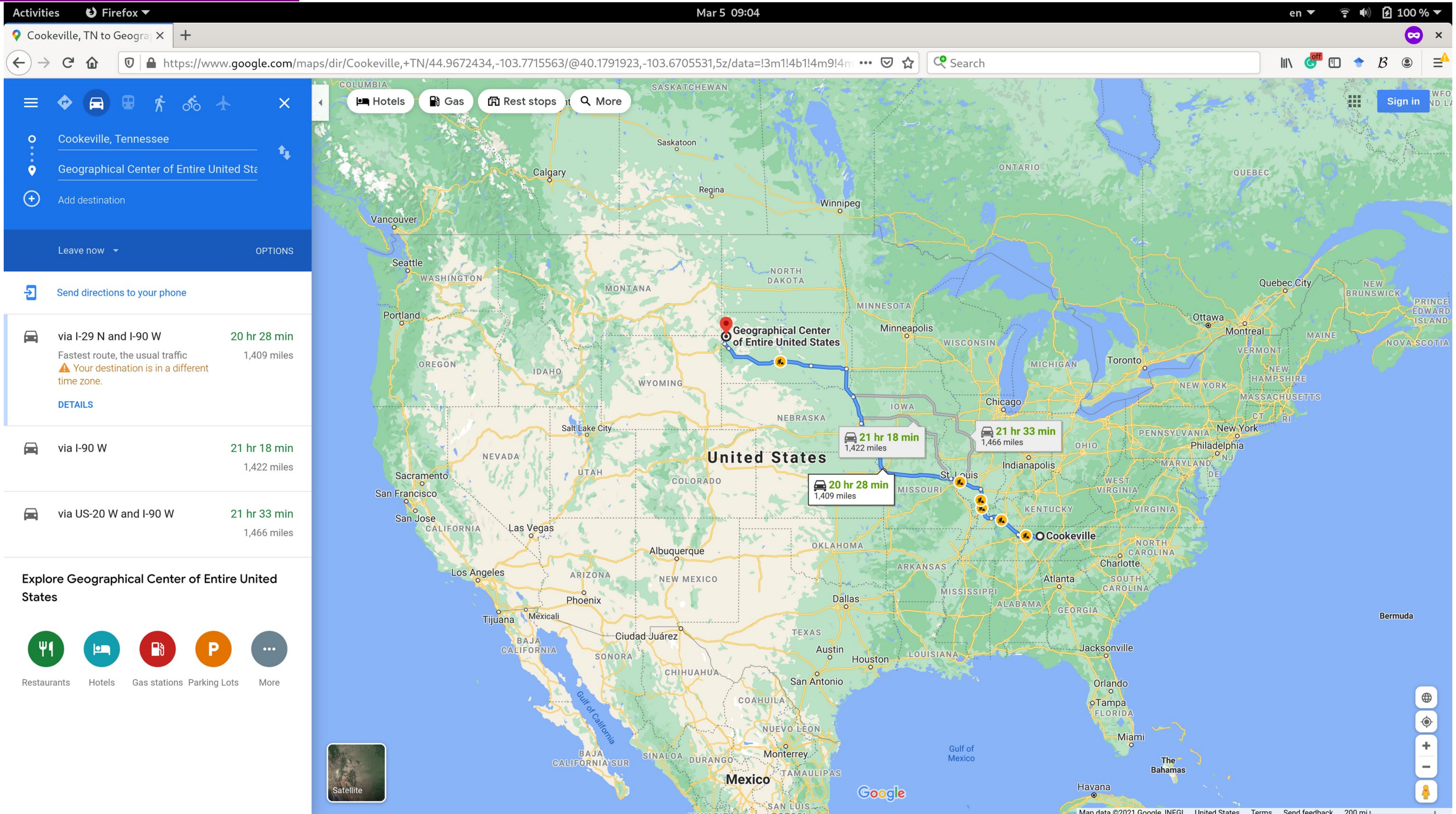
Forwarding vs Routing

- Forwarding:
  - to select an output port based on destination address and routing table
  - **Local path**

