Data Communication and Computer Networks

7. Network Layer
PART-B

Dr. Aiman Hanna

Department of Computer Science & Software Engineering
Concordia University, Montreal, Canada

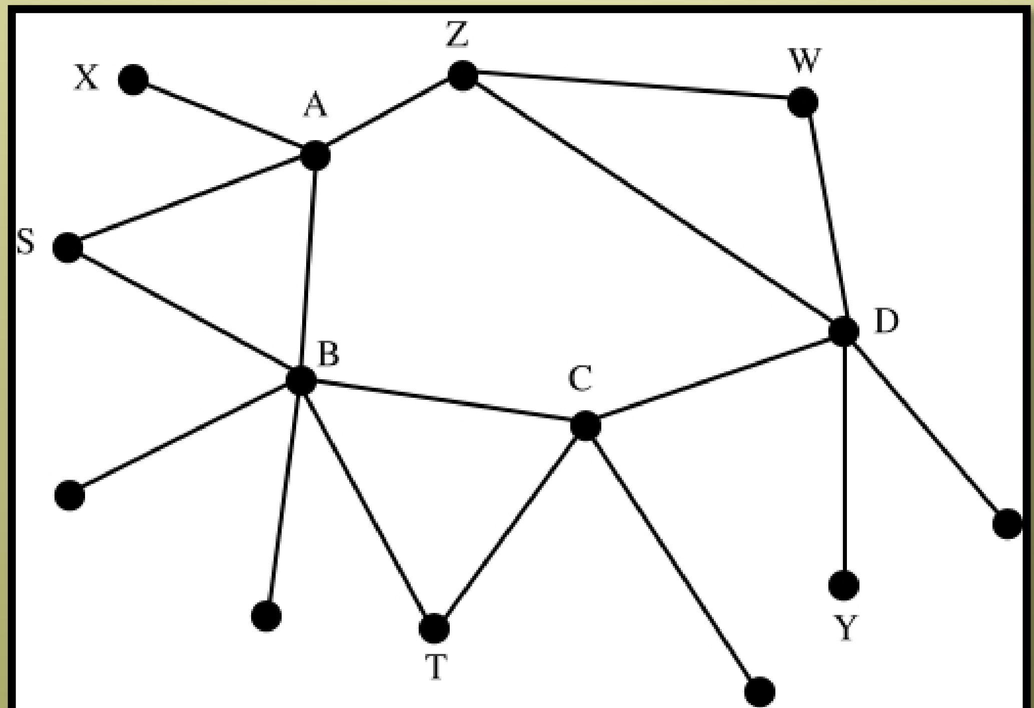
These slides have mainly been extracted, modified and updated from original slides of :
Computer Networking: A Top Down Approach, 6th edition Jim Kurose, Keith Ross
Addison-Wesley, 2013

Additional materials have been extracted, modified and updated from:
Understanding Communications and Networking, 3e by William A. Shay 2005

Wide Area Networks (WANs)

- ***Wide Area Networks***, **WANs**, are large networks that may span the globe

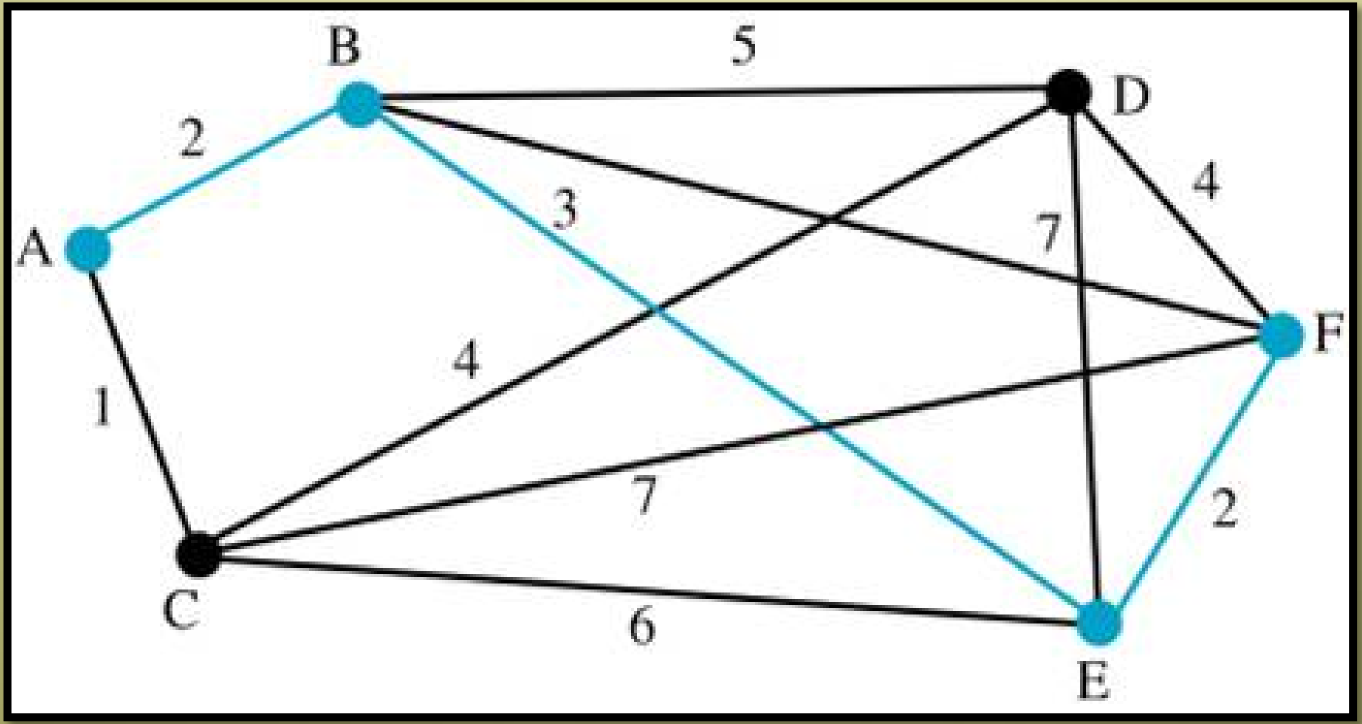
- WANs have a generalized topologies and require more sophisticated techniques to perform what they are expected to



Generalized Network Topology

Routing Tables

- these table do not normally specify the entire route
- rather, they specify the next node in a route to a specified destination, and the cost to get there



