

Network Layer



Instructor: C. Pu (Ph.D., Assistant Professor)

Lecture 15

puc@marshall.edu



Network Layer Functions

Recall: two network-layer functions:

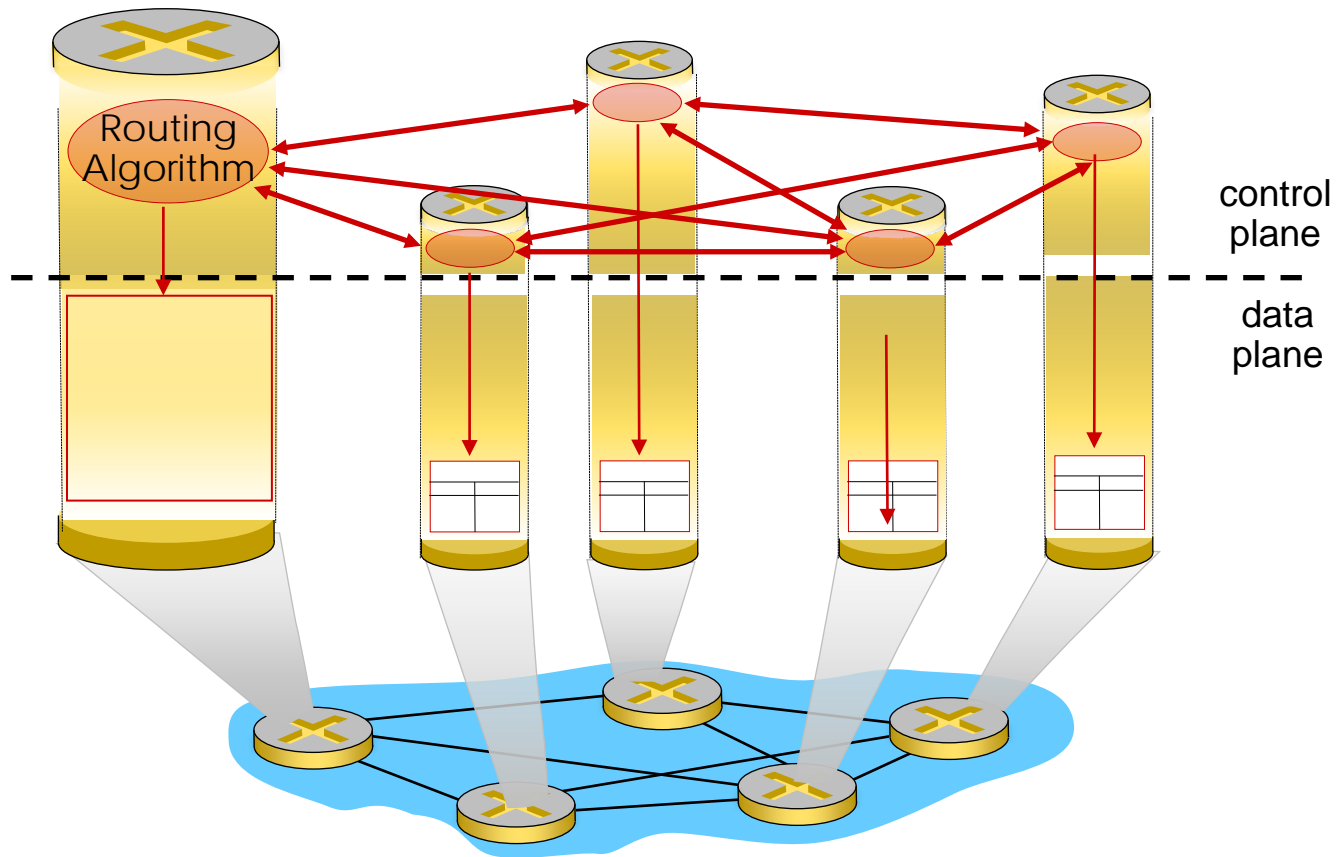
- *forwarding*: move packets from router's input to appropriate router output *data plane*
- *routing*: determine route taken by packets from source to destination *control plane*

Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)

