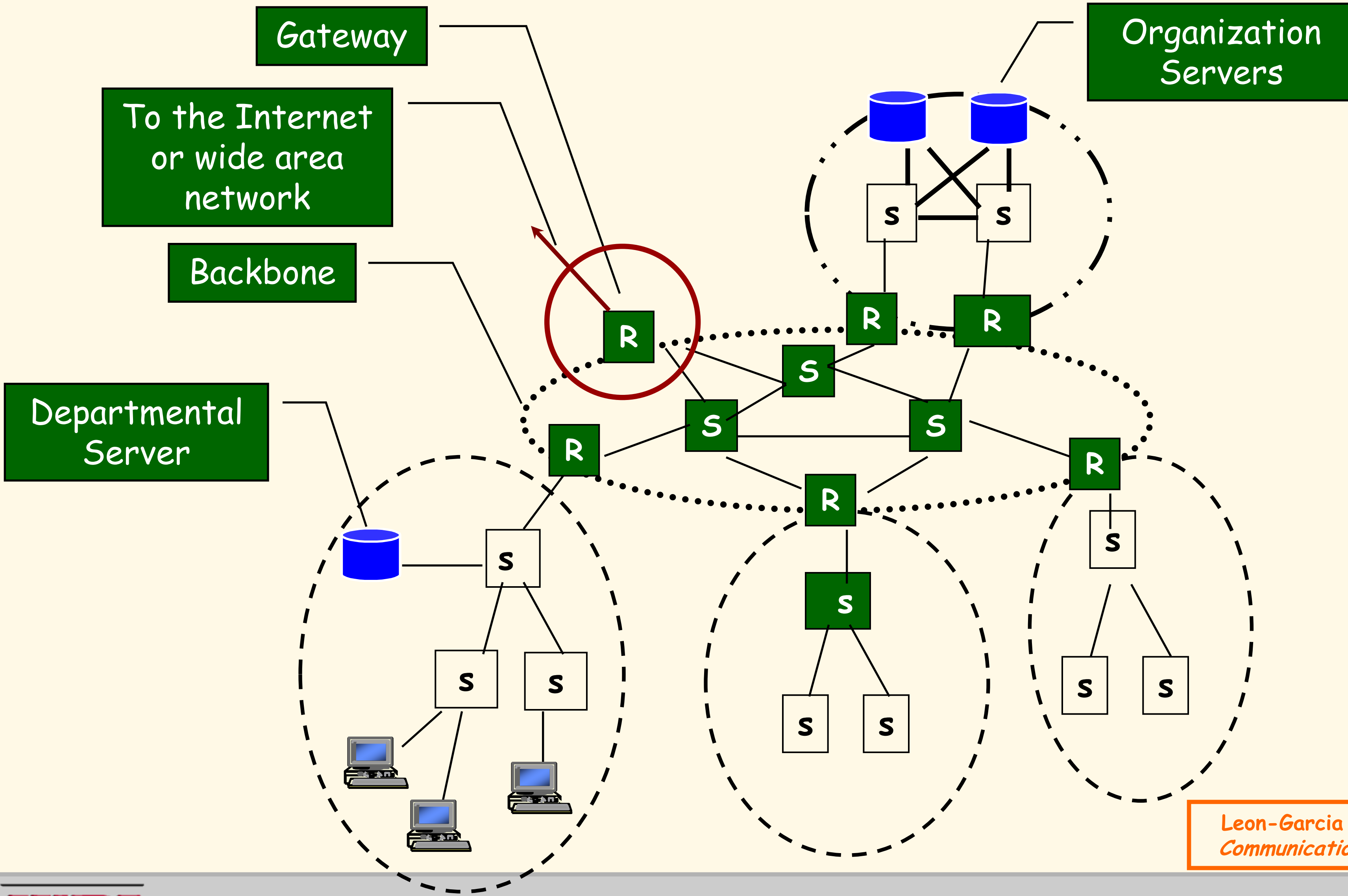
Routing Primer

Routing Outline

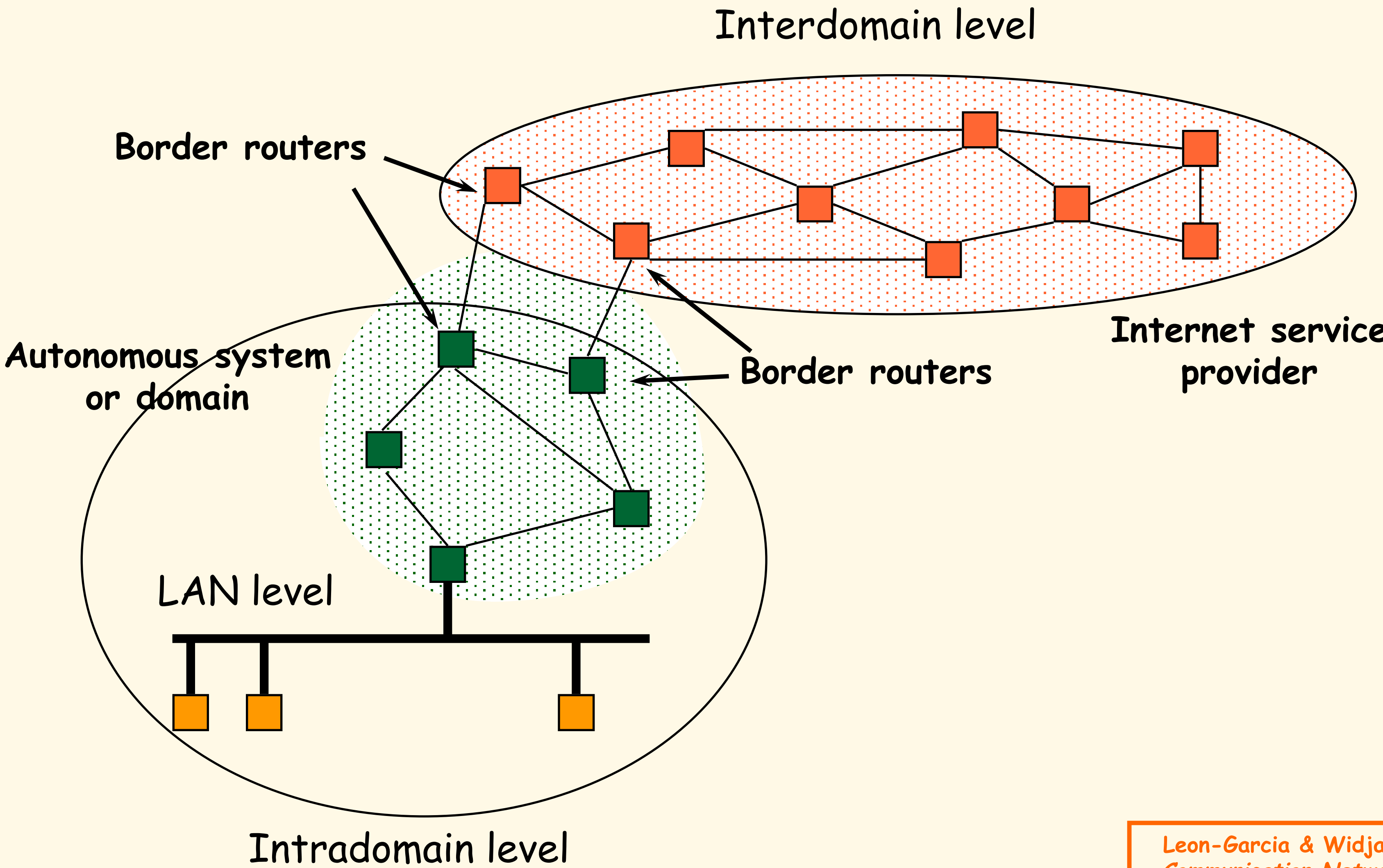
- Overview of Point-to-Point Routing (WAN)
- Routing Algorithm Classification
- Distance Vector Routing
- Link State Routing
- RIP
- OSPF
- BGP

Metropolitan Area Network (MAN)



