

# **CSC4200/5200 – COMPUTER NETWORKING**

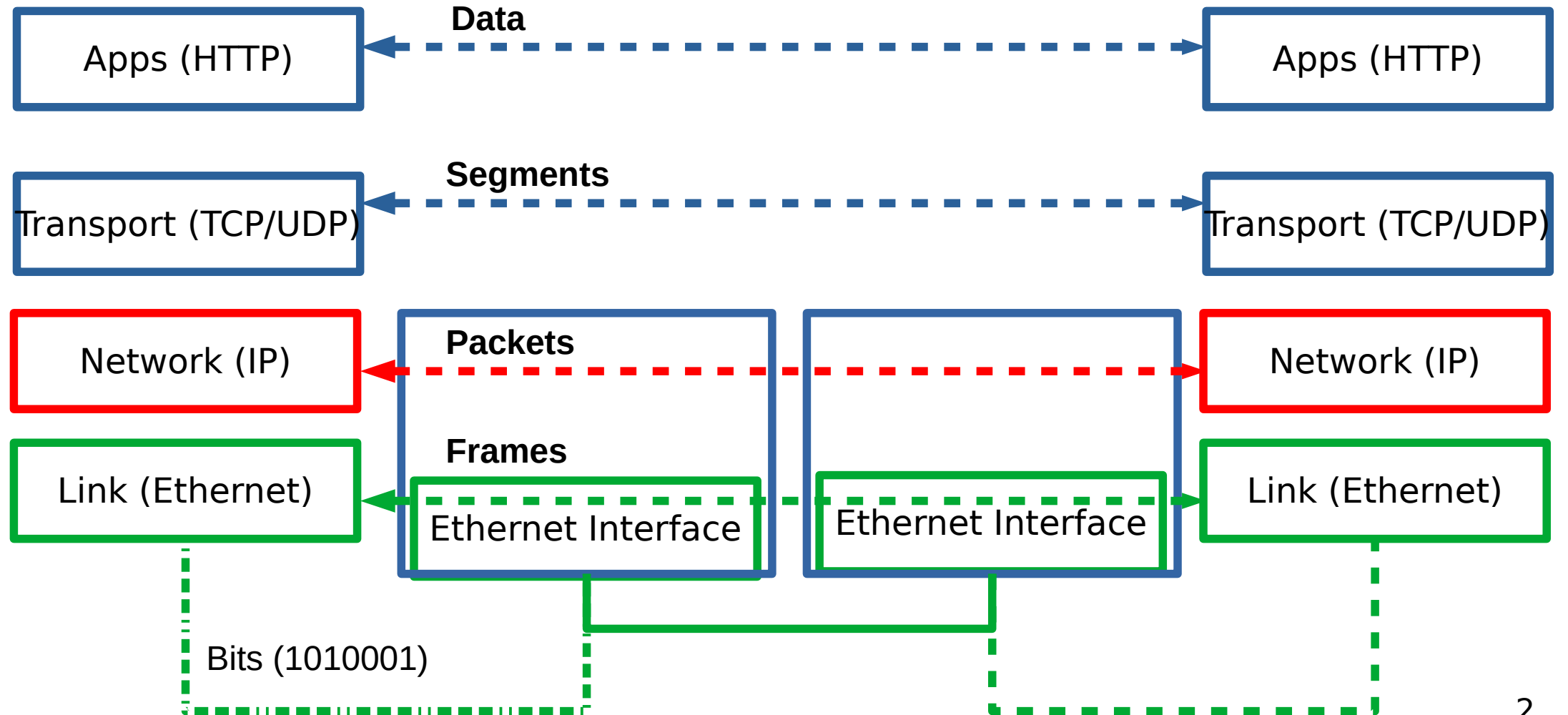
**Instructor: Susmit Shannigrahi**

**ROUTING**

**sshannigrahi@tnitech.edu**

**GTA: dereddick42@students.tnitech.edu**





# So far...

---

- We now know how communication works
- How does routing work?