Network Example

Routing Tables

- Example: partial tables for the routers shown in the example network

DESTINATION	NEXT NODE	COST
B	B	2
C	C	1
D	C	5
E	B	5
F	B	7

(a) Partial routing table for node A

DESTINATION	NEXT NODE	COST
D	D	5
E	E	3
F	E	5

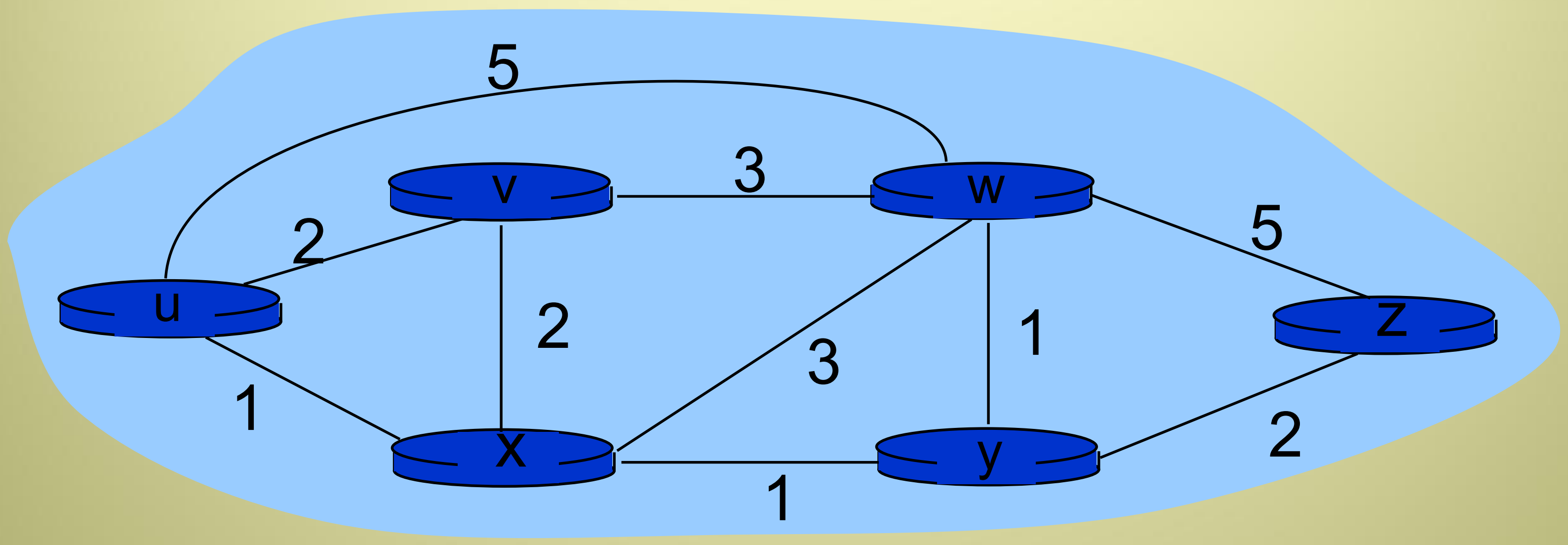
(b) Partial routing table for node B

DESTINATION	NEXT NODE	COST
F	F	2

(c) Partial routing table for node E

Partial Routing Tables for A, B & E

Graph abstraction

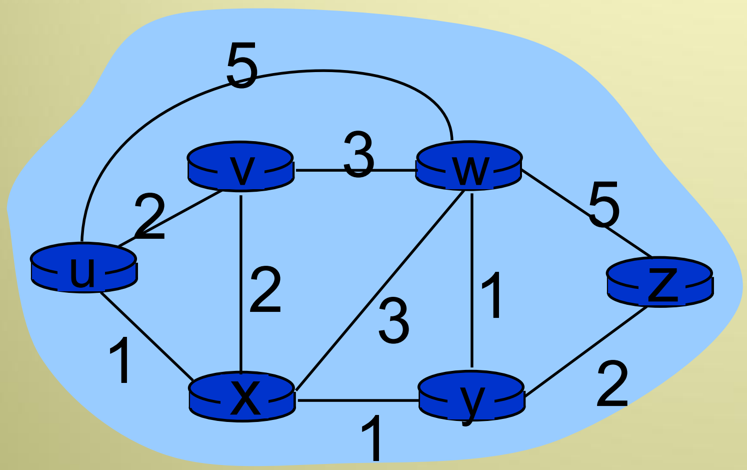


graph: G = (N,E) - Nodes and Edges

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

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

Graph abstraction: costs



$c(x,x')$ = 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 $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

key question: what is the least-cost path between u and z ?
routing algorithm: algorithm that finds that least cost path

Routing algorithm classification

Q: global or decentralized information?

global:

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

decentralized:

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

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

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

Routing algorithm classification

Centralized Routing

- all interconnection information is generated and maintained at a single central station
- that station then broadcasts this information to all network nodes, so they can define their own routing tables
- one way to maintain the central information is through **routing matrix**

Routing algorithm classification

Centralized Routing

		A	B	C	D	E	F
	A	—	B	C	C	B	B
	B	A	—	A	D	E	E
source node	C	A	A	—	D	E	F
	D	C	B	C	—	F	F
	E	B	B	C	F	—	F
	F	E	E	C	D	E	—

Routing Matrix for the Example Network

Routing algorithm classification

Distributed Routing

- no central control; each node must determine and maintain routing information independently
- this can be achieved by knowing the neighbors, cost to get to a neighbor and the cost from that neighbor to a destination
- more complex than centralized since each node must communicate with each of its neighbors instead of just one central station
- the complexity is driven also by the fact that the devices have a very limited knowledge of the entire network

Routing algorithm classification

Static Routing

- once the node determines the routing table, it does not change it
- in other words, the initial cheapest path may not really be the cheapest path after sometime, yet it is always considered as the cheapest path
- the assumption here is that the conditions that led to the initial definition of the tables do not change
- sometimes, this is a reasonable assumption, when?

Routing algorithm classification

Adaptive Routing

- allows the network to respond to changes and update its routing tables accordingly
- if the cost of the route to be used changed, then adapt and use another route
- could this lead to serious problems sometime?
- in general, it is difficult to implement adaptive routing efficiently, why?

A Link-State Routing Algorithm

Dijkstra's algorithm

- sometimes called the ***Shortest-path Algorithm*** or ***Forward Search Algorithm***
- centralized, static algorithm, however it can be made adaptive by executing it periodically
- a node executing this algorithm is required to know the link costs among the nodes
- each node executes this algorithm to determine the cheapest route to each network node

A Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes 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 destinations

notation:

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

Dijsktra’s Algorithm

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

Loop
  find w not in S such that D(w) is a minimum
  add w to S
  update D(v) for all v adjacent to w and not in S :
      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 S
```

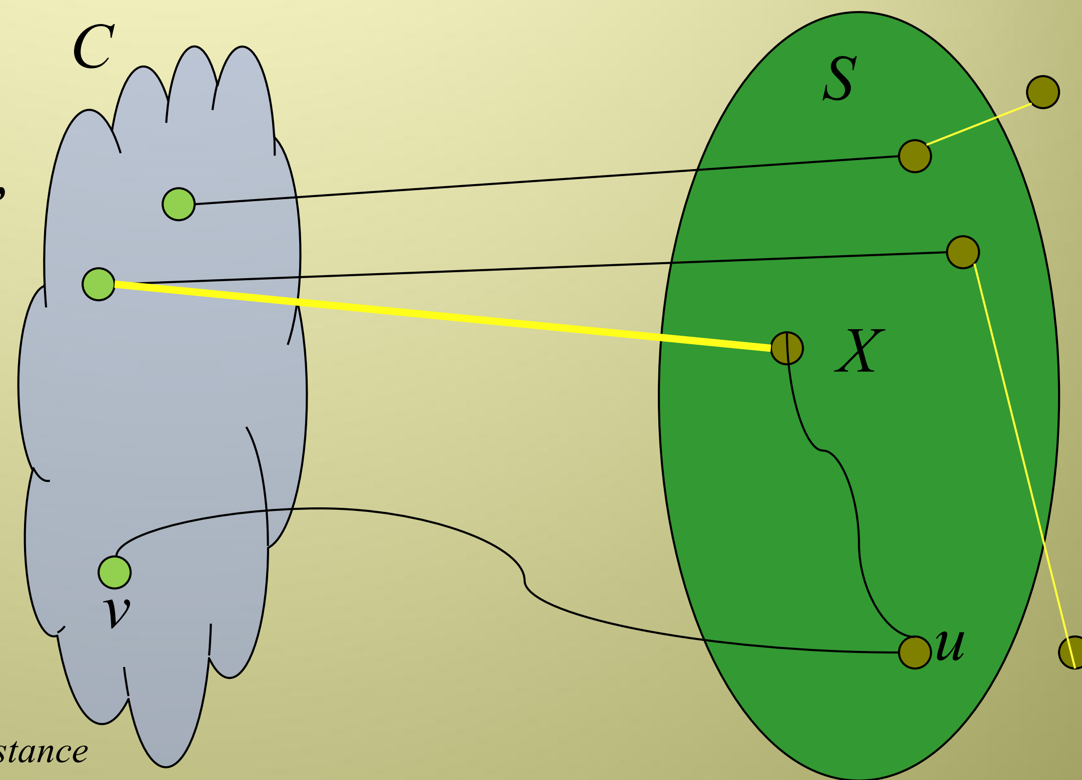
Dijsktra's Algorithm

- Iterative procedure.

  C: Set of nodes to which cheap route is known,

  S: Set of nodes directly connected to a node in C,

  X: A node in *S* to where cheap path found



Distances yet to be determined

Distances to potential vertices

Potential vertex with shortest distance

X *Selected vertex to move to C*

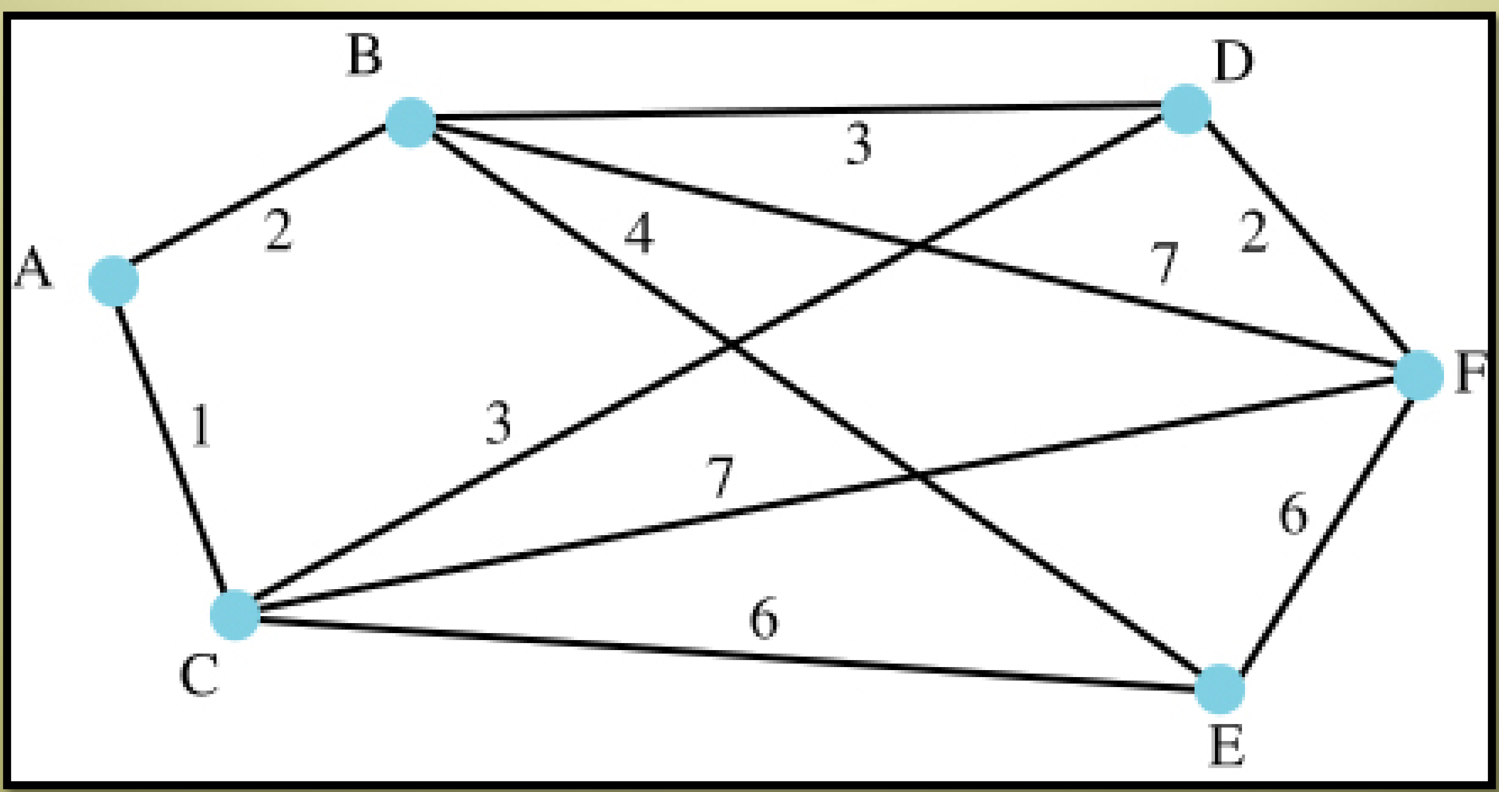
Different routes to u.

Is there a cheaper route through X to other vertices in S? If so, update distances to these vertices.

Dijsktra's Algorithm

- In details, the algorithm performs the repeated moving operations from S to C as follows:
  - Move one vertex from S to C. Let us refer to that vertex as X. This vertex, X, must satisfy the following:
    1) It is either directly connected to v or directly connected to another vertex that is already in C
    2) Has the smallest distance to v among all the potential vertices that are considered for movement to C
  - Whenever the vertex X is moved to C, its distance is the smallest distance (shortest path) to v
  - Additionally, once this vertex X is chosen, move it to C and recheck all distances of all vertices that have direct connection to X and are NOT yet in C. If a smaller value than what we already have is found, update this value