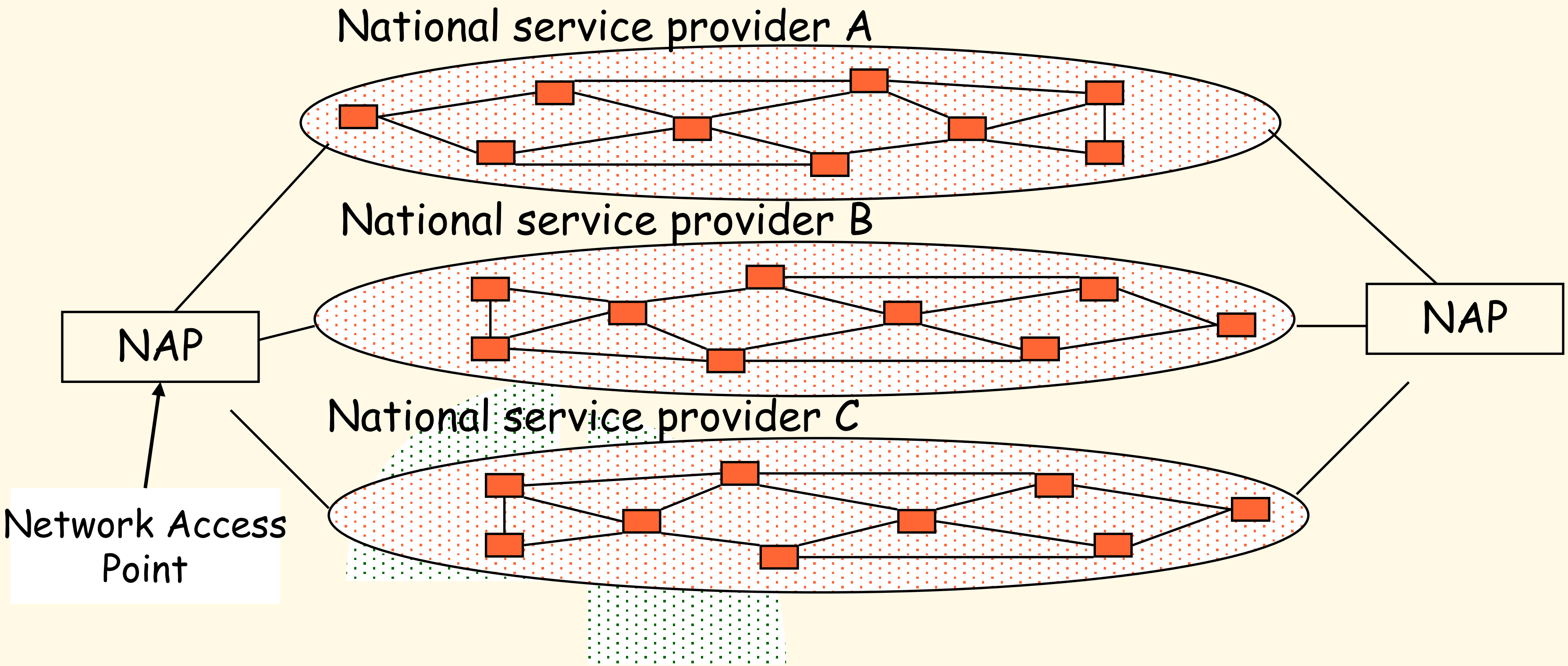
Leon-Garcia & Widjaja: *Communication Networks*

Wide Area Network (WAN)



Leon-Garcia & Widjaja:
Communication Networks

Modern Internet Backbone

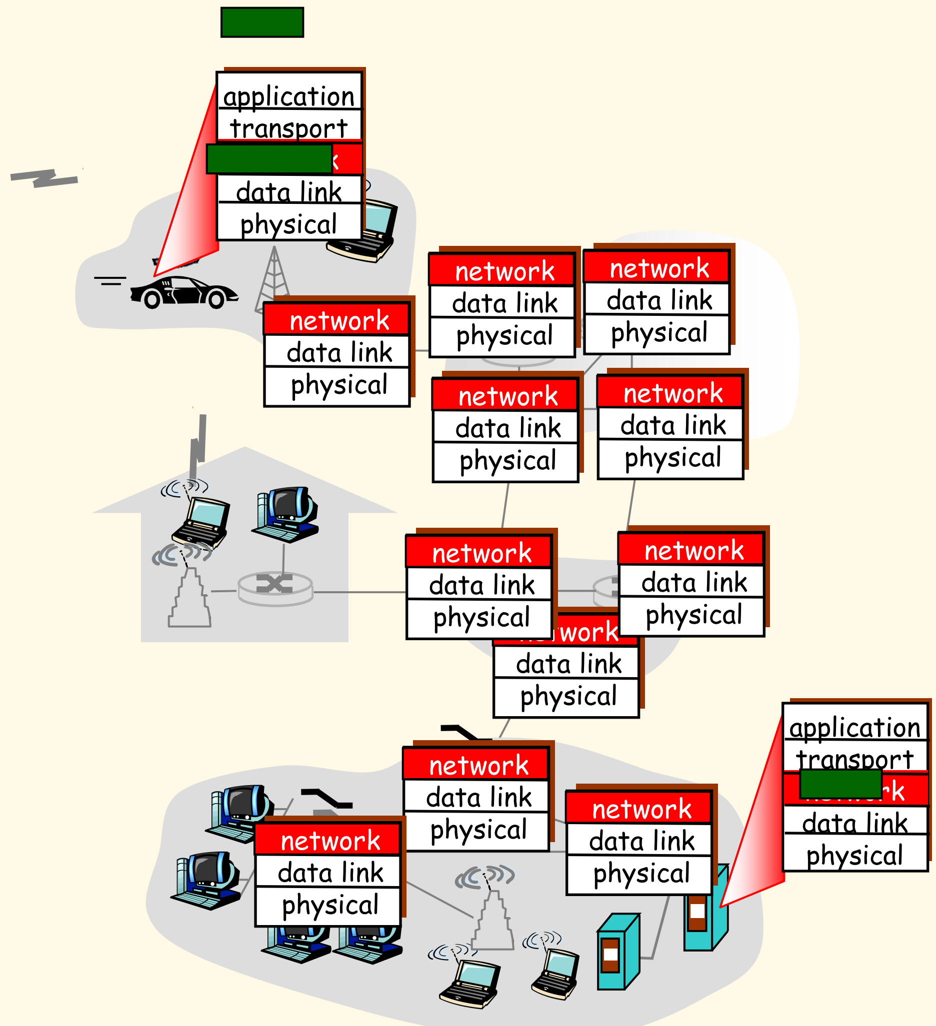


National Internet Service Providers

Leon-Garcia & Widjaja:
Communication Networks

Network Layer

- transport segment from sending to receiving host.
- on sending side, encapsulates segments into datagram packets.
- on receiving side, delivers segments to transport layer.
- network layer protocols in *every* host, router.
- router examines header fields in all IP datagrams passing through it.



K & R

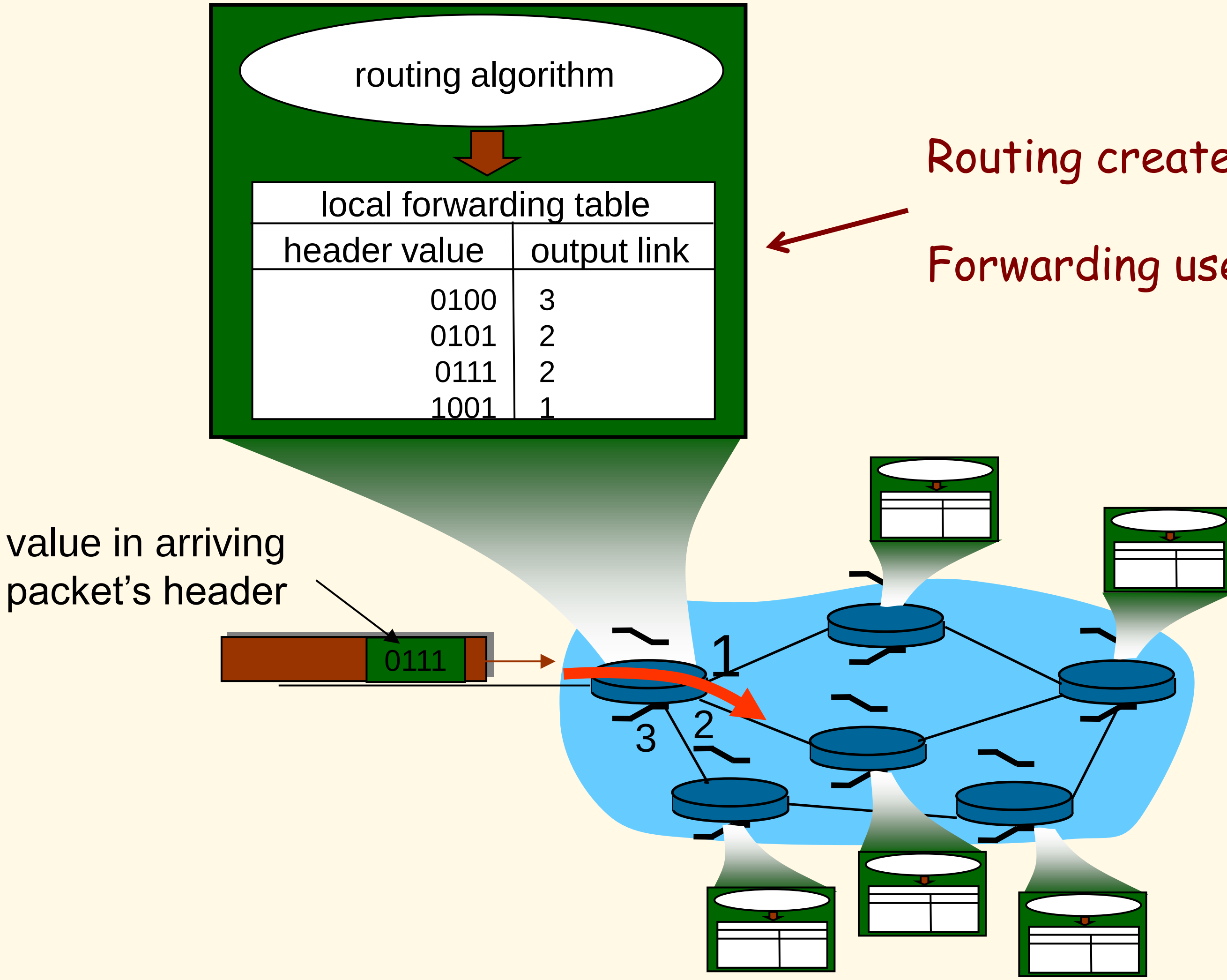
Two Key Network Layer Functions

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

analogy:

- **routing:** process of planning trip from source to destination
- **forwarding:** process of getting through single interchange