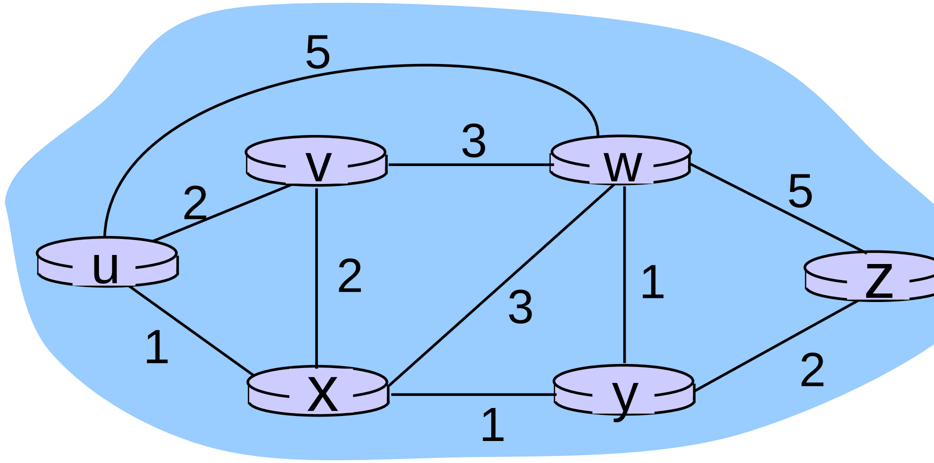
SubnetNumber	SubnetMask	NextHop
128.96.34.0	255.255.255.128	Interface 0
128.96.34.128	255.255.255.128	Interface 1
128.96.33.0	255.255.255.0	R2

- Routing:
  - process by which routing table is built
  - **End-to-end path**

Routing = Navigation



Why bother?



- Quality of path affects performance
  - Longer path = more delay
- Balance path usage, avoid congested paths
- Deal with failures