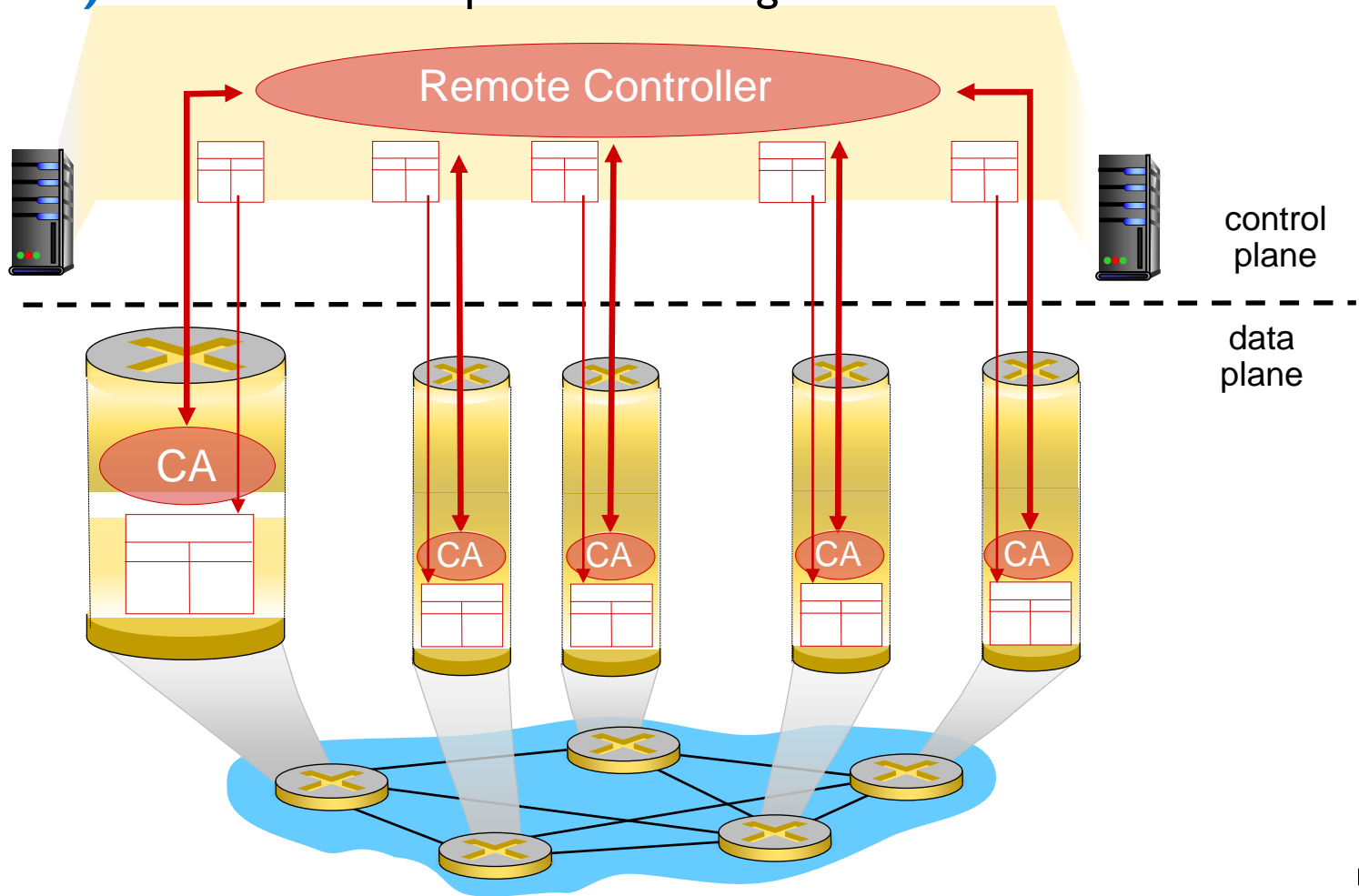
Per-router Control Plane

Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



Logically Centralized Control Plane

A distinct (typically remote) controller interacts with **local control agents (CAs)** in routers to compute forwarding tables





Routing Protocols

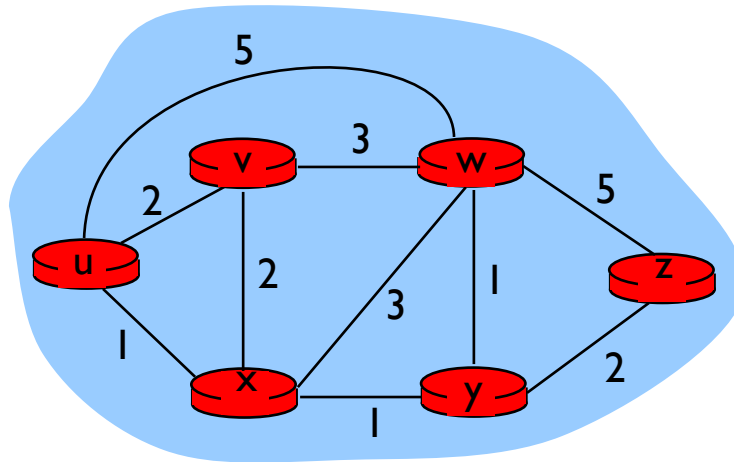
- ***Routing protocol goal:***
 - determine “**good**” paths (equivalently, routes), from sending hosts to receiving host, through network of routers
- **path:**
 - sequence of routers packets will traverse in going from given initial source host to given final destination host
- “good”: least “cost”, “fastest”, “least congested”
- routing: a “top-10” networking challenge!