Interplay between Routing and Forwarding

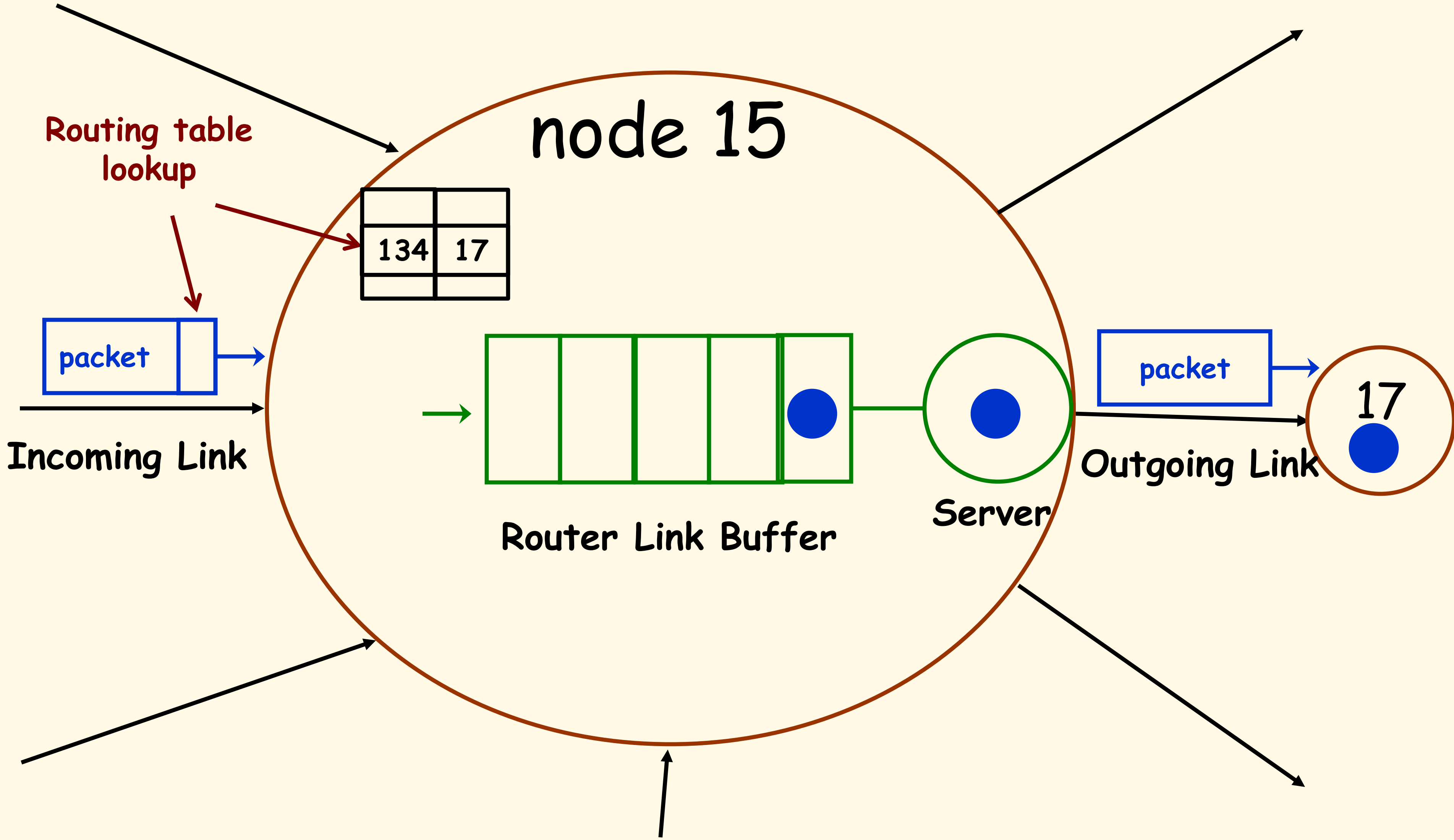


Routing creates the tables.

Forwarding uses the tables.

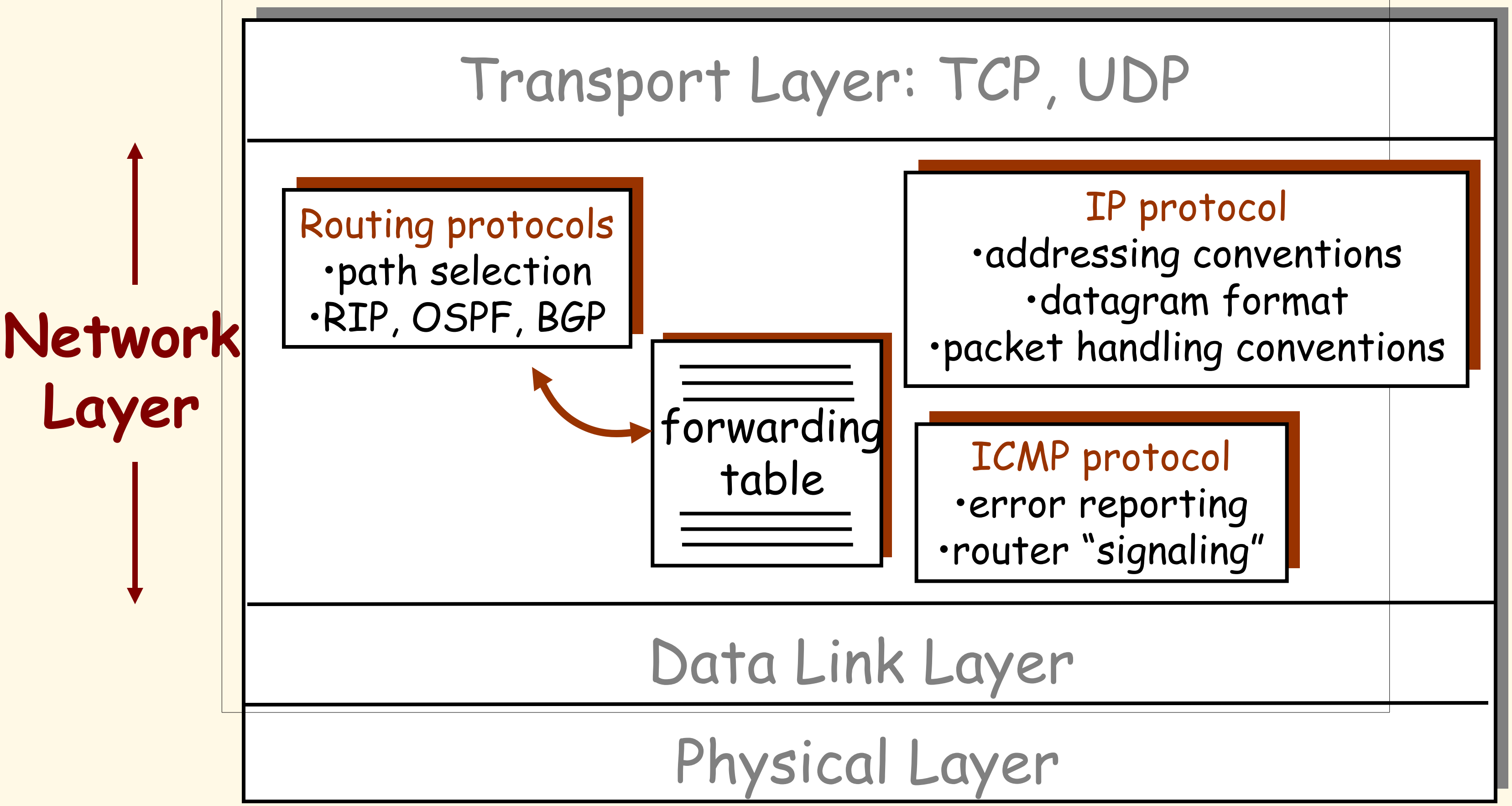
K & R

Router Node Forwarding



The Internet Network Layer

Host, router network layer functions:



Routing Algorithm Classification

Routing

Routing algorithm:: that part of the Network Layer responsible for deciding on which output line to transmit an incoming packet.

- Remember: For virtual circuit subnets the routing decision is made ONLY at set up.

Algorithm properties:: correctness, simplicity, robustness, stability, fairness, optimality, and scalability.

Routing is Graph Theory Problem

edges have costs

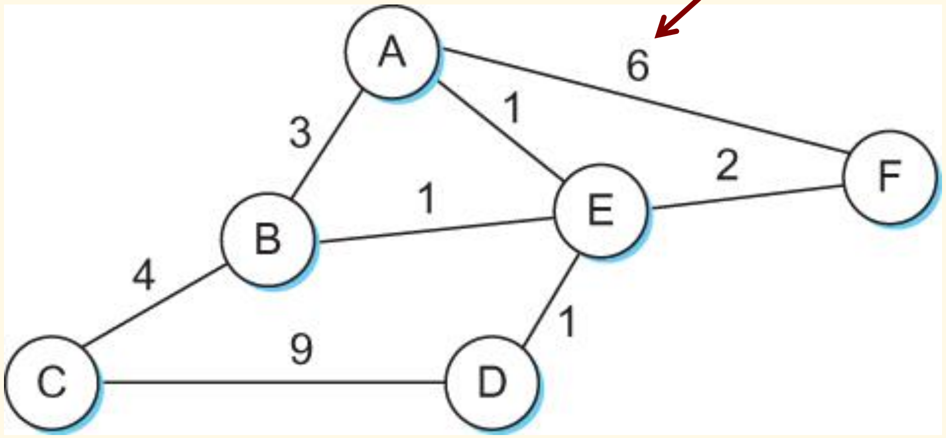


Figure 3.28 Network represented as a graph.