# Forwarding vs Routing

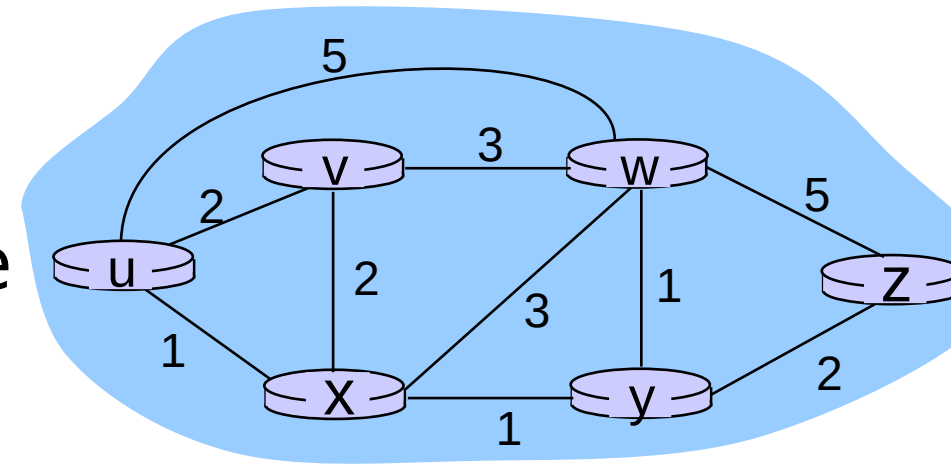
---

- Forwarding:
  - to select an output port based on destination address and routing table
  - **Local path**
- Routing:
  - process by which routing table is built
  - **End-to-end 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

# Why bother?

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



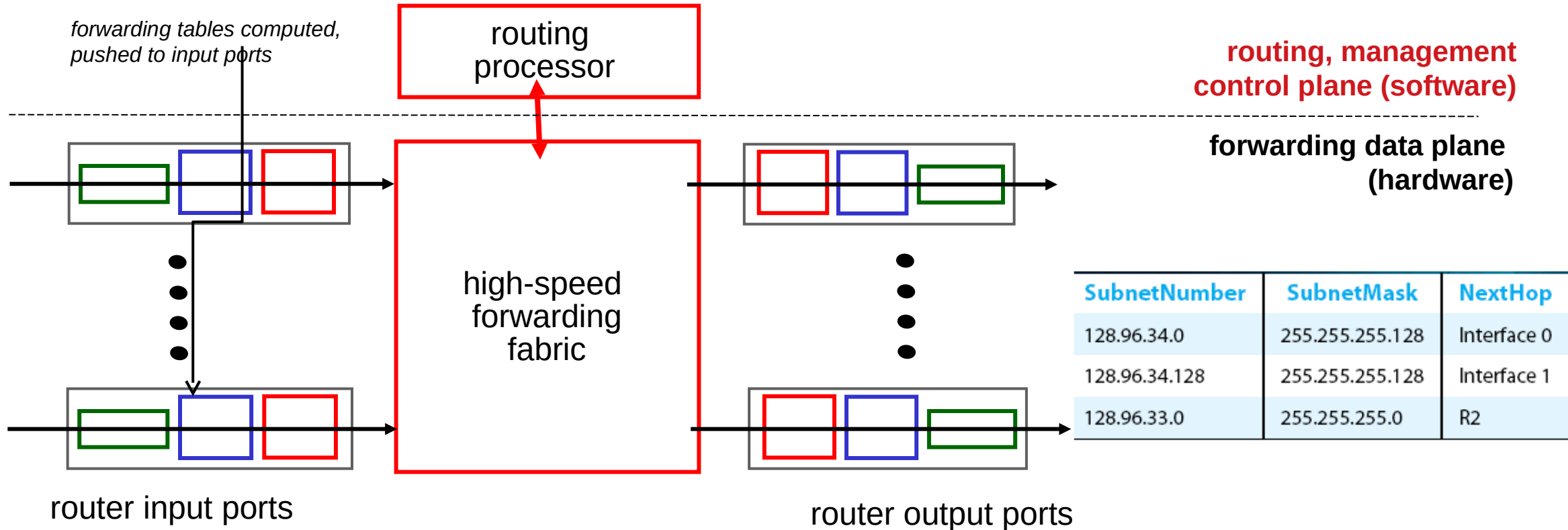
SubnetNumber	SubnetMask	NextHop
128.96.34.0	255.255.255.128	Interface 0
128.96.34.128	255.255.255.128	Interface 1
128.96.33.0	255.255.255.0	R2