Routing Protocols

- A host is attached directly to one router, the **default router** for the host.
- Whenever a host sends a packet, the packet is transferred to its default router.
- We refer to the default router of the source host as the **source router** and the default router of the destination host as the **destination router**.
- The problem of routing a packet from source host to destination host
==
the problem of routing the packet from source router to destination router

Graph Abstraction

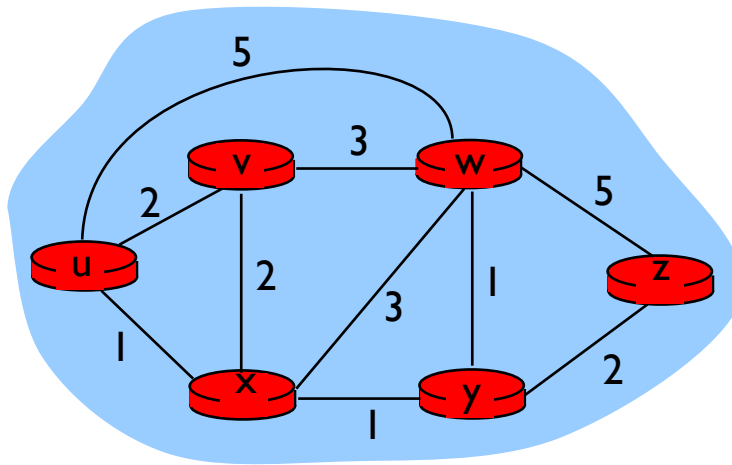


Graph: $G = (N, E)$

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

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

Graph Abstraction: Cost



- $c(x, x') = \text{cost of link } (x, x')$

- e.g., $c(w, z) = 5$

- cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path $(x_1, x_2, x_3, \dots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path



Routing Algorithm Classification

Global or decentralized information?

Global routing algorithm:

- all routers have complete topology, link cost info
- “link state” algorithms

Decentralized routing algorithm:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- “distance vector” algorithms

Static or dynamic?

Static routing algorithm:

- routes change slowly over time

Dynamic routing algorithm:

- routes change more quickly
 - periodic update
 - in response to link cost changes



A Link-State Routing Algorithm

Dijkstra's algorithm

- net. Topology and link costs known to all nodes
 - accomplished via “link state **broadcast**”
 - all nodes have same info.
- compute least cost paths from one node (“source”) to all other nodes
 - gives **forwarding table** for that node
- iterative: after k iterations, know least cost path to k dest.'s

Notation:

- $c(x, y)$: link cost from node x to y ; $= \infty$ if not direct neighbors
- $D(v)$: current value of cost of path from source to dest. v
- $p(v)$: predecessor node along path from source to v
- N' : set of nodes whose least cost path definitively known



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

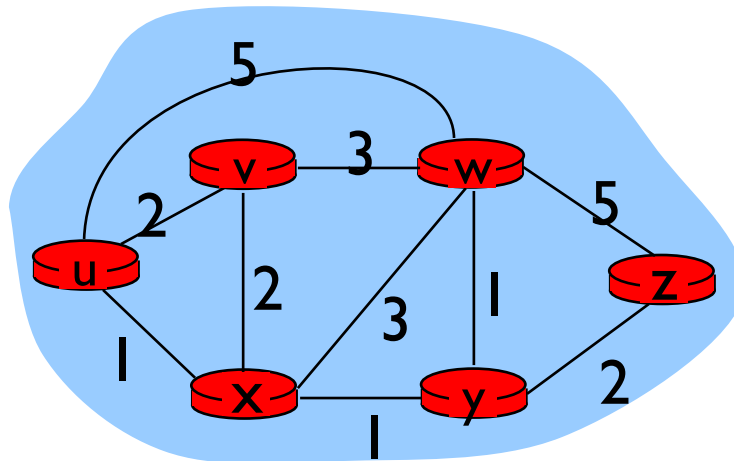
14 shortest path cost to w plus cost from w to v */

15 **until all nodes in N'**



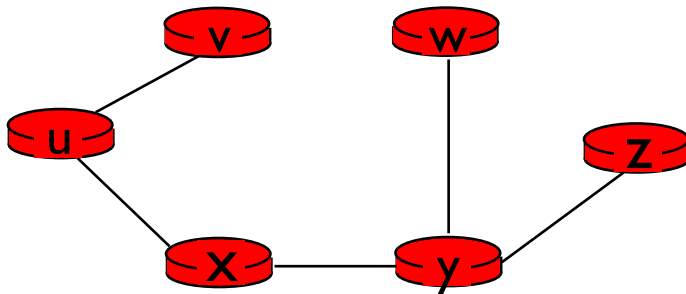
Dijkstra's Algorithm: Example

Step	N'	D(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 Algorithm: Example (cont.)

Resulting shortest-path tree from u:



Resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)