Routing Classification

Adaptive Routing

based on current measurements of traffic and/or topology.

1. centralized
2. isolated
3. distributed

Non-Adaptive Routing

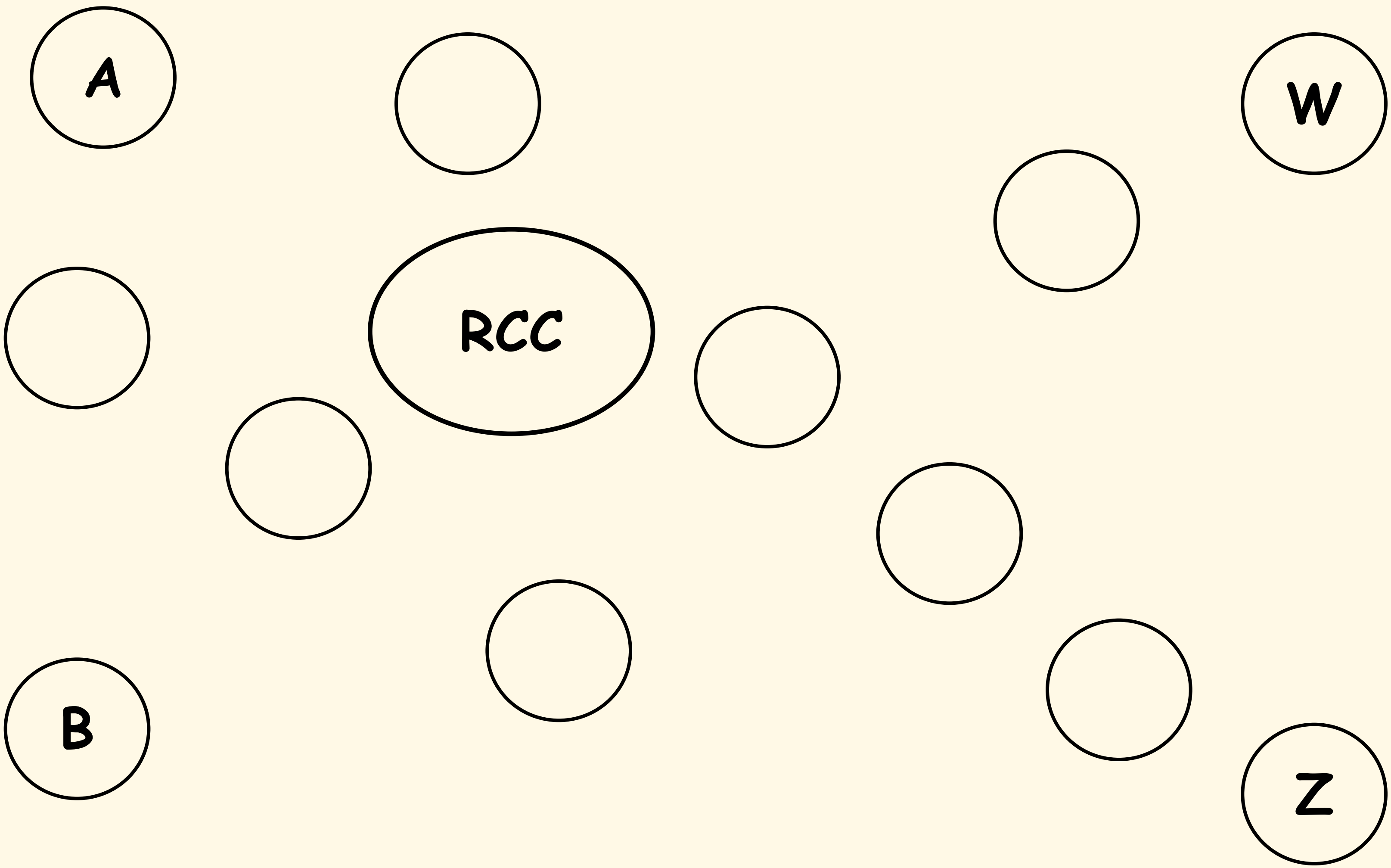
routing computed in advance and off-line

1. flooding
2. static routing using shortest path algorithms

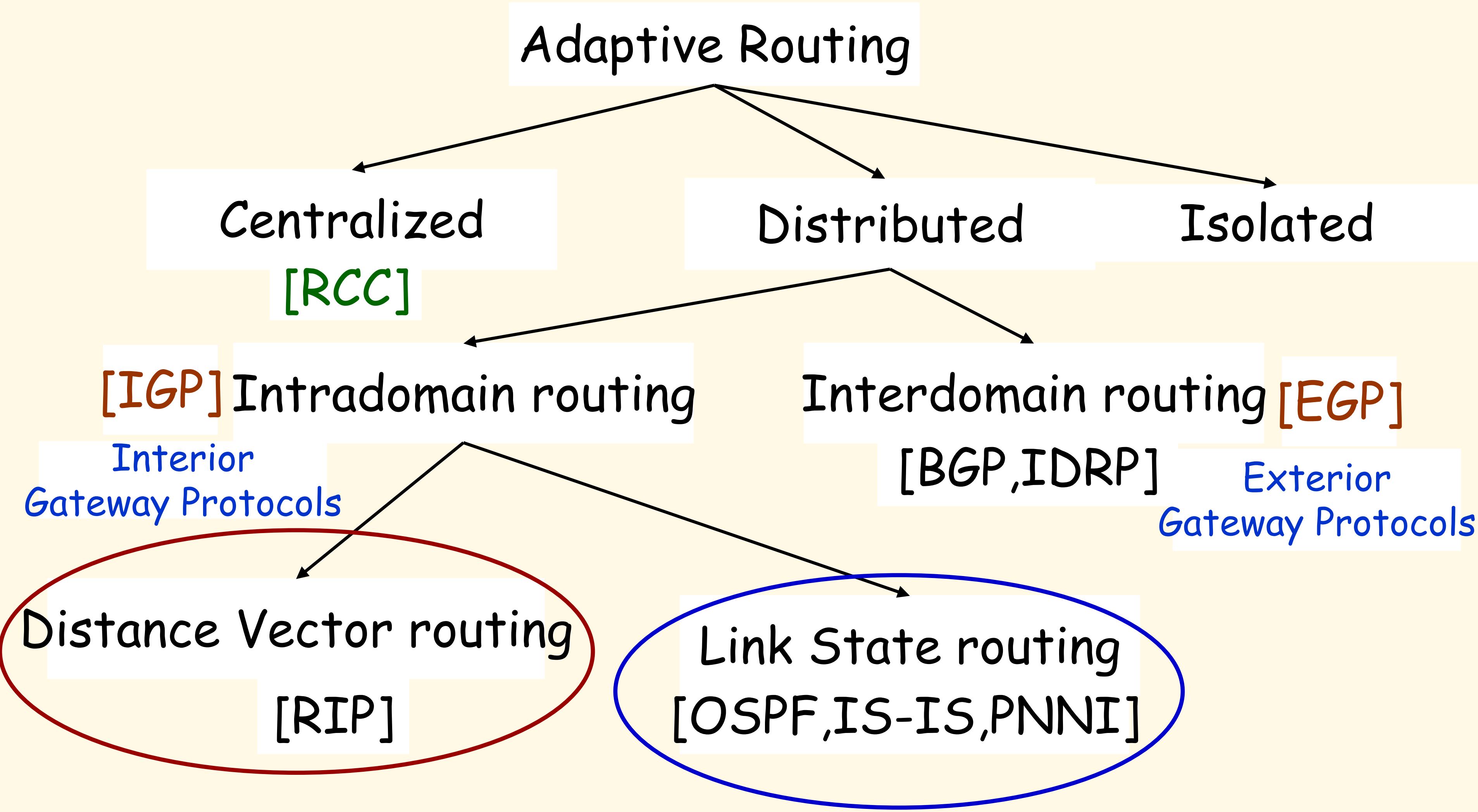
Flooding

- Pure flooding :: every incoming packet to a node is sent out on every outgoing line.
  - Obvious adjustment – do not send out on arriving link (assuming full-duplex links).
  - The routing algorithm can use a hop counter (e.g., TTL) to dampen the flooding.
  - Selective flooding :: only send on those lines going "approximately" in the right direction.

Centralized Routing



Internetwork Routing [Halsall]



Adaptive Routing Design

Design Issues:

1. How much overhead is incurred due to gathering the routing information and sending *routing packets*?
2. What is the time frame (i.e, the frequency) for sending *routing packets* in support of adaptive routing?
3. What is the complexity of the routing strategy?

Adaptive Routing

Basic functions:

1. Measurement of pertinent network data {e.g. the cost metric}.
2. Forwarding of information to where the routing computation will be done.
3. Compute the routing tables.
4. Convert the routing table information into a routing decision and then dispatch the data packet.

Shortest Path Routing

1. Bellman-Ford Algorithm [Distance Vector]
2. Dijkstra's Algorithm [Link State]

What does it mean to be the shortest (or optimal) route?

We need a cost metric (edges in graph):

a. Minimize the number of hops along the path.
b. Minimize the mean packet delay.
c. Maximize the network throughput.