# Router architecture overview

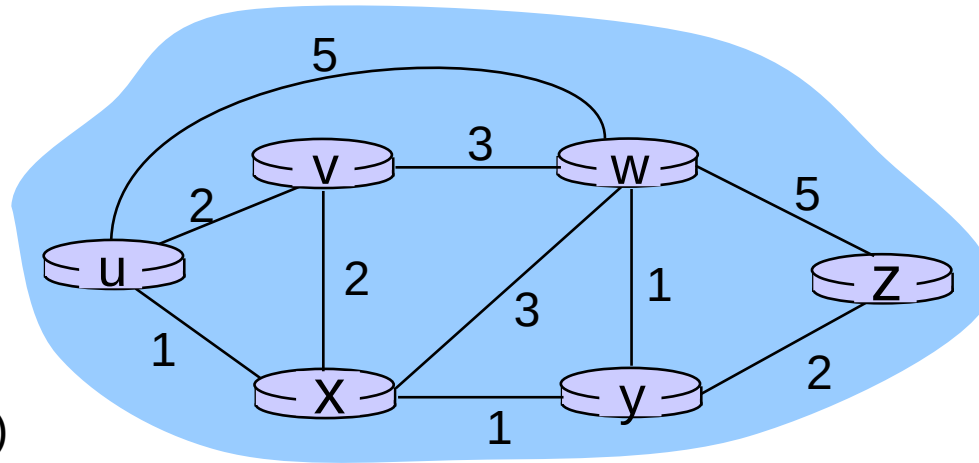
Two key router functions:

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

Control Plane = routing  
Vs  
Data Plane = forwarding



# 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  $(x,v,w,z) = \text{cost}(x,v) + \text{cost}(v,w) + \text{cost}(w,z) = 10$

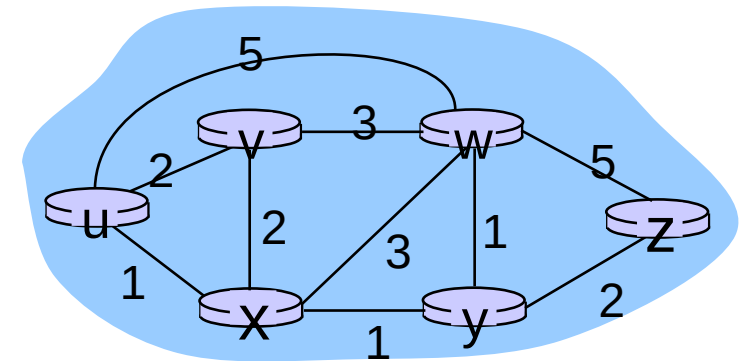
Cost  $(x,w,z) = \text{cost}(x,w) + \text{cost}(w,z) = 8$

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  $u$
  - $D(v) = \infty$  for all other nodes
  - Update  $D(v)$  as we go





# Dijkstra's Algorithm

1 **Initialization:**

2  $N' = \{u\}$

3 for all nodes  $v$

4 if  $v$  adjacent to  $u$

5 then  $D(v) = c(u,v)$

6 else  $D(v) = \infty$

7

8 **Loop**

9 find  $w$  not in  $N'$  such that  $D(w)$  is a minimum

10 add  $w$  to  $N'$

11 update  $D(v)$  for all  $v$  adjacent to  $w$  and not in  $N'$  :