  - In each step, we also update how to go to these vertices (that is, through which vertex); we refer to this vertex as the ***prior function***
  - The steps are repeated until all the graph vertices are moved to C. The final obtained distances represent the shortest paths, and the prior functions indicate the direction from v (through which vertex) to obtain these shortest paths to each of the other vertices in the graph

Dijsktra's Algorithm - Example



Network and Associated Connection Costs

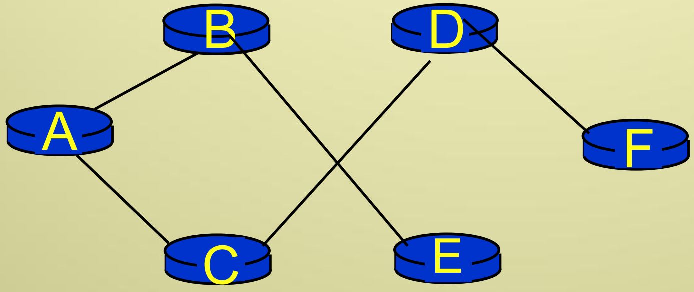
Dijsktra's Algorithm - Example

				Cost function						Prior function					
	C	Potential elements of S	**X**	A	B	C	D	E	F	A	B	C	D	E	F
0	{}	{A}	**A**	**0**	∞	∞	∞	∞	∞	**A**	-	-	-	-	-
1	{A}	{B,C}	**C**	0	2	**1**	∞	∞	∞	A	A	**A**	-	-	-
2	{A, C}	{B,D,E,F}	**B**	0	**2**	1	4	7	8	A	**A**	A	C	C	C
3	{A,B,C}	{D,E,F}	**D**	0	2	1	**4**	*6*	8	A	A	A	**C**	*B*	C
4	{A,B,C,D}	{E,F}	**E**	0	2	1	4	**6**	*6*	A	A	A	C	**B**	*D*
5	{A,B,C,D,E}	{F}	**F**	0	2	1	4	6	**6**	A	A	A	C	B	**D**

Note: See comments attached to this slide for detailed information on how the operations progressed.

Dijkstra's algorithm: example

resulting shortest-path tree from A:



resulting forwarding table in A:

destination	link
B	(A,B)
C	(A,C)
D	(A,C)
E	(A,B)
F	(A,C)

Hierarchical routing

our routing study thus far - idealization

- all routers identical
- all run the same routing algorithm
- network "flat"