Distance Vector Routing

{Tanenbaum & Perlman version}

Distance Vector Routing

- Historically known as the old ARPANET routing algorithm {or known as Bellman-Ford (BF) algorithm}.
- BF Basic idea: each router maintains a Distance Vector table containing the *distance* between itself and ALL possible destination nodes.
- Distances,based on a chosen metric, are computed using information from the neighbors' distance vectors.
- Distance Metric: usually hops or delay

Distance Vector Routing

Information kept by DV router

1. each router has an ID
2. associated with each link connected to a router, there is a link cost (static or dynamic).

Distance Vector Table Initialization

Distance to itself = 0

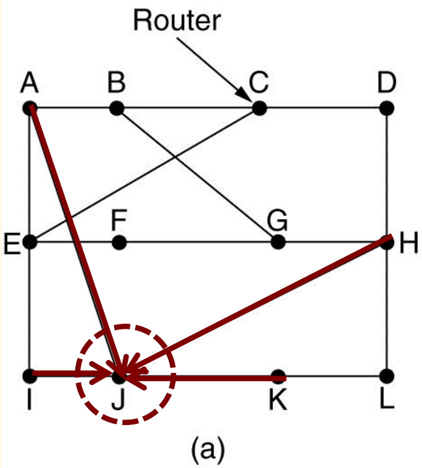
Distance to ALL other routers = infinity number

Distance Vector Algorithm [Perlman]

1. A router transmits its distance vector to each of its neighbors in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbors.
3. A router recalculates its distance vector when:
   a. It receives a distance vector from a neighbor containing different information than before.
   b. It discovers that a link to a neighbor has gone down (i.e., a topology change).

- The DV calculation is based on minimizing the cost to each destination.

Distance Vector Example



To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

(b)

Figure 5-9.(a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.

Tanenbaum

Distance Vector Routing

{Kurose & Ross version}

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Define

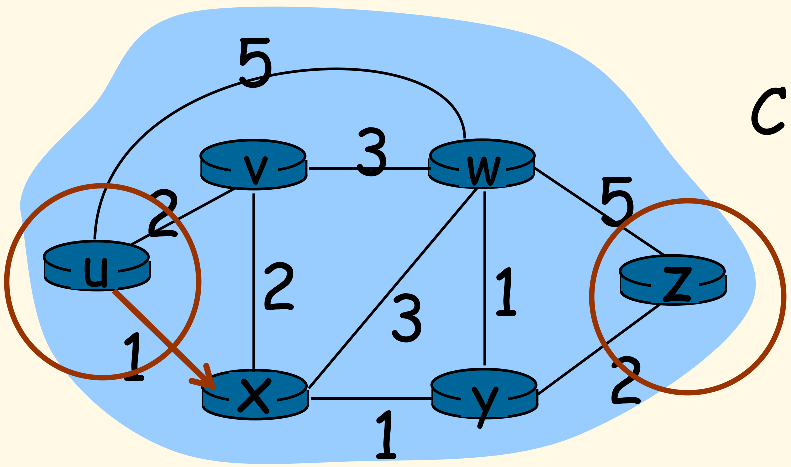
$d_x(y)$:= cost of least-cost path from x to y

Then

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

where min is taken over all neighbors v of x.

Bellman-Ford Example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

The node that achieves minimum is next hop in shortest path ➜ forwarding table.

Namely, packets from u destined for z are forwarded out link between u and x.

Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from x to y
- Node x knows cost to each neighbor v:

  $c(x,v)$

- Node x maintains distance vector

  $D_x = [D_x(y): y \in N]$

- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains

    $D_v = [D_v(y): y \in N]$

Distance Vector Algorithm

DV Basic idea:

- From time-to-time, each node sends its own distance vector estimate to neighbors.
- Asynchronous
- When a node x receives a new DV estimate from any neighbor v, it saves v's distance vector and it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under minor, natural conditions, the estimate $D_x(y)$ converges to the actual least cost $d_x(y)$.

Distance Vector Algorithm

Iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary.

Each node:

wait for (change in local link cost or msg from neighbor)

↓

recompute estimates

↓

if DV to any destination has changed, *notify* neighbors

$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$

node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

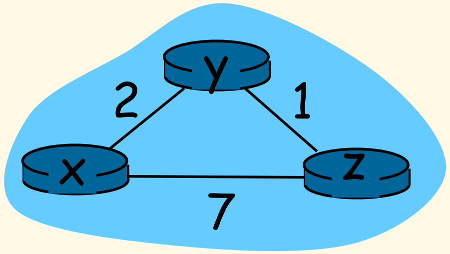
cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



time

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

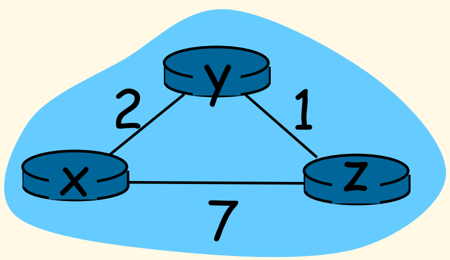
node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

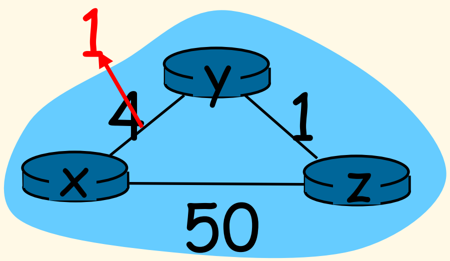
time →



Distance Vector: Link Cost Changes

Link cost changes:

- node detects local link cost change.
- updates routing info, recalculates distance vector.
- if DV changes, it notifies neighbors



"good news travels fast"

- At time t_0, y detects the link-cost change, updates its DV, and informs its neighbors.

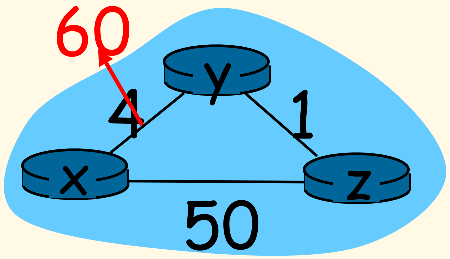
At time t_1, z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbors its DV.

At time t_2, y receives z's update and updates its distance table. y's least costs do not change and hence y does *not* send any message to z.

Distance Vector: Link Cost Changes

Link cost changes:

- good news travels fast
- bad news travels slow - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see P&D page 248!



Possible solutions:

1. Keep 'infinity ' small {depends on graph diameter}.
2. Split Horizon: node does not send those routes learned from a neighbor back to that neighbor.
3. Split Horizon with Poison Reverse:
   - If z routes through y to get to x, z tells y its (z's) distance to x is infinite (so y won't route to x via z).

- Does this solve count to infinity problem?

Link State Algorithm

1. Each router is responsible for meeting its neighbors and learning their names.
2. Each router constructs a link state packet (LSP) which consists of a list of names and cost to reach each of its neighbors.
3. The LSP is transmitted to ALL *other routers.* Each router stores the most recently generated LSP from each other router.
4. Each router uses complete information on the network topology to compute the shortest path route to each destination node.

Reliable Flooding

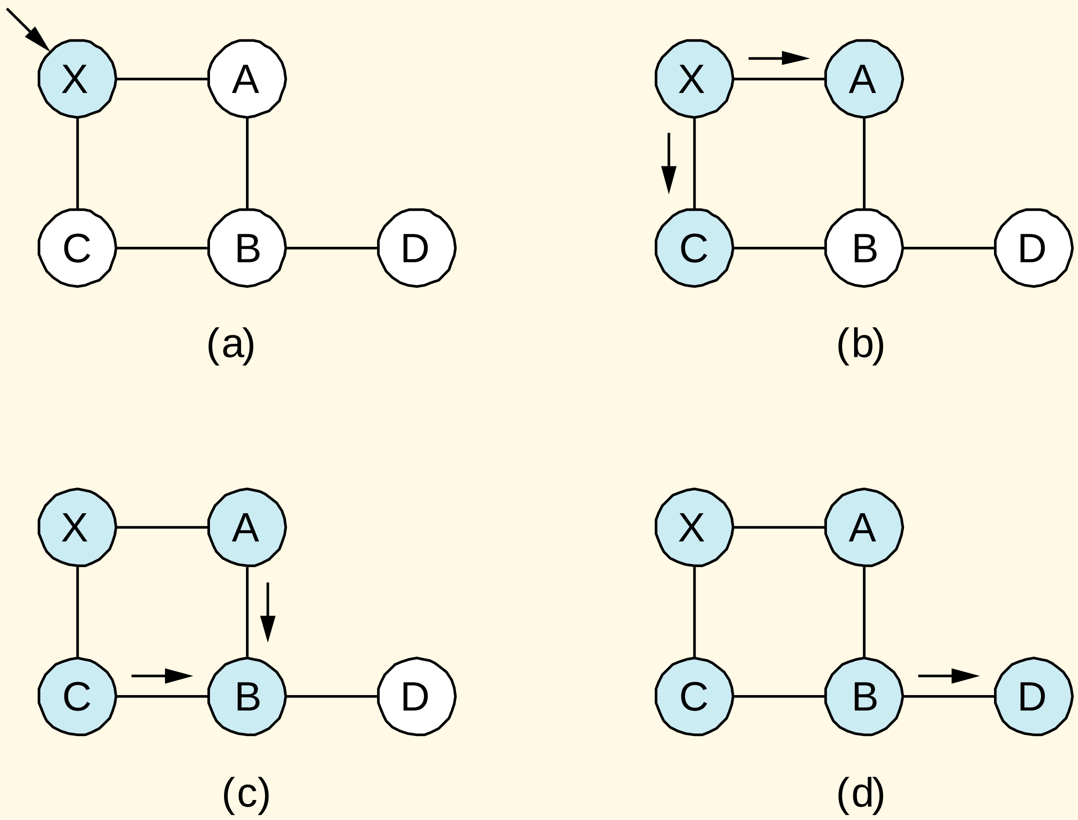


Figure 4.18 Reliable LSP Flooding

P&D slide

Reliable Flooding

- The process of making sure all the nodes participating in the routing protocol get a copy of the link-state information from all the other nodes.
- **LSP** contains:
  - Sending router's node ID
  - List of connected neighbors with the associated link cost to each neighbor
  - Sequence number
  - Time-to-live (TTL) *{an aging mechanism}*

Reliable Flooding

- First two items enable route calculation.
- Last two items make process reliable
  - ACKs and checking for duplicates is needed.
- Periodic **Hello** packets used to determine the demise of a neighbor.
- The sequence numbers are not expected to wrap around.
  - →this field needs to be large (64 bits) !!