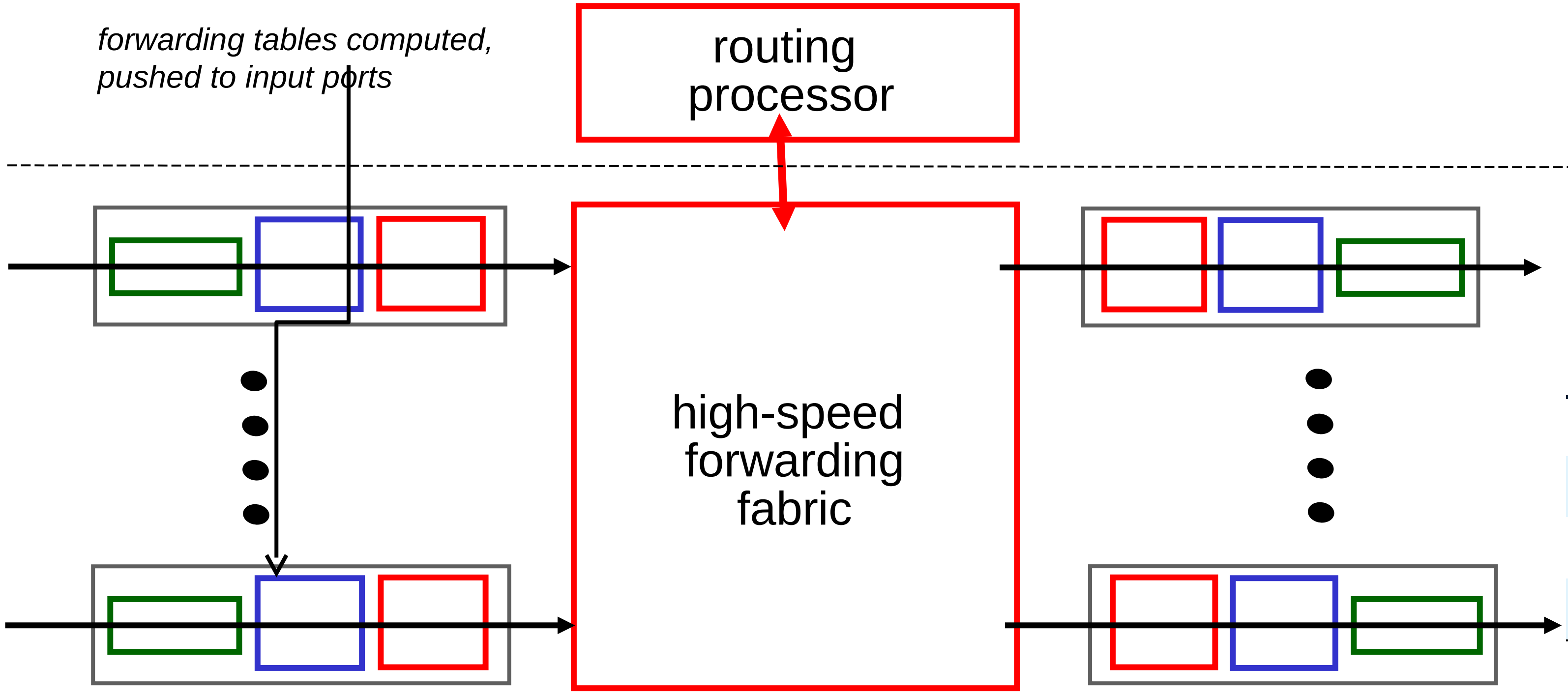
SubnetNumber	SubnetMask	NextHop
128.96.34.0	255.255.255.128	Interface 0
128.96.34.128	255.255.255.128	Interface 1
128.96.33.0	255.255.255.0	R2

Router architecture overview

Two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link

Control Plane = routing
Vs
Data Plane = forwarding



routing, management control plane (software)

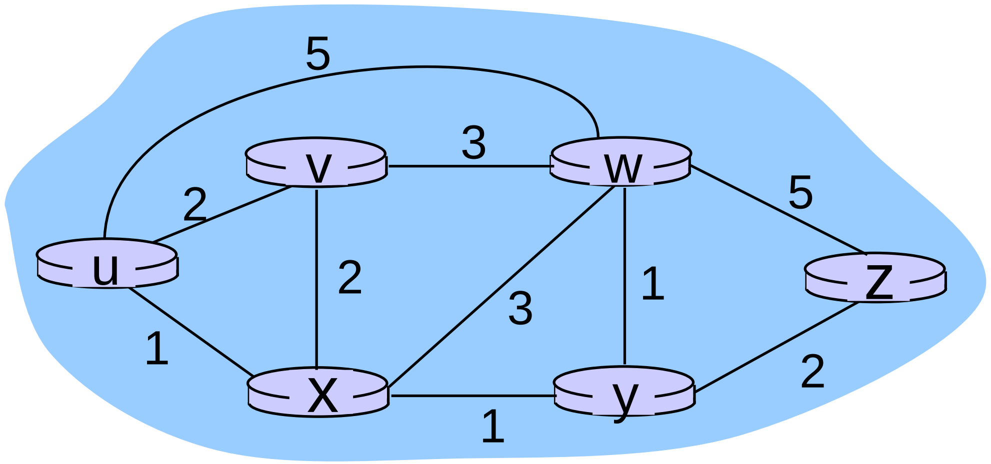
forwarding data plane (hardware)