12  **$D(v) = \min( D(v), D(w) + c(w,v) )$**

13 /\* new cost to  $v$  is either old cost to  $v$  or known

14 shortest path cost to  $w$  plus cost from  $w$  to  $v$  \*/

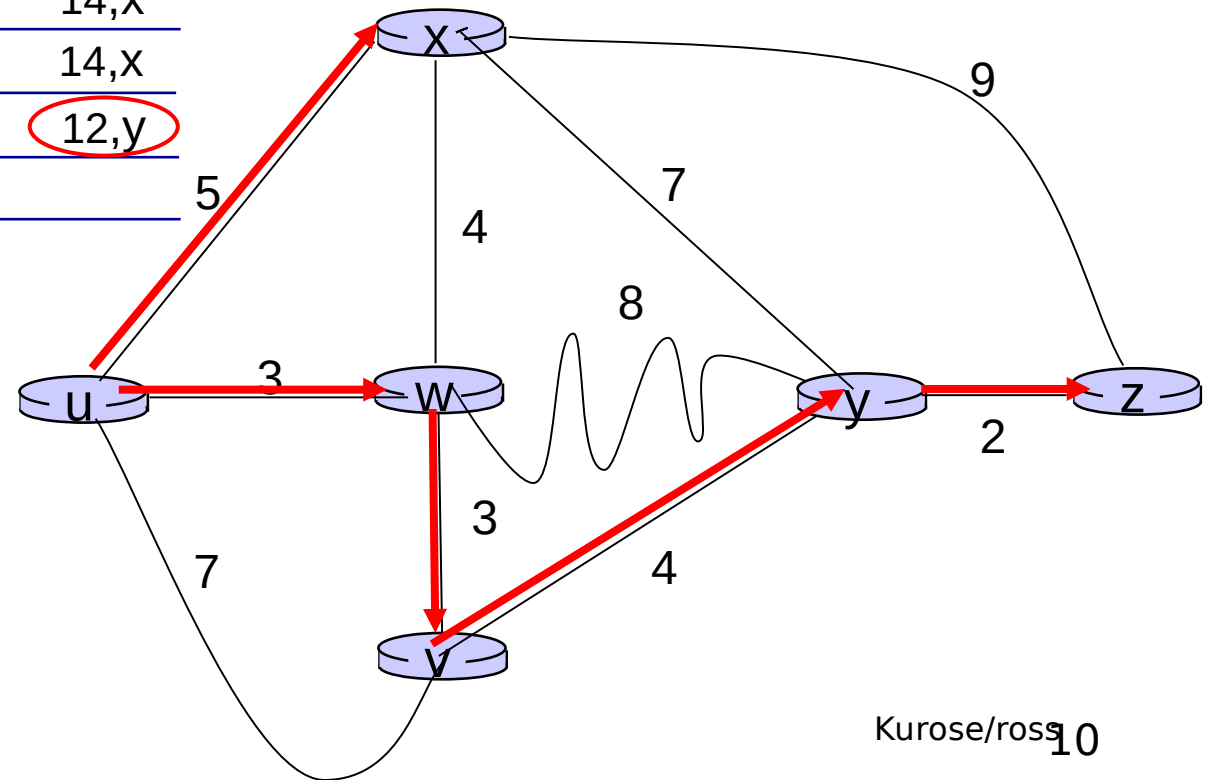
15 **until all nodes in  $N'$**

# Dijkstra's algorithm: example

Step	N'	D( <b>v</b> ) p(v)	D( <b>w</b> ) p(w)	D( <b>x</b> ) p(x)	D( <b>y</b> ) p(y)	D( <b>z</b> ) p(z)
0	u	7,u	3,u	5,u	$\infty$	$\infty$
1	uw	6,w		5,u	11,w	$\infty$
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

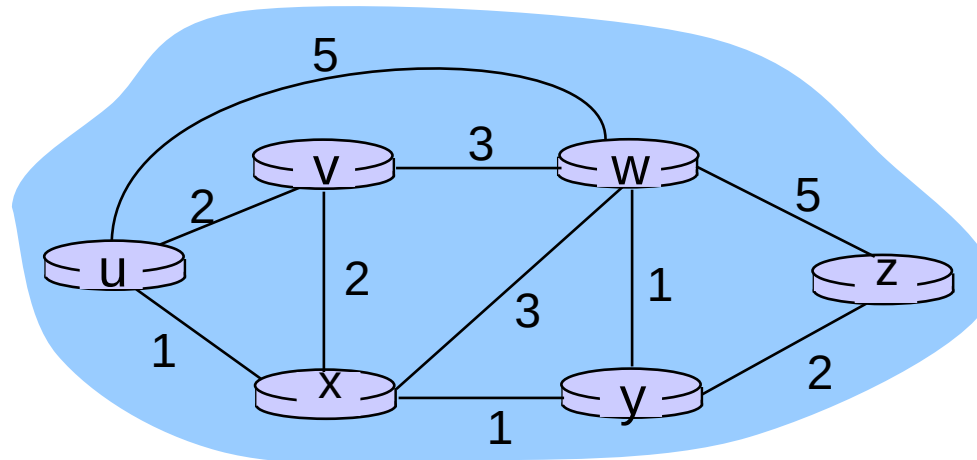
## notes:

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



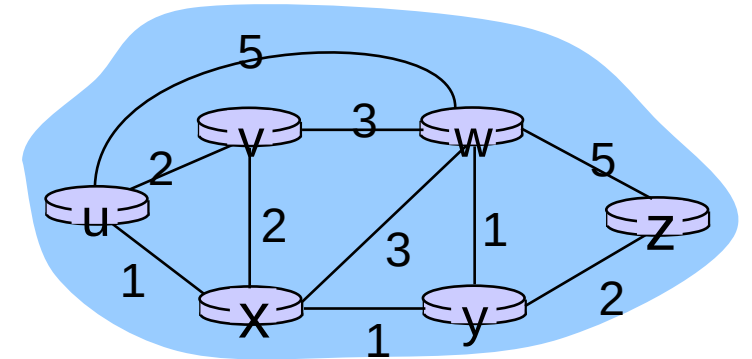
# Dijkstra's algorithm: another example

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	$\infty$	$\infty$
1	ux	2,u	4,x		2,x	$\infty$
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

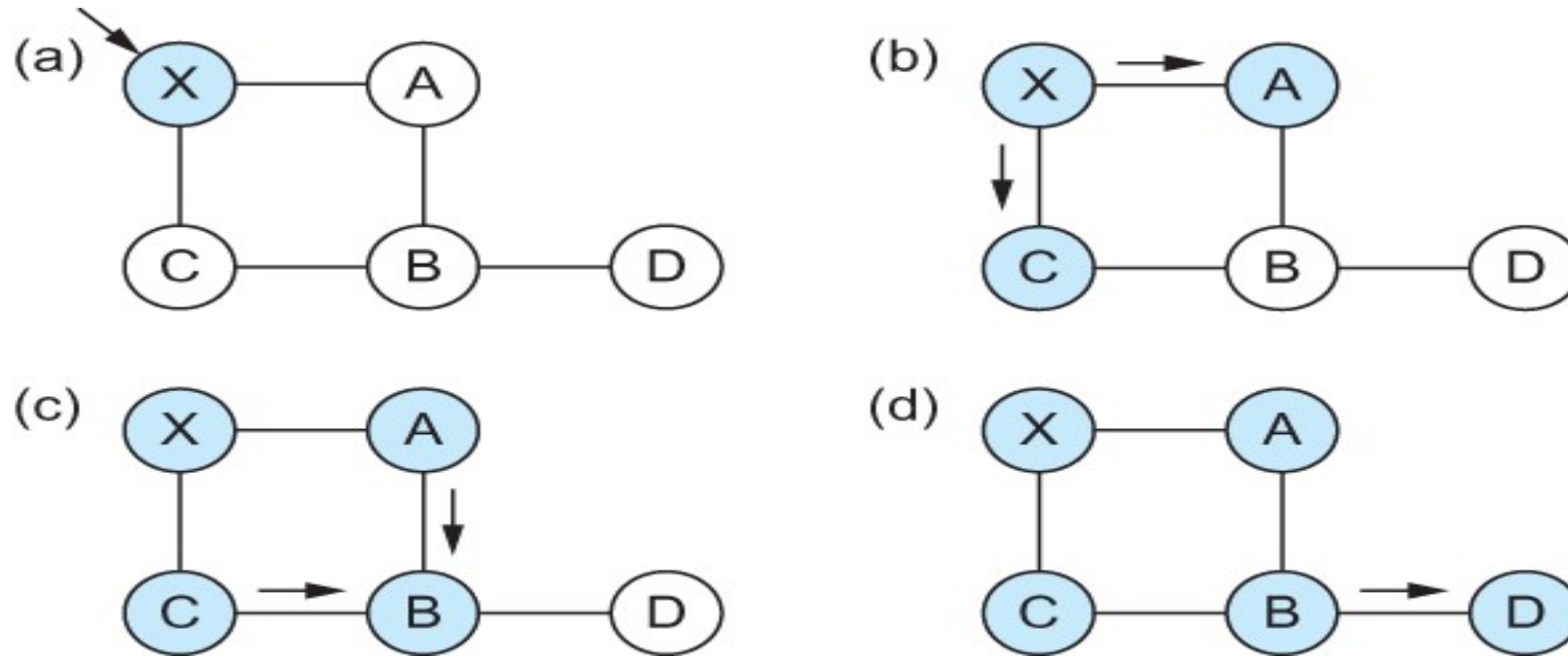


# Dijkstra's → Link State Routing

- Each node keeps track of adjacent links
- Each router broadcasts it's state
- Each router runs Dijkstra's algorithm
- Each router has complete picture of the network
- Example: Open Shortest Path First (OSPF)