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 destination v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path is definitively known.

K & R

Dijsktra's Shortest Path Algorithm

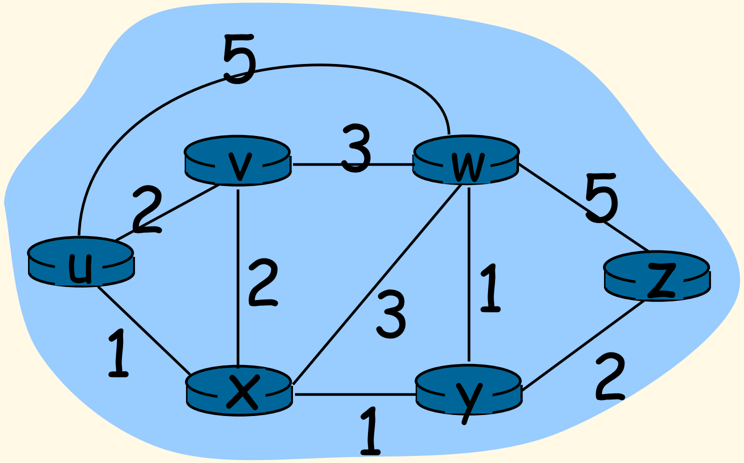
```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

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'
```

[K&R]

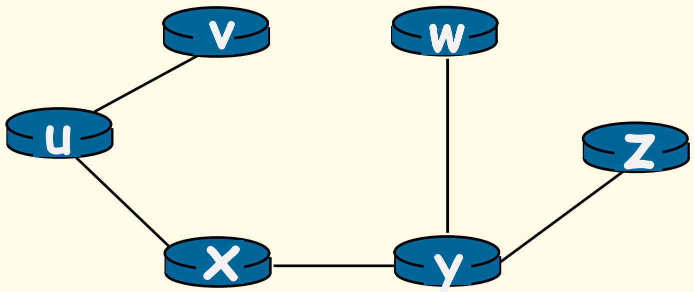
Dijkstra's Algorithm: Example

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					



Dijkstra's Algorithm: Example (2)

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)

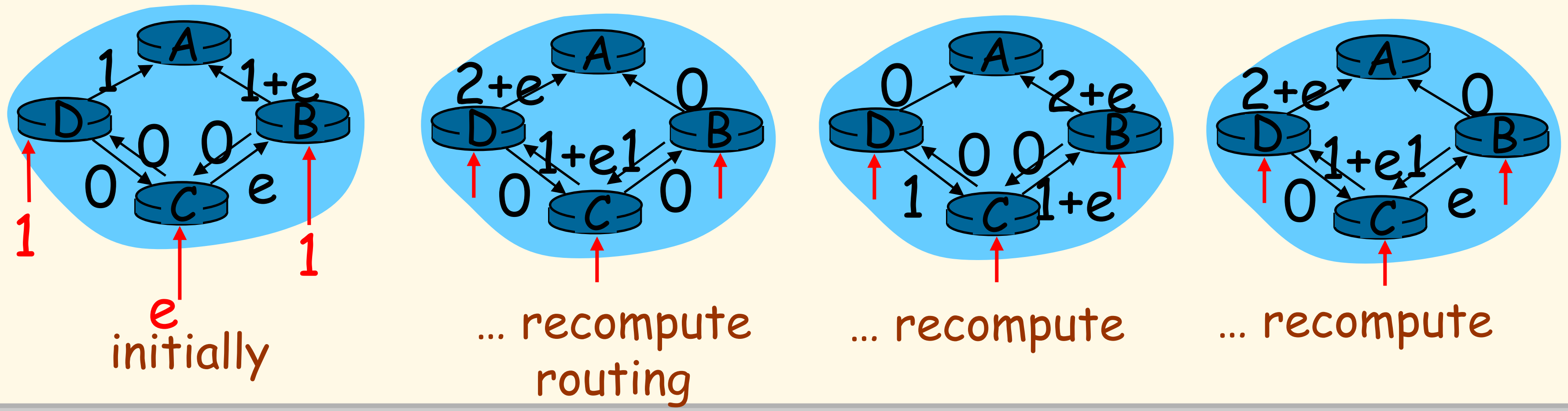
Dijkstra's Algorithm, Discussion

Algorithm complexity: n nodes

- each iteration: need to check all nodes, w, not in N
- n(n+1)/2 comparisons: $O(n^2)$
- more efficient implementations possible: O(nlogn)

Oscillations possible:

- e.g., link cost = amount of carried traffic



Intra-AS Routing

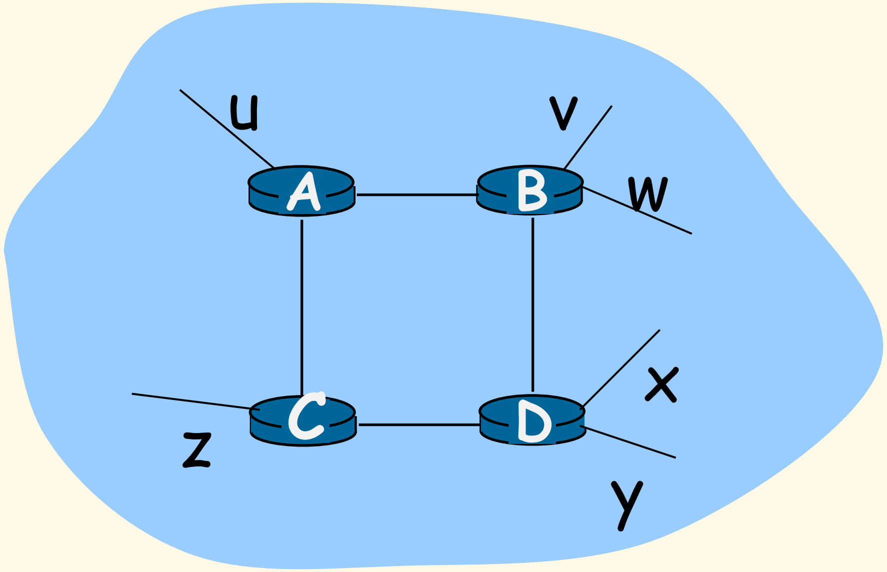
- also known as Interior Gateway Protocols (IGP)
- most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

Routing Information Protocol (RIP)

- RIP had widespread use because it was distributed with BSD Unix in "routed", a router management daemon in 1982.
- RIP - most used Distance Vector protocol.
- RFC1058 in June 1988
- Runs over UDP.
- Metric = hop count
- BIG problem is max. hop count =16

  ➔ RIP limited to running on small networks (or AS's that have a small diameter)!!

Routing Information Protocol (RIP)



From router A to subnets:

destination	hops
u	1
v	2
w	2
x	3
y	3
z	2

- Sends DV packets every 30 seconds (or faster) as Response Messages (also called advertisements).
- each advertisement: list of up to 25 destination subnets within AS.
- Upgraded to RIPv2