SubnetNumber	SubnetMask	NextHop
128.96.34.0	255.255.255.128	Interface 0
128.96.34.128	255.255.255.128	Interface 1
128.96.33.0	255.255.255.0	R2

router input ports

router output ports

Graph abstraction



graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

X→ Z

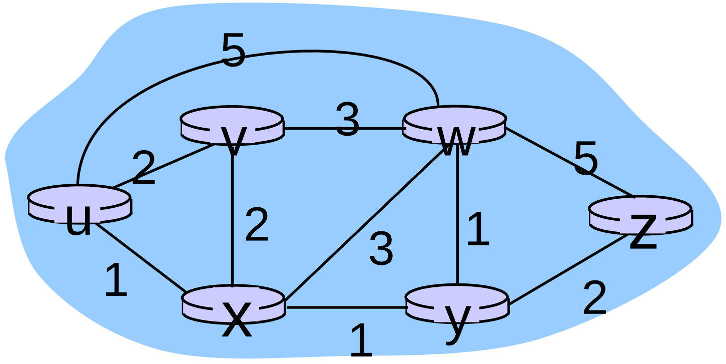
Cost (x,v,w,z) = cost(x,v) + cost (v, w) + cost(w,z) = 10

Cost (x,w,z) = cost(x,w) + cost(w,z) = 8

Cost(x, y, z) = ?

Objective → find the lowest cost path between all nodes

Dijkstra's Shortest-Path Algorithm



- Given a graph (network) with link costs
- Find the lowest cost paths to all nodes

- Iterative
  - After n iterations, you will find least cost path to *n nodes*

- *S = Least cost paths already known, initially source node {U}*
- *D(v): current cost of path from source(U) to node V*
  - *Initially, D(v) = c(u,v) for all nodes v adjacent to u*
  - *D(v) = ∞ for all other nodes*
  - *Update D(v) as we go*

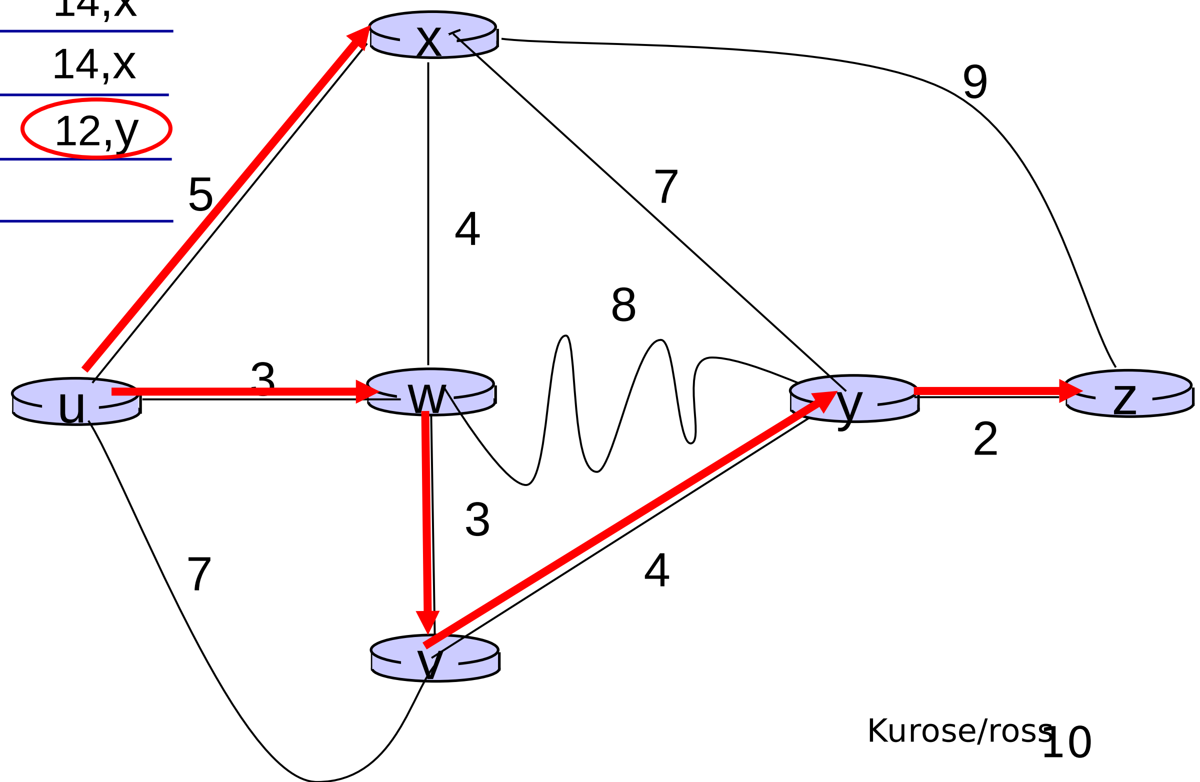
Dijsktra’s Algorithm

```
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'
```