… *not* true in practice

scale: with 600 million destinations:

- cannot store all dest's in routing tables!
- routing table exchange would swamp links!

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network, while still being able to connect to other networks

Hierarchical routing

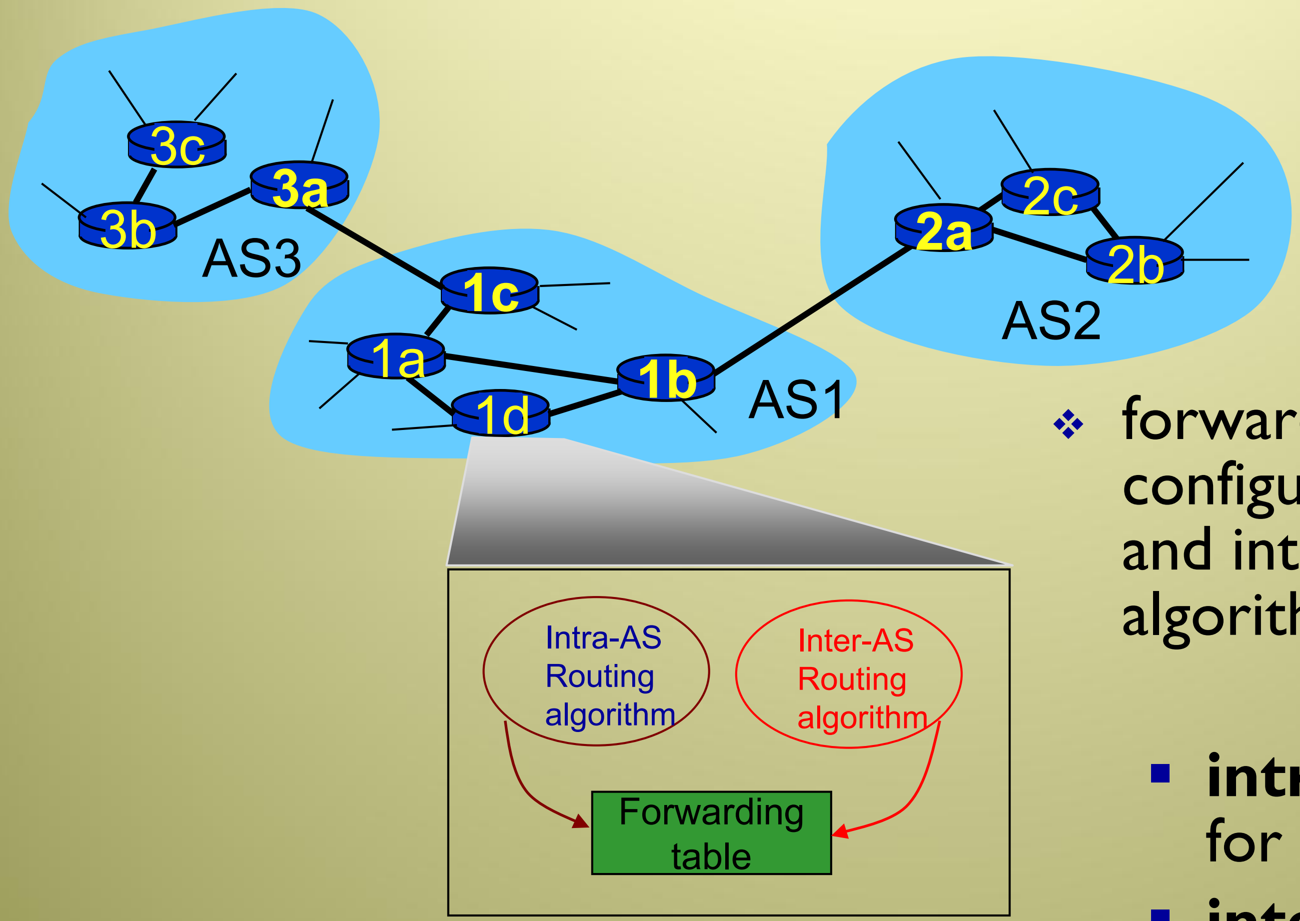
- organize routers into regions, "autonomous systems" (AS)

- routers in same AS:
  - typically under the same admin control (i.e. operated by same ISP, or belonging to same company)
  - run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different **intra-AS routing protocol**

gateway router:

- an "edge" of its own AS
- has the task of forwarding packets to destination outside its AS
- has link to router in another AS

Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - **intra**-AS sets entries for internal dests
  - **inter**-AS & intra-AS sets entries for external dests

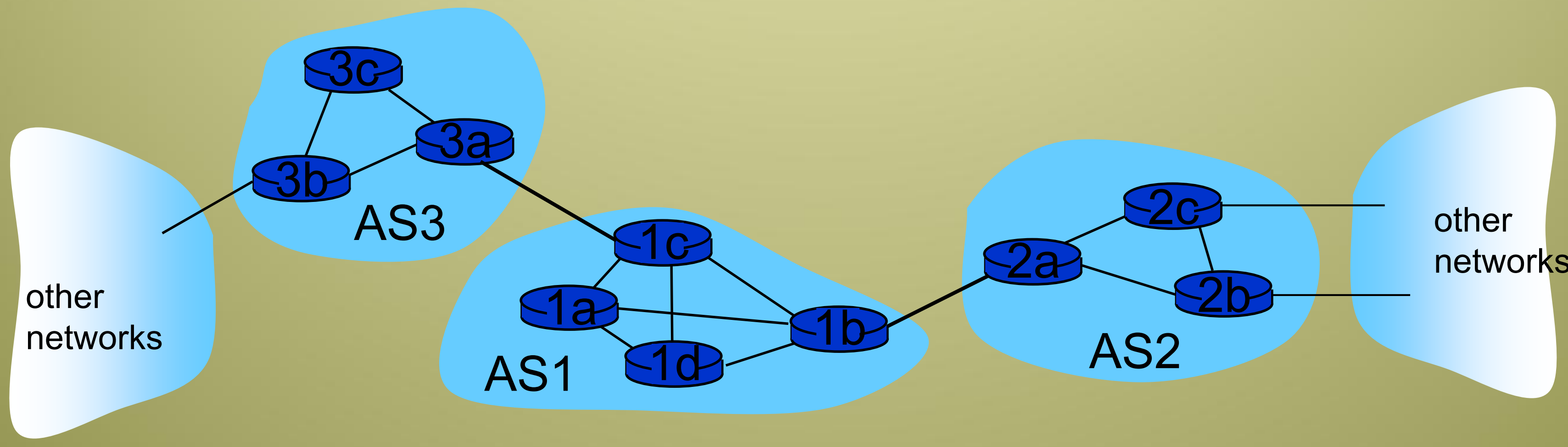
Inter-AS tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

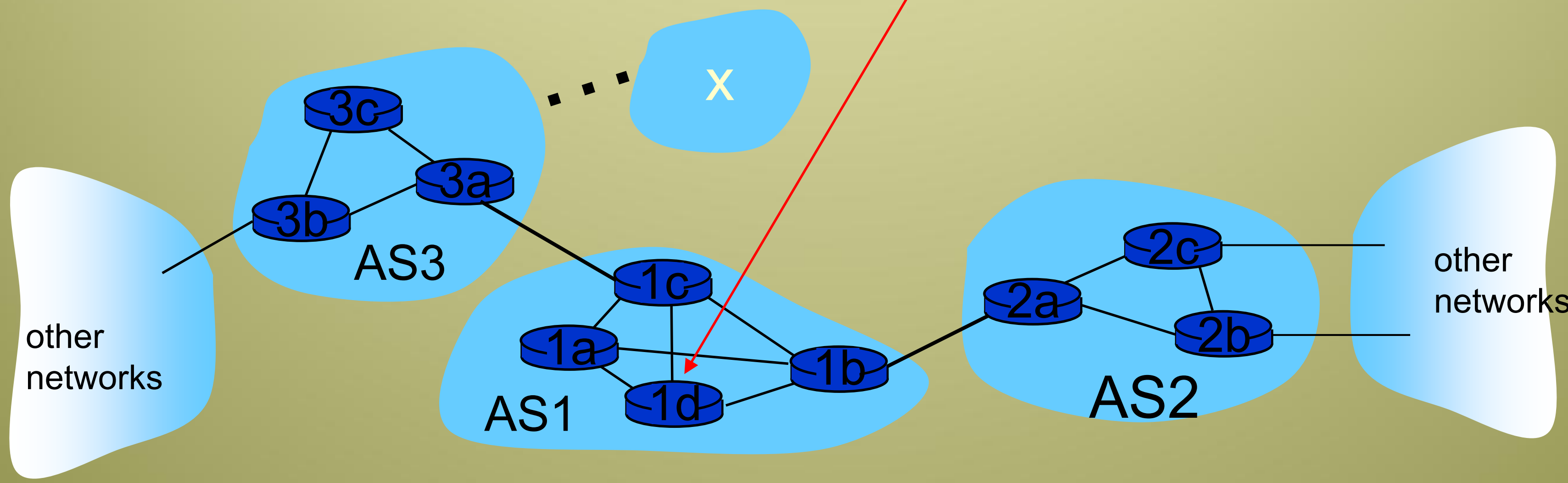
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!



Example: setting forwarding table in router 1d

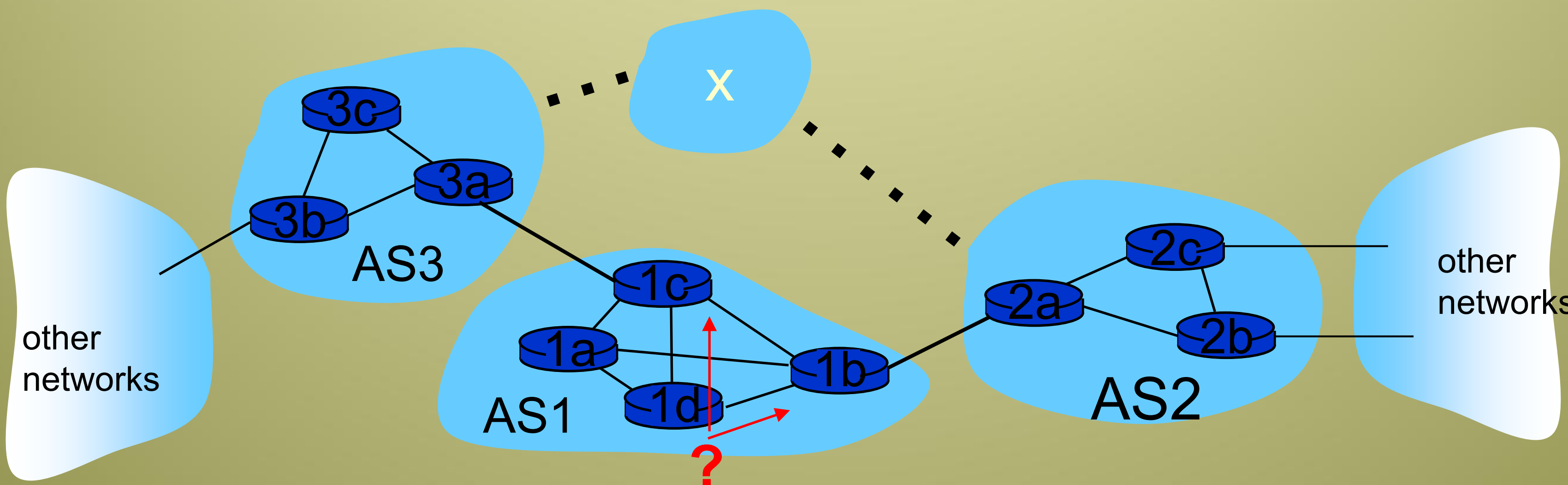
- suppose AS1 learns (via inter-AS protocol) that subnet *x* reachable via AS3 (gateway 1c), but not via AS2
  - inter-AS protocol propagates reachability info to all internal routers
- router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c
  - installs forwarding table entry *(x,I)*