RIP Packets

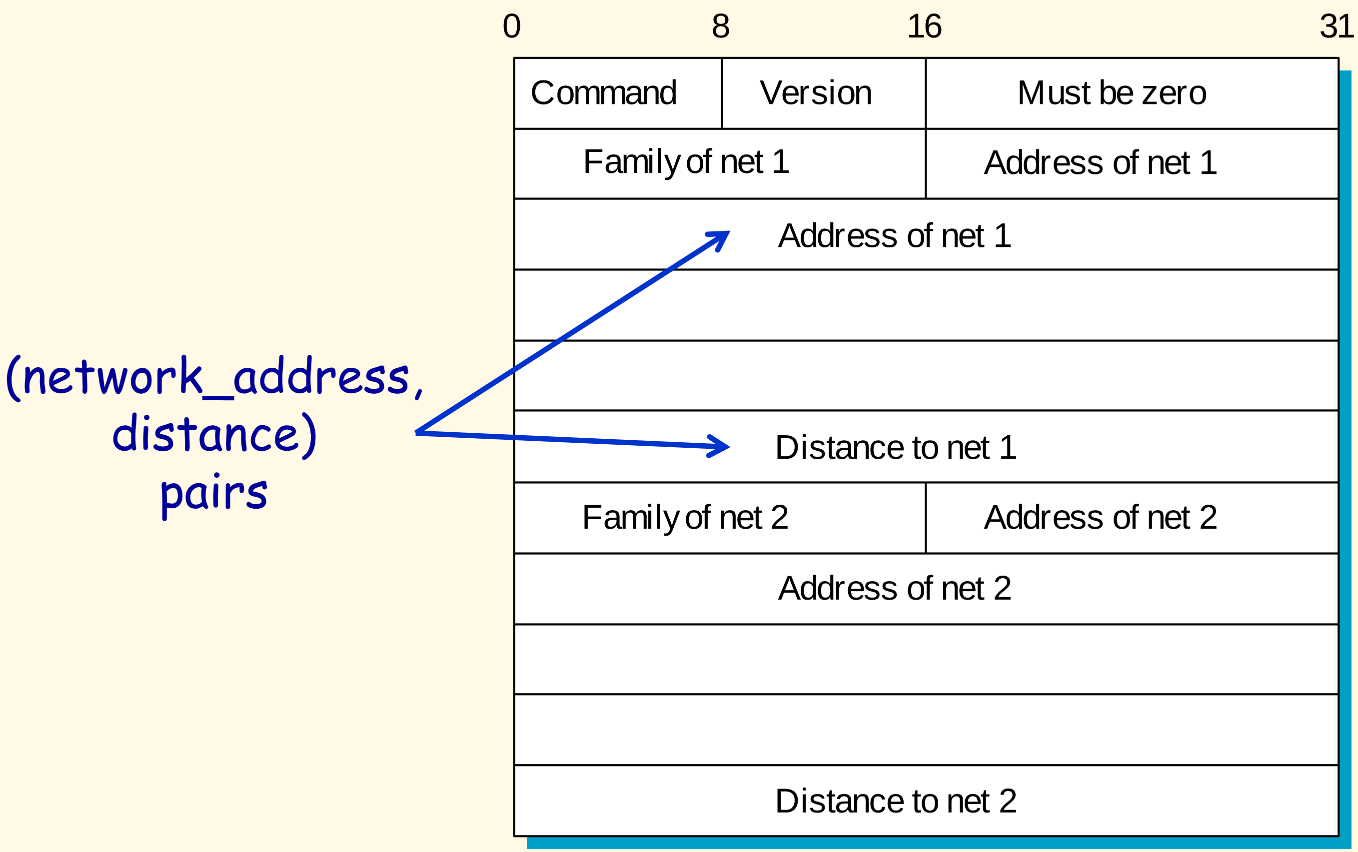


Figure 4.17 RIP Packet Format

P&D slide

RIPv2

- Allows routing on a subnet (subnet masks)
- Has an authentication mechanism
- Tries to deal with multicast
- Uses route tags
- Has the ability for router to announce routes on behalf of another router.

RIPv2 Packets

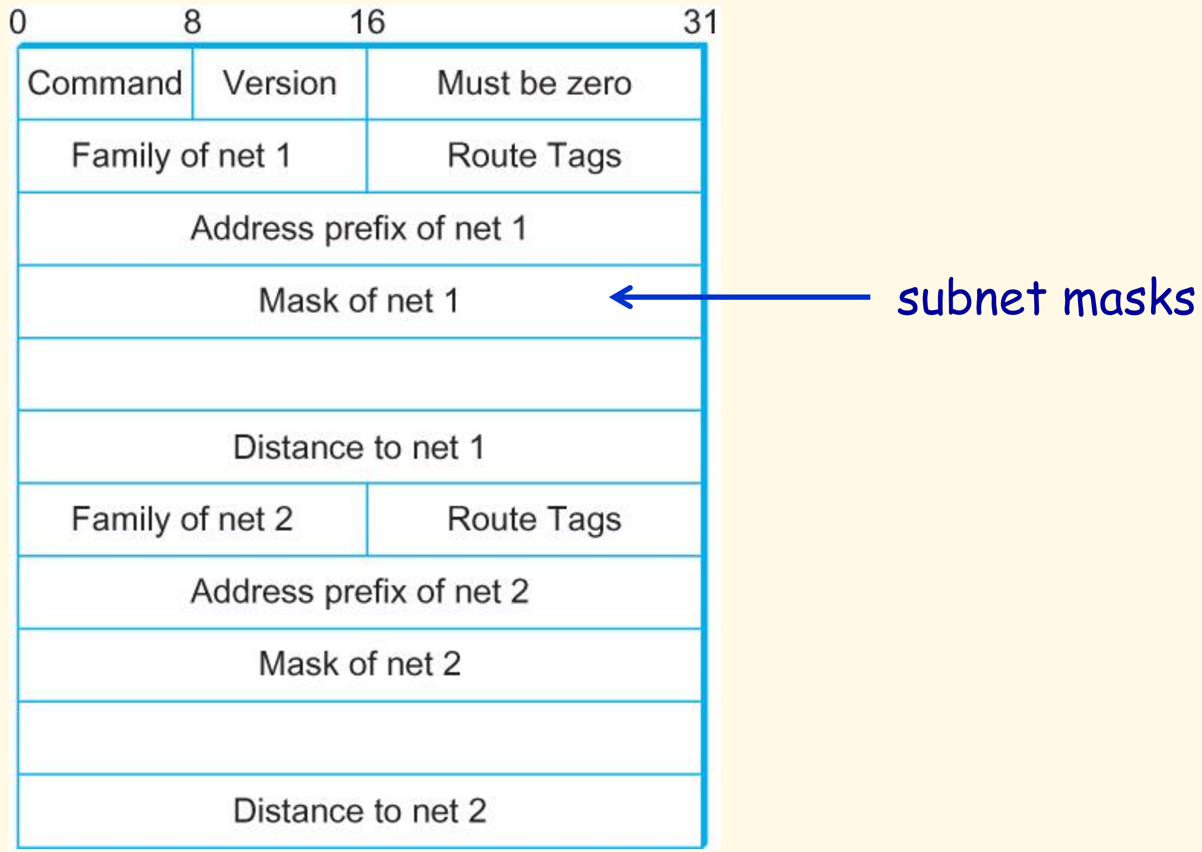


Figure 3.31 RIPv2 Packet Format

OSPF (Open Shortest Path First)

- "open" :: publicly available (due to IETF)
- uses Link State algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation uses Dijkstra's algorithm.
- OSPF advertisement carries one entry per neighbor router.
- advertisements disseminated to entire AS (via flooding*).
  - carried in OSPF messages directly over IP (rather than TCP or UDP).

* However hierarchy (partitioning domains into areas) reduces flooding impact.

OSPF "Advanced" Features (not in RIP)

- security: all OSPF messages authenticated (to prevent malicious intrusion).
- multiple same-cost paths allowed (only one path in RIP).
- For each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort; high for real time).
- integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF.
- hierarchical OSPF used in large domains.