Dijkstra's algorithm: example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

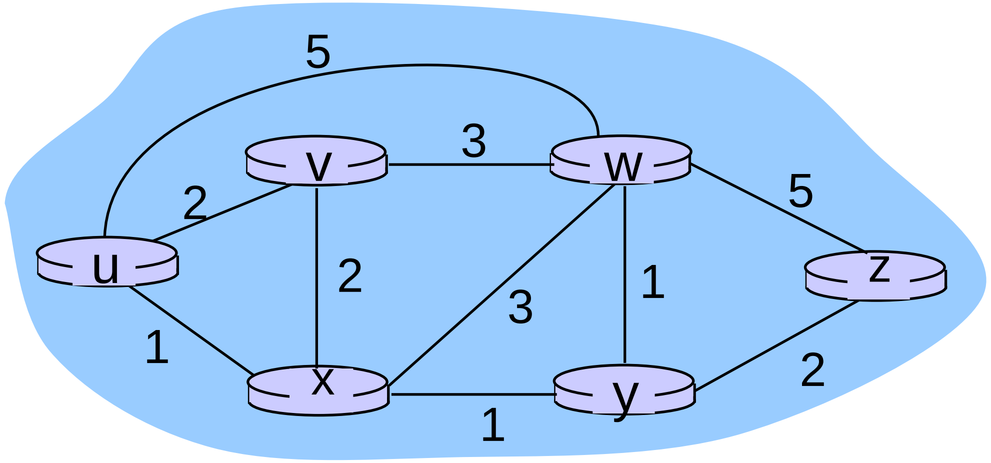


notes:

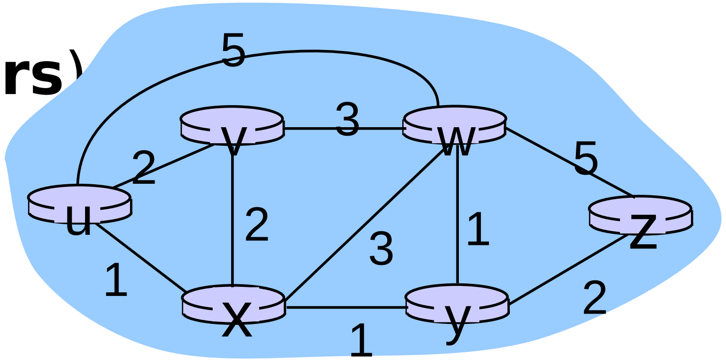
- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