Example: choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.

- to configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest x
  - this is also job of inter-AS routing protocol!



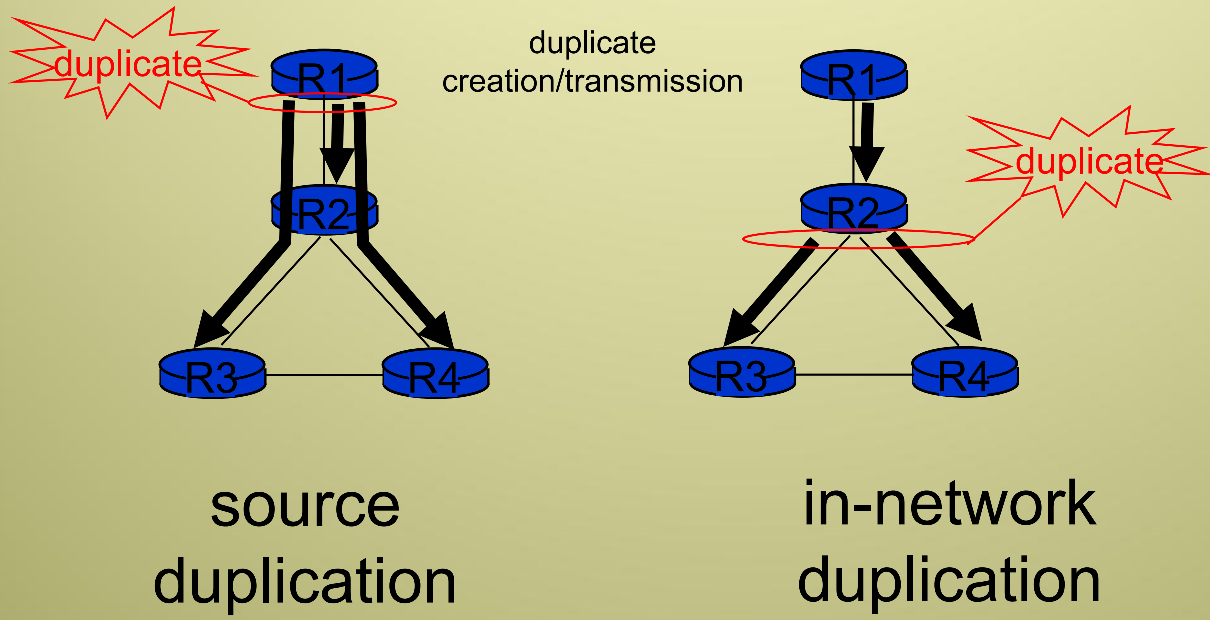
Example: choosing among multiple ASes

- if more than one gateway can be used, then the router must determine which one packets must be forwarded
- one common approach is the: ***hot-potato routing***
- *hot potato routing:*
  - *AS gets rid of the packet as quickly as possible*
  - *send* packet towards closest of the two (or the many) routers

learn from inter-AS protocol that subnet *x* is reachable via multiple gateways → use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways → hot potato routing: choose the gateway that has the smallest least cost → determine from forwarding table the interface I that leads to least-cost gateway. Enter *(x,I)* in forwarding table

Broadcast routing

- deliver packets from source to **all** other nodes
- source duplication (**N-way-unicast**) is inefficient:

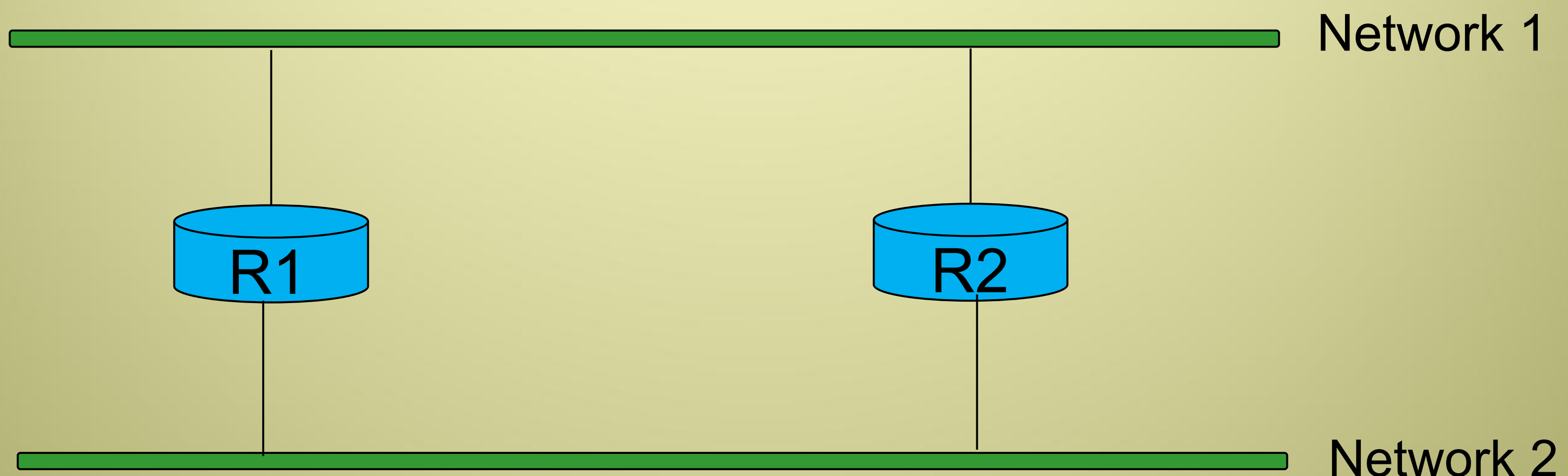


- source duplication: how does source determine recipient addresses?

In-network duplication

- *uncontrolled flooding:* when node receives broadcast packet, sends copy to all neighbors
  - problems: cycles & broadcast storm
- *controlled flooding:* node only broadcasts pkt if it hasn't broadcast same packet before
  - node keeps track of packet ids already broadacsted
  - or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source