# Link State Routing – controlled flooding

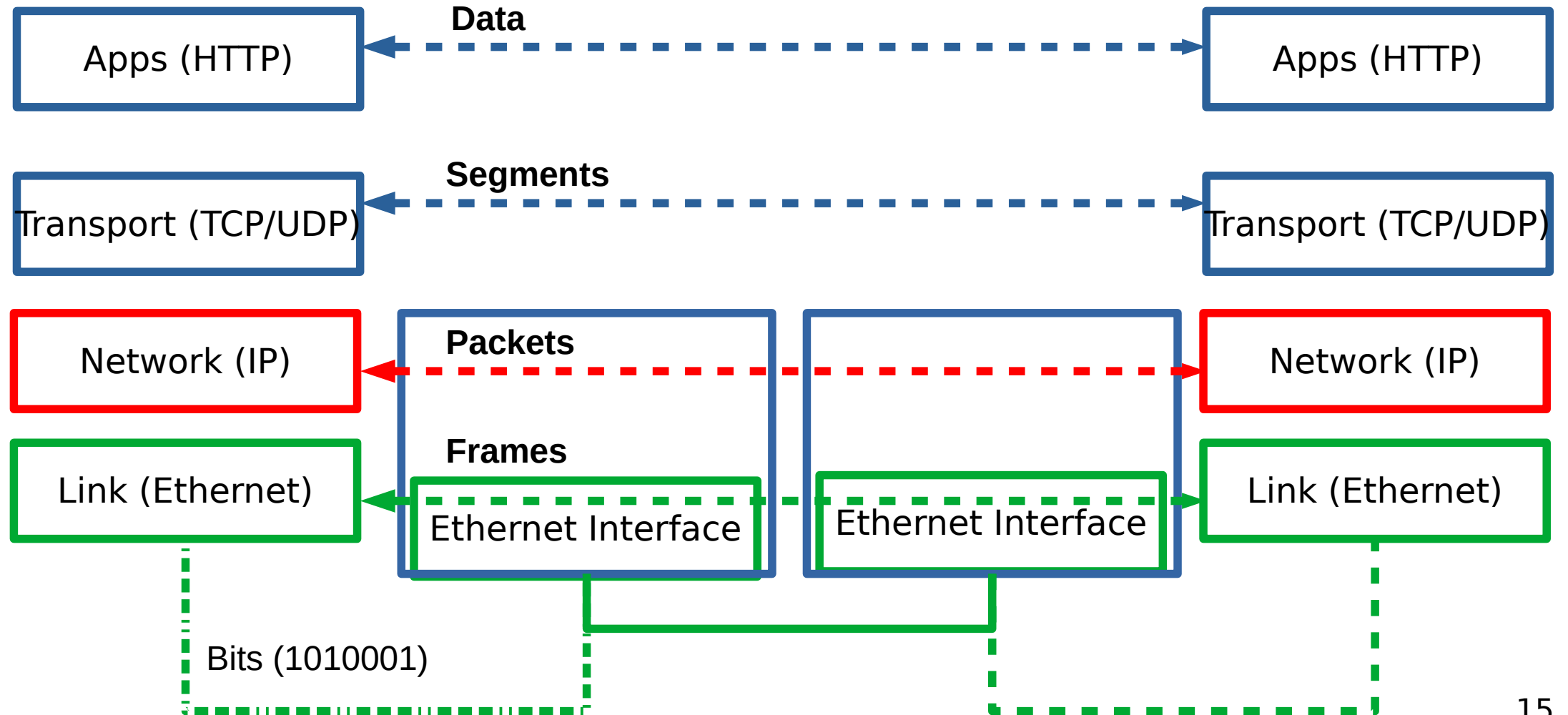


Flooding of link-state packets. (a) LSP arrives at node X; (b) X floods LSP to A and C; (c) A and C flood LSP to B (but not X); (d) flooding is complete

# 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



# Next Steps

---

Distance Vector routing  
Midterm review