Partitioning Domains

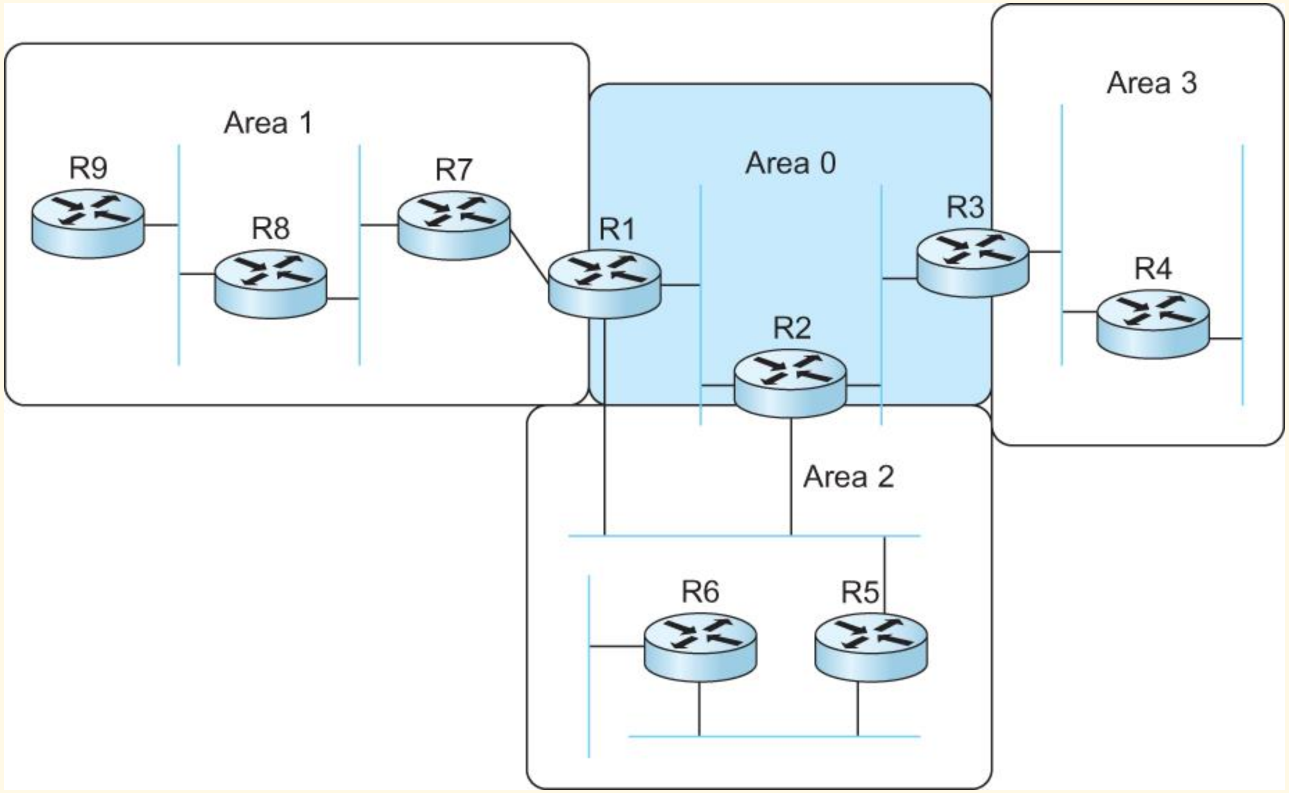
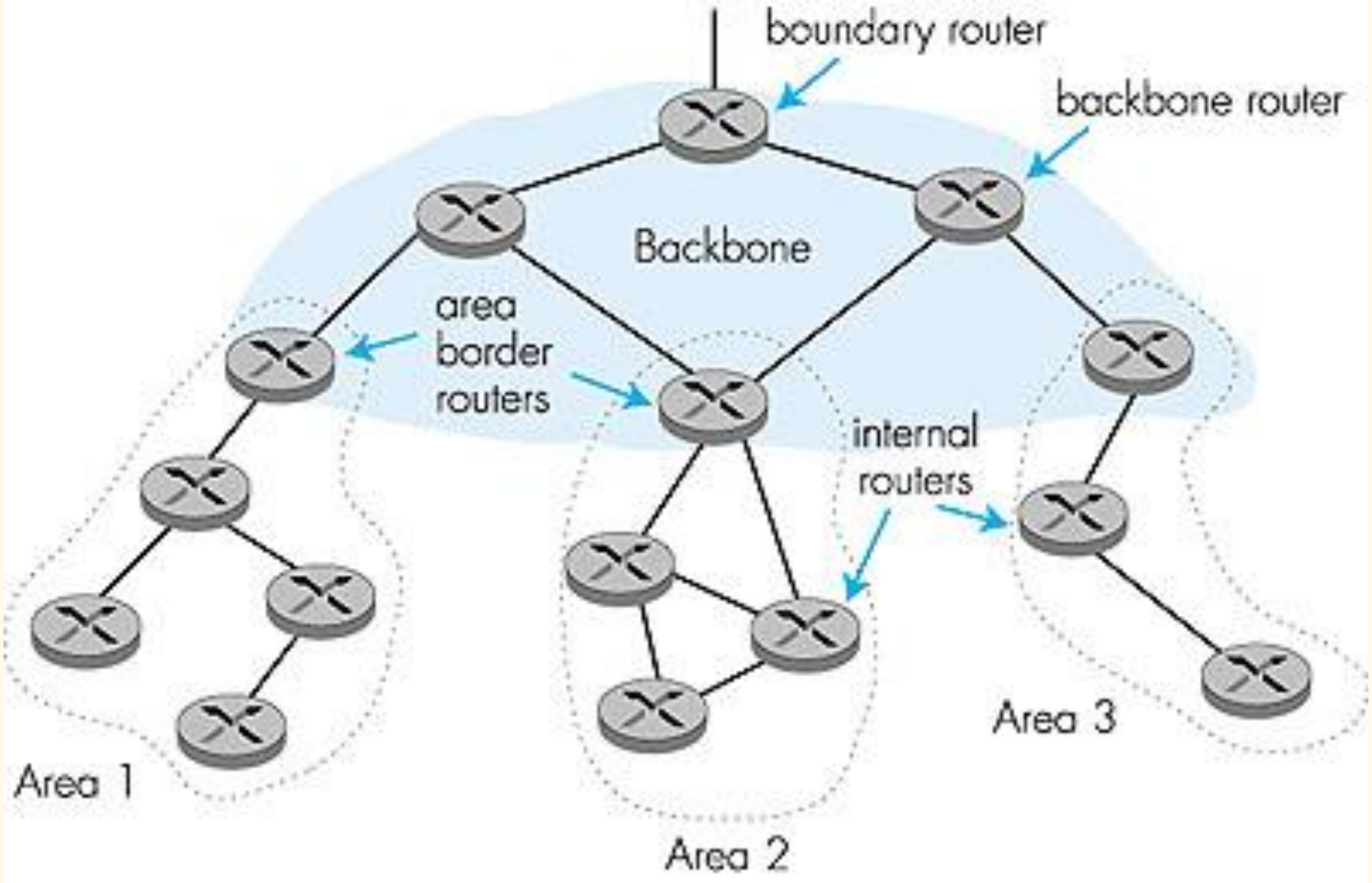


Figure 4.2 A domain divided into areas

Hierarchical OSPF



Hierarchical OSPF

- **Two-Level Hierarchy:** local area, backbone.
  - Link-State Advertisements (LSAs) only in area
  - each node has detailed area topology; only knows direction (shortest path) to nets in other areas.
- ***area border routers:*** "summarize" distances to nets in own area, advertise to other Area Border routers.
- ***backbone routers:*** run OSPF routing limited to backbone.
- ***boundary routers:*** connect to other AS's.

OSPF LSA Types

1. Router link advertisement [Hello message]
2. Network link advertisement
3. Network summary link advertisement
4. AS border router's summary link advertisement
5. AS external link advertisement

OSPF

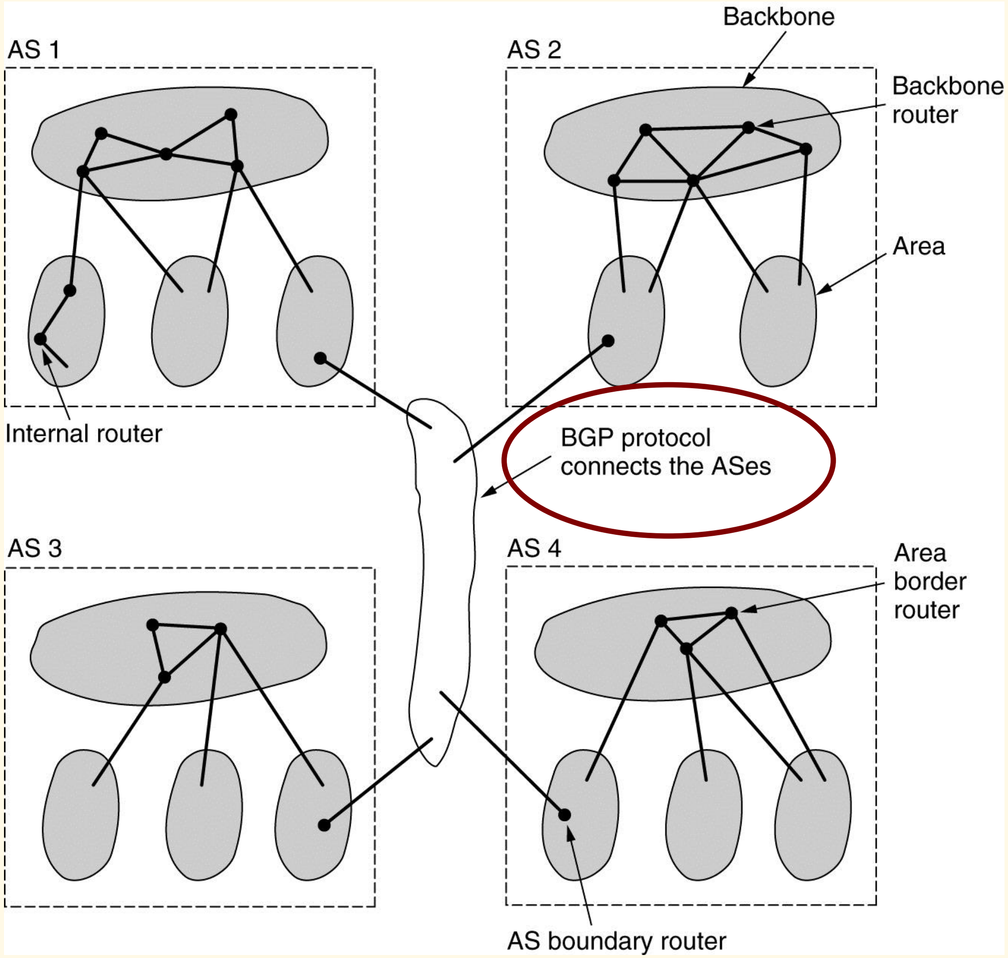


Figure 5-65. The relation between AS's, backbones, and areas in OSPF

Tanenbaum

Internet Inter-AS routing: BGP

- BGP (Border Gateway Protocol): the de facto standard
- BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighboring ASs.
  2. Propagate reachability information to all AS-internal routers.
  3. Determine "good" routes to subnets based on reachability information and policy.
- allows subnet to advertise its existence to rest of Internet: "I am here!"

Routing Primer Summary

- Routers forward and route over WANs
  - Produce look up tables in routers
- Routing Classification:
  - Adaptive or non-adaptive
  - Interdomain and Intradomain
- Distance Vector Routing (DV)
  - Perlman version
  - Tanenbaum example
  - K&R version

Routing Primer Summary

- Link State Routing (LS)
  - Uses reliable flooding; Dijkstra's SP algorithm
- RIP
  - Old ARPA routing; unicast DV routing
- OSPF
  - Two-Level Hierarchical LS routing
  - Five LSA types for router communication
- BGP
  - Interdomain routing using reachability