Dijkstra's algorithm: another example

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					



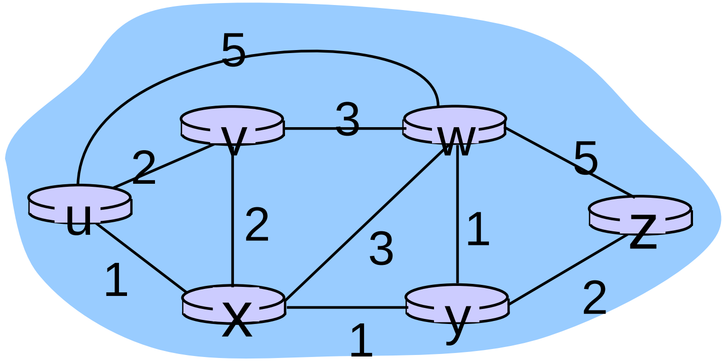
Dijsktra’s → Link State Routing



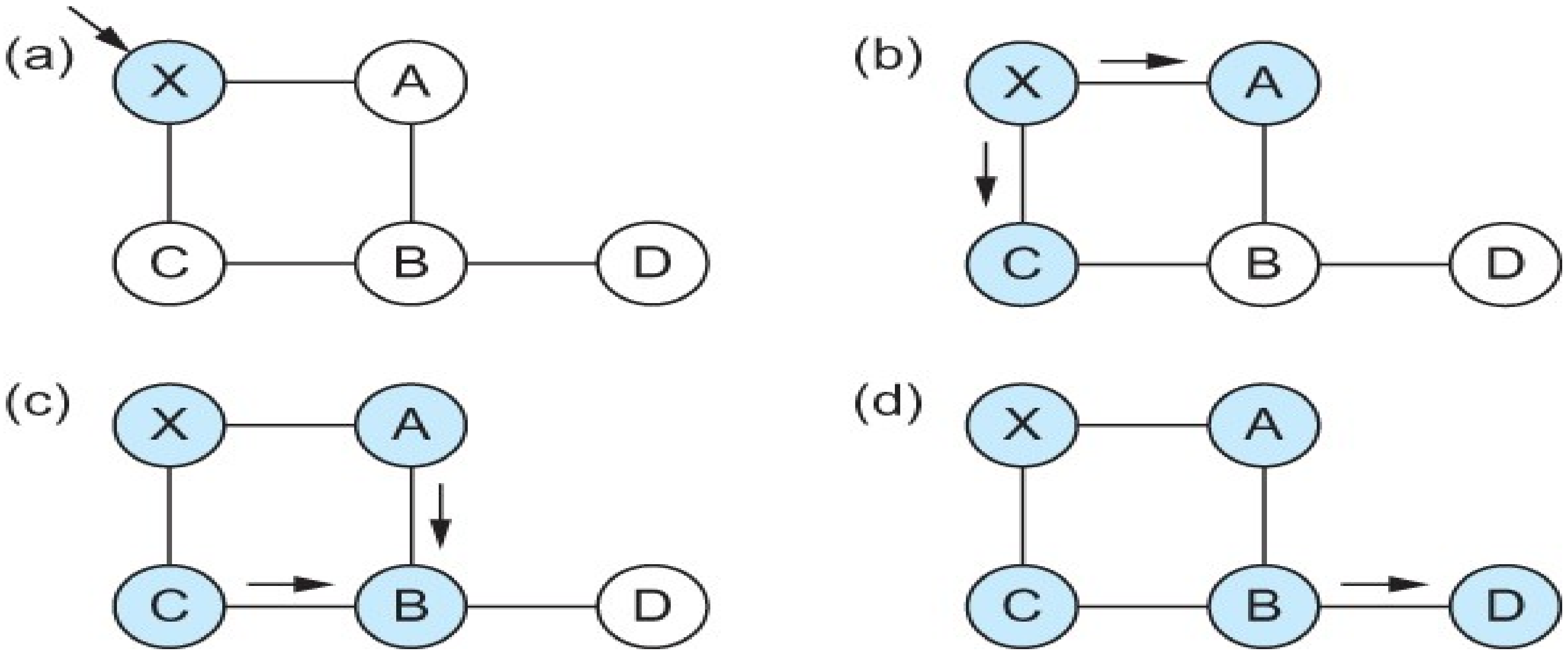
- Each node keeps track of adjacent links (**neighbors**)
- Each router broadcasts it’s state (**network map**)
- Each router runs Dijkstra’s algorithm
  (finds the shortest path)
- Each router has complete picture of the network
- Example: Open Shortest Path First (OSPF)