- *spanning tree:*
  - no redundant packets received by any node

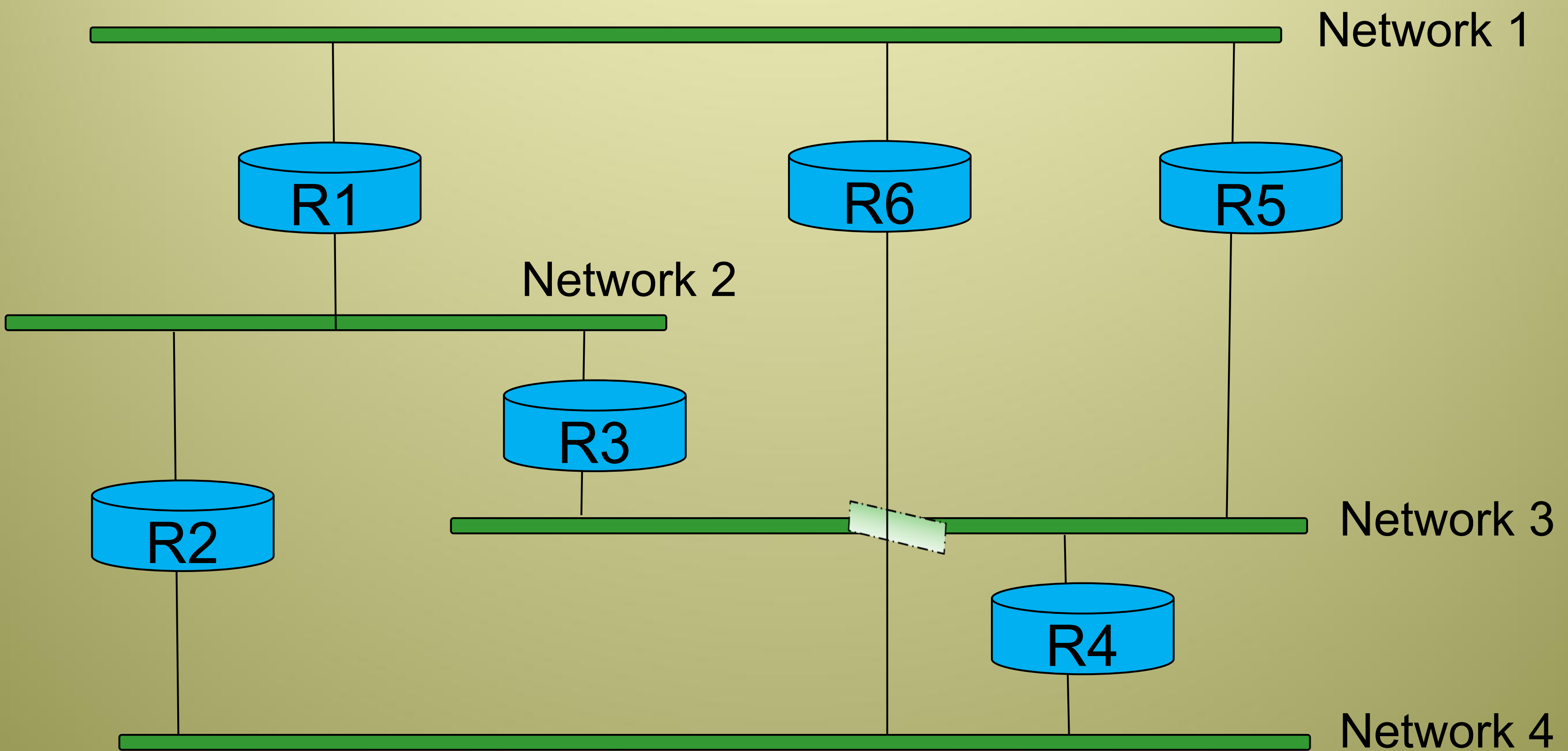
Uncontrolled flooding



- What happens when a devices on network 1 sends a packet to another on network 2?

Uncontrolled flooding

- no multiple routers in between any two networks. Is there still a problem now?



Sequence-number controlled flooding

- source node puts its address (or a unique identifier), as well as a **broadcast sequence number** into the broadcast packet, then
- sends a copy to each of its neighbors
  - problems: potential cycles & broadcast storm
- each node maintains a list of the source address/sequence number for each broadcast packet that it received, duplicated, and forwarded
- if a received packet is in that list, the packet is dropped; if not, the packet is duplicated and forwarded to all neighbors, except the one it came from

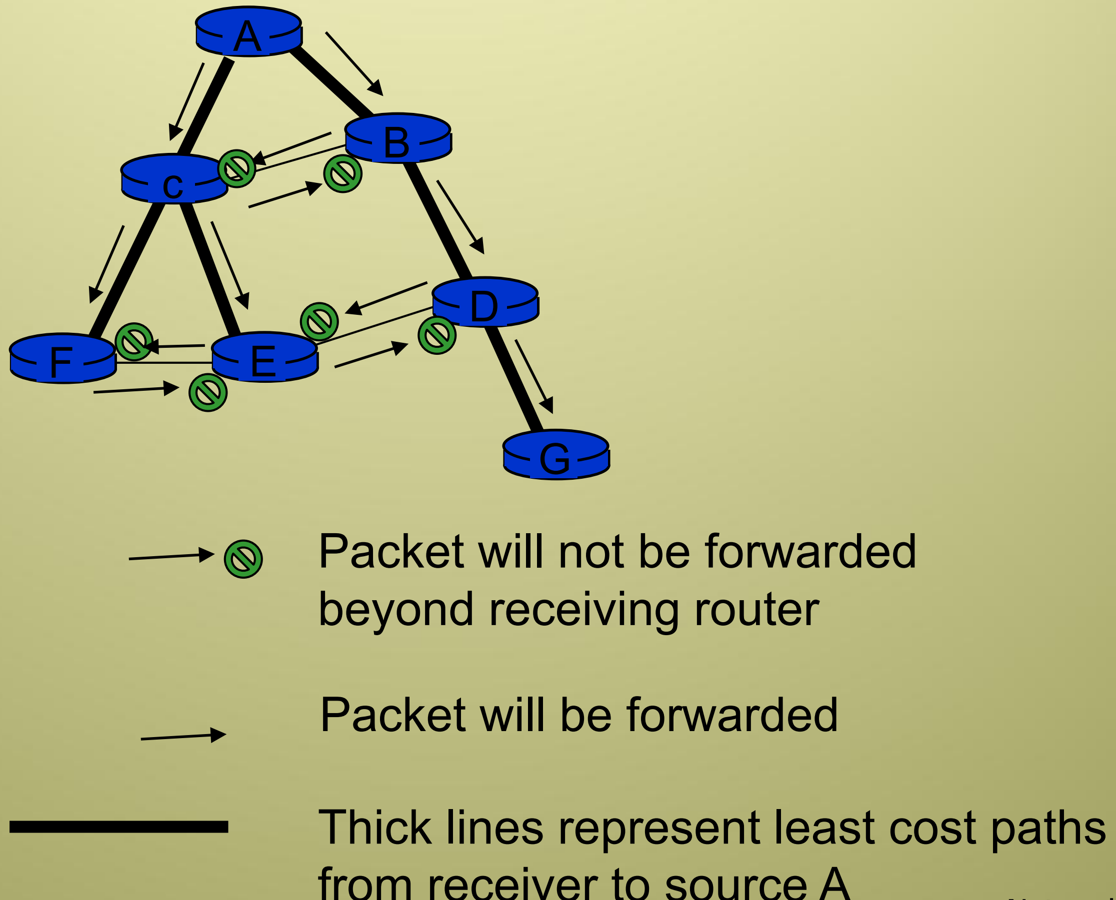
Reverse Path Forwarding (RPF) controlled flooding

- when a router receives a broadcast packet from a source address, it forwards it to all its neighbors (except where it came from) ONLY IF:
  - Packet arrived on the link that is on its own shortest unicast path back to the source
  - Otherwise, packet is discarded

- packets from any unexpected paths are dropped since the router knows that it either will receive, or has already received, a copy of this packet on the link that is on its own shortest path back to the sender

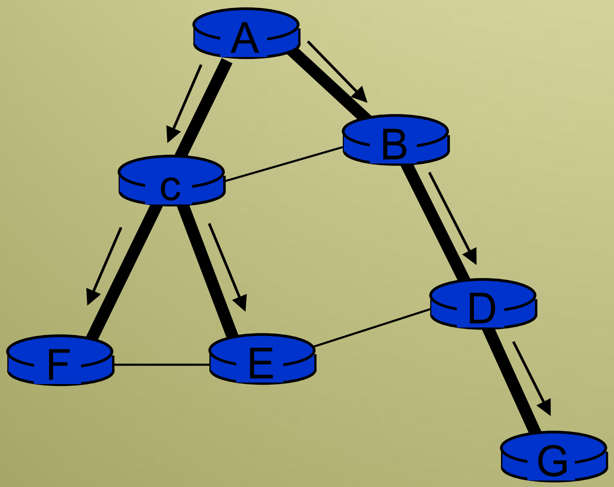
Reverse Path Forwarding (RPF) controlled flooding

- Example: packets are sent from source A.
  - least-cost paths are represented by thick lines

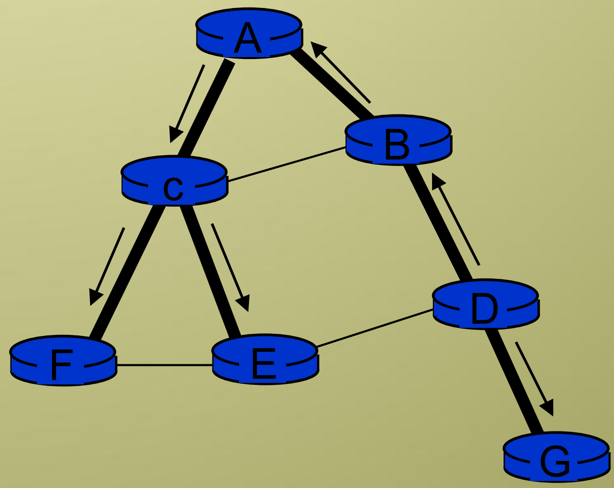


Spanning tree

- both sequence-number and RPF controlled flooding do not completely avoid the transmission of redundant broadcast packets
- first construct a spanning tree
- nodes then forward/make copies only along spanning tree



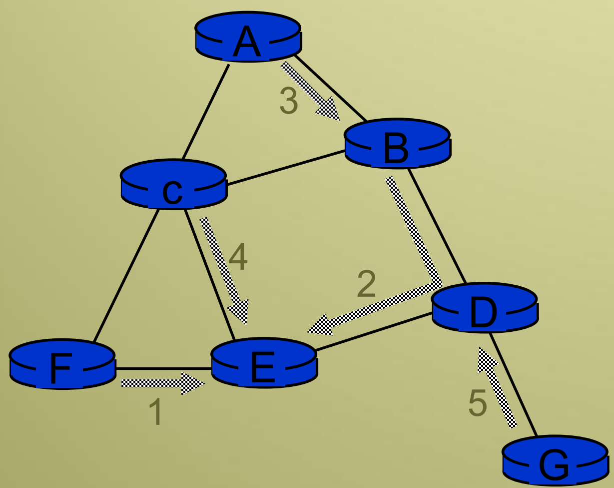
(a) broadcast initiated at A



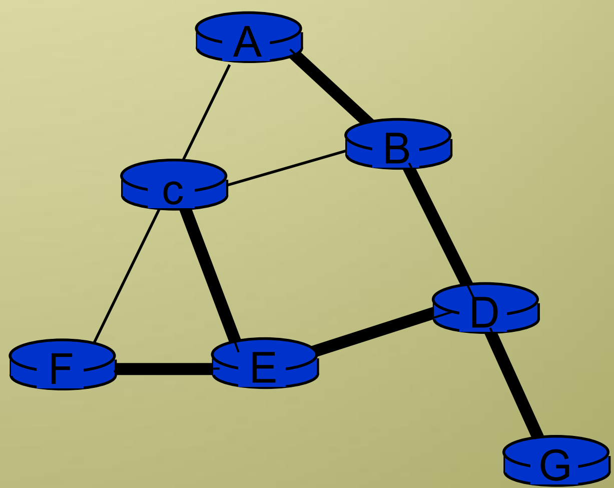
(b) broadcast initiated at D

Spanning tree: creation

- center node
- each node sends unicast join message to center node
  - message forwarded until it arrives at a node already belonging to spanning tree



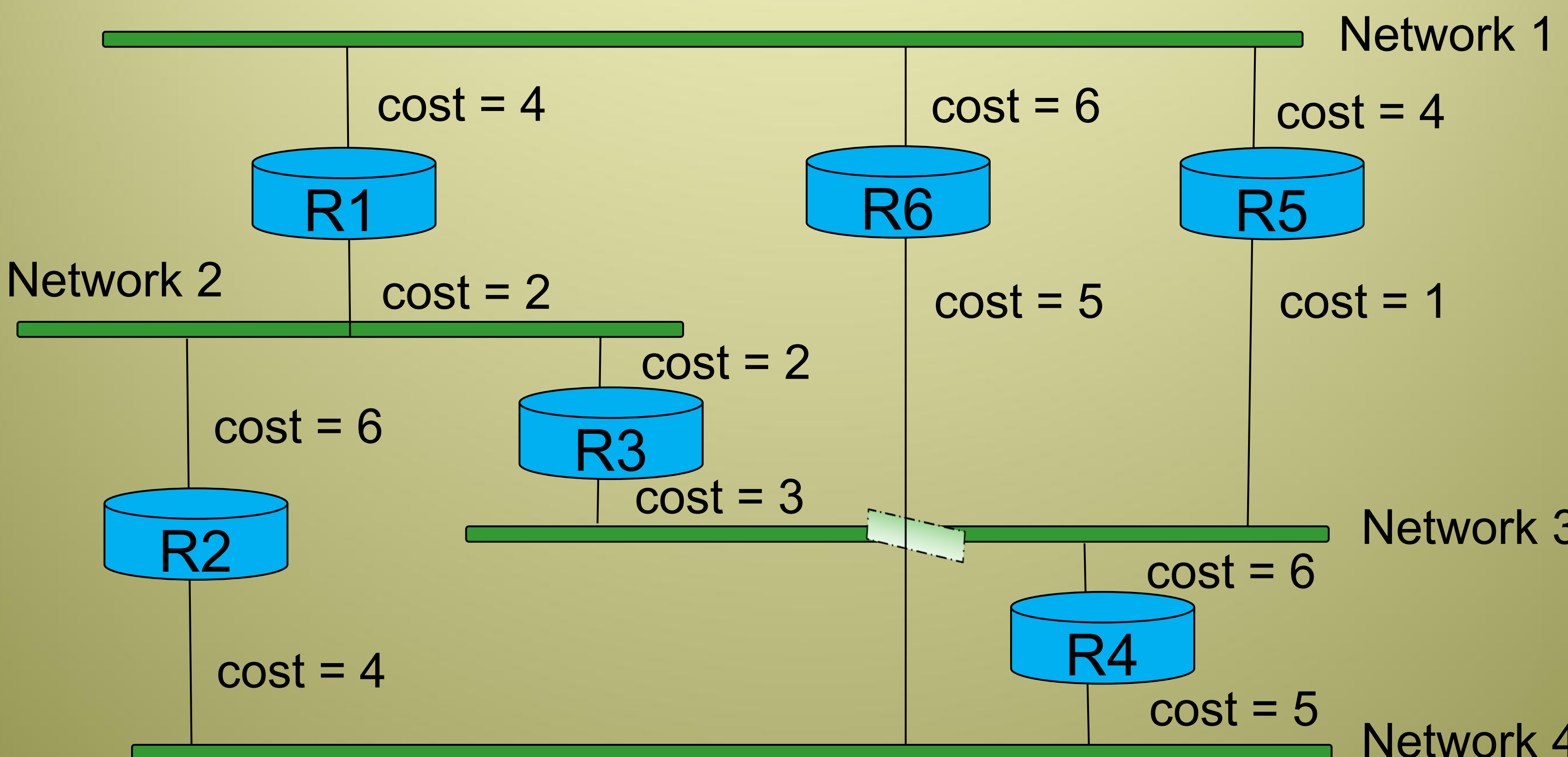
(a) stepwise construction of spanning tree (center: E)



(b) constructed spanning tree

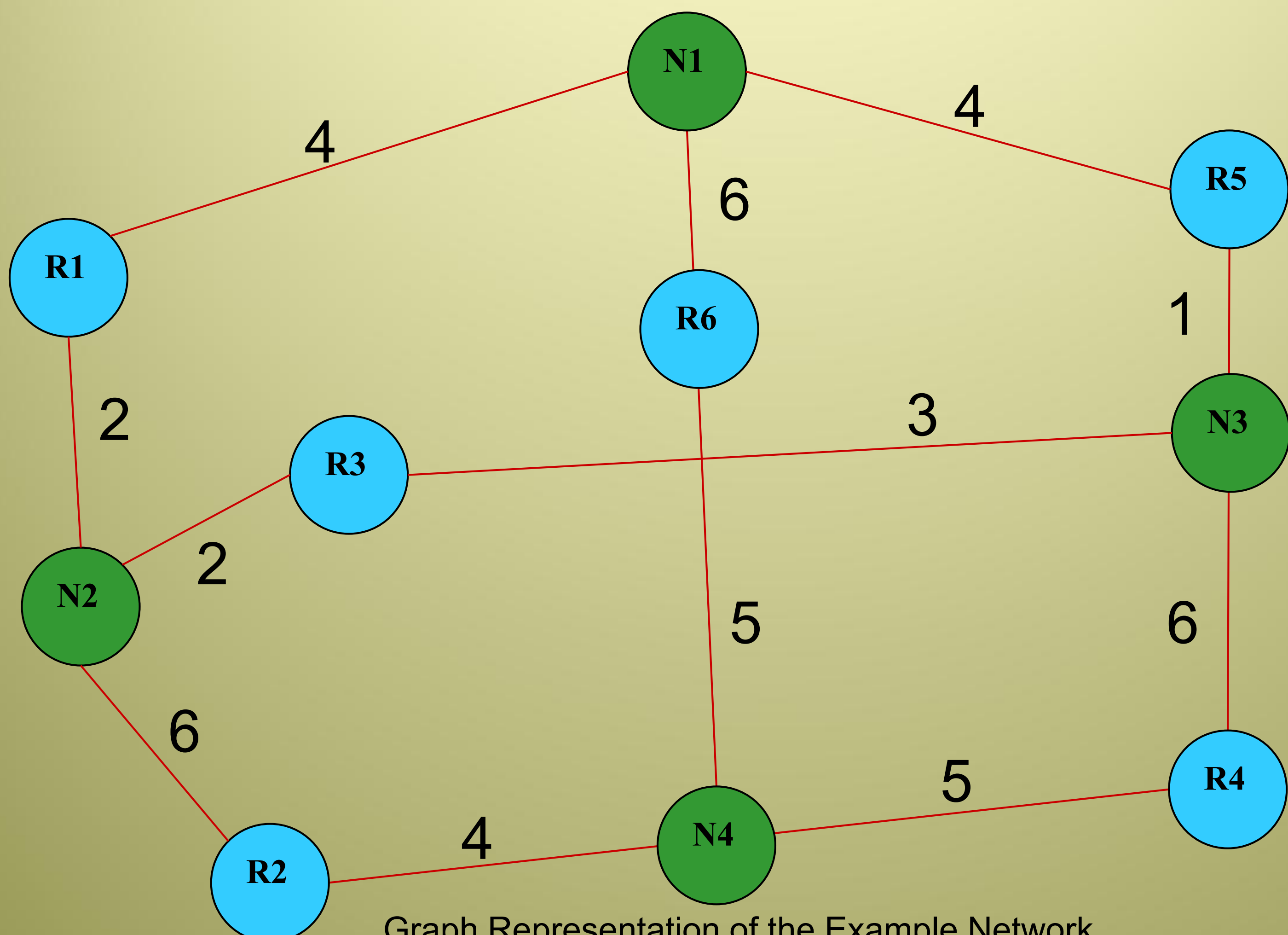
Spanning tree: creation

- If each link has an associated cost, the spanning tree can be constructed based on this cost



Spanning tree: creation

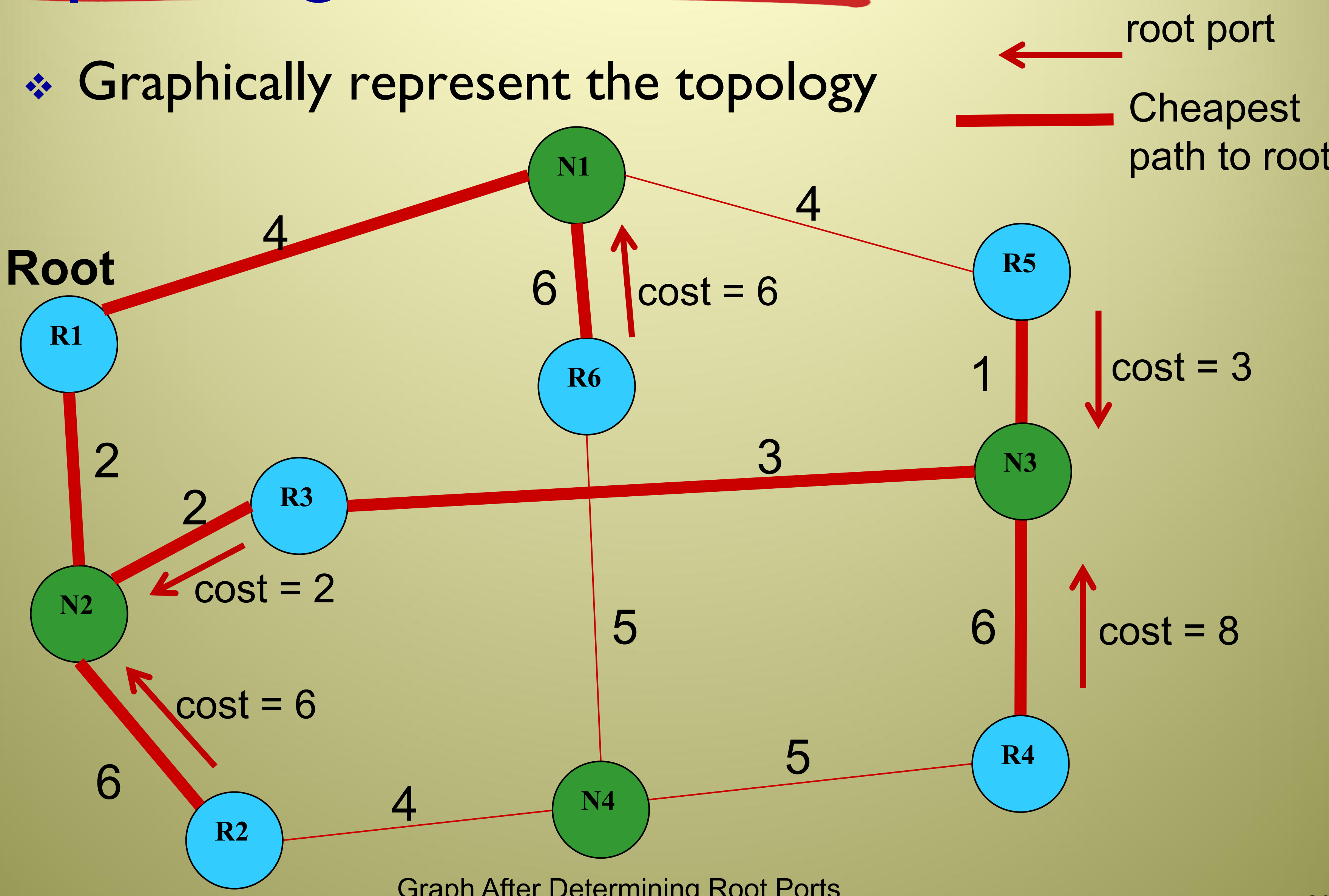
- Graphically represent the topology



Graph Representation of the Example Network

Spanning tree: creation

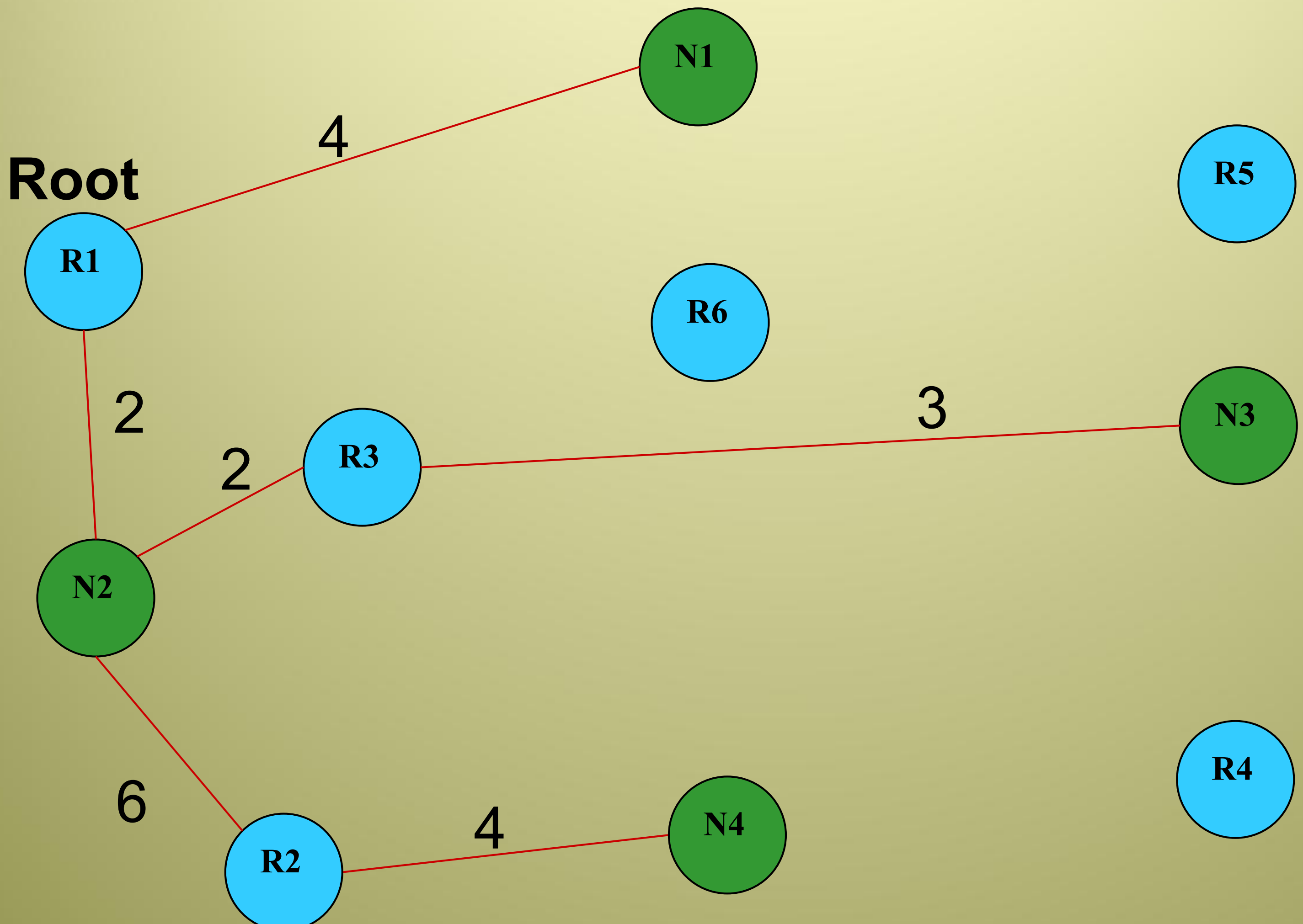
- Graphically represent the topology



Graph After Determining Root Ports

Spanning tree: creation

- Construct the spanning tree



Spanning Tree of the Example Network

Spanning tree: creation

- network can hence be setup as follows (dashed linkes represent routers not active in flooding)