OSPF – Open Shortest Path First

- SPF – another name for Link State Routing
- Each node creates an update packet - link-state packet (LSP)
  - The ID of the node that created the LSP (U)
  - A list of directly connected neighbors and the cost of the link ((V, 2), (X, 1), **(W, 5)**)
- A sequence number (1122)
- A time to live for this packet (16)
- LSP → ({U}, {(V, 2), (X, 1), (W, 5)}, {1122}, {16})



OSPF – controlled flooding



Flooding of link-state packets. (a) LSP arrives at node X; (b) X floods LSP to A and C; (c) A and C flood LSP to B (but not X); (d) flooding is complete

Link State Routing – controlled flooding

- Flood when topology changes or link goes down
  - Detected by periodic hello messages
  - If message missed → link down

- Refresh and flood periodically

- Problems?
  - High computational cost
  - Reliable flooding may not be reliable

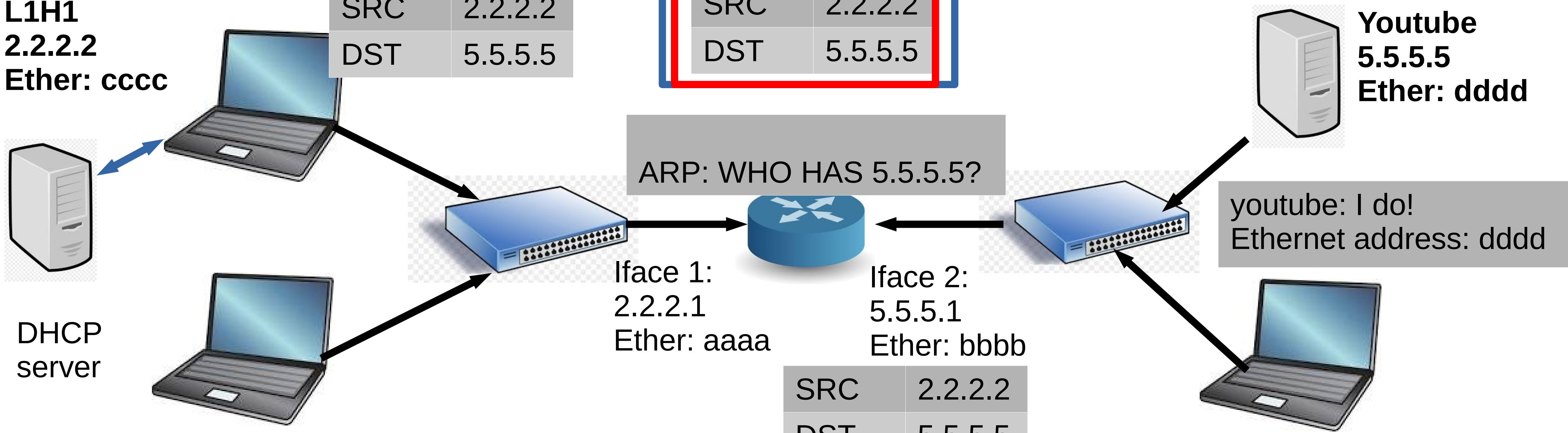
Tying it all together in the network layer

SRC	2.2.2.2
DST	5.5.5.5

Decapsulate IP packet

SRC	bbbb
DST	dddd

SRC	2.2.2.2
DST	5.5.5.5



SRC	2.2.2.2
DST	5.5.5.5

SRC	2.2.2.2
DST	5.5.5.5

Routing Table

We are populating this!!!

5.5.5.0/8	IF: 2
2.2.2.0/8	IF: 1

Reading Assignment

- Network as a graph:
  - https://book.systemsapproach.org/internetworking/routing.html#network-as-a-graph
  - Approximately 5 minutes

- Link state:
  - https://book.systemsapproach.org/internetworking/routing.html#link-state-ospf
  - Approximately 20 minutes