Distance Vector Routing

Computer Networks B10

DV Routing Outline

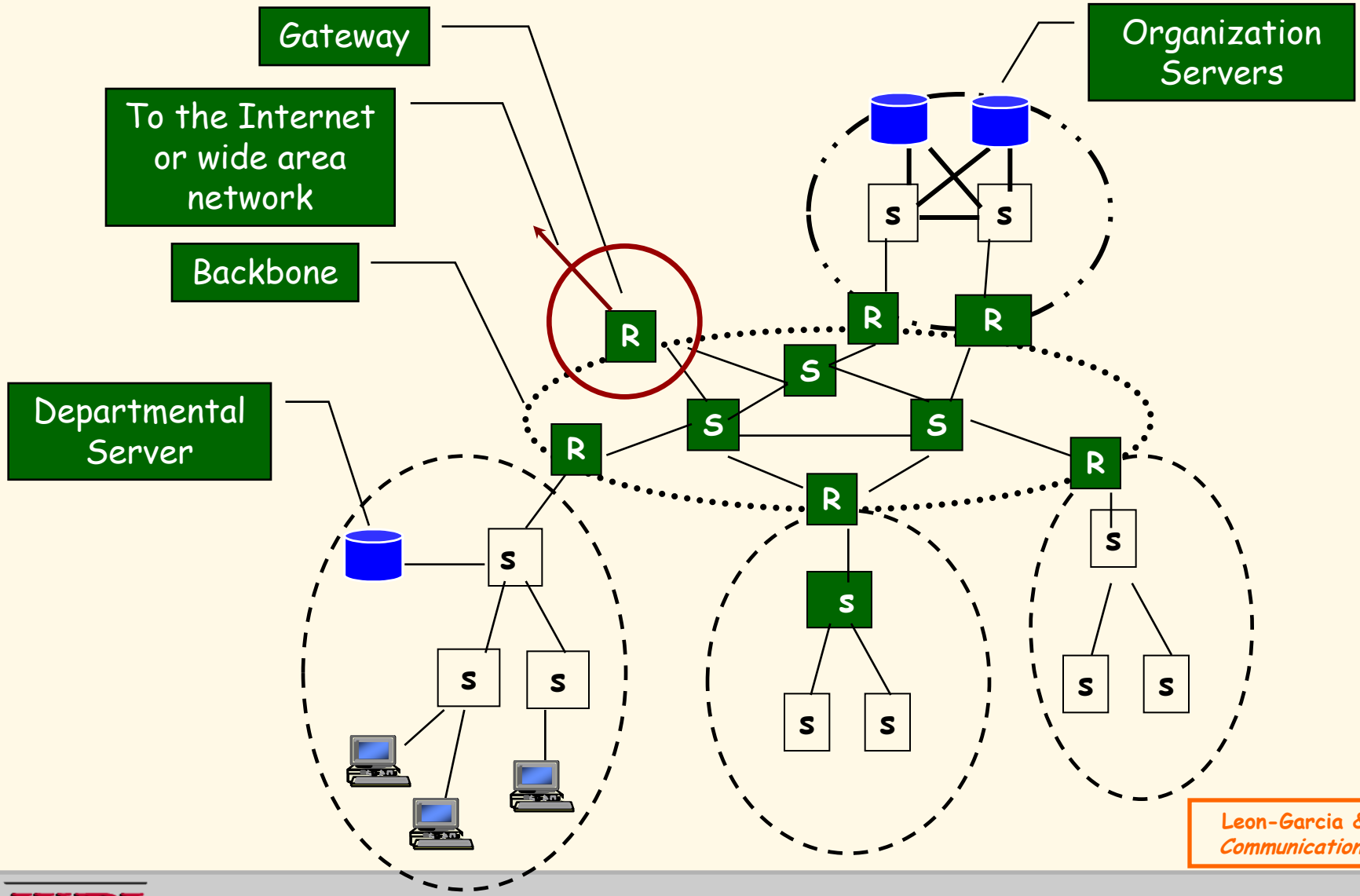
- Internet Context
- Network Layer Routing (**K&R slides)
- Quick Routing Overview
- Distance Vector Routing (my version)
  - Adapted from Tanenbaum & Perlman Texts
- Distance Vector Routing (K&R version)
- Summary

Internet Context

Computer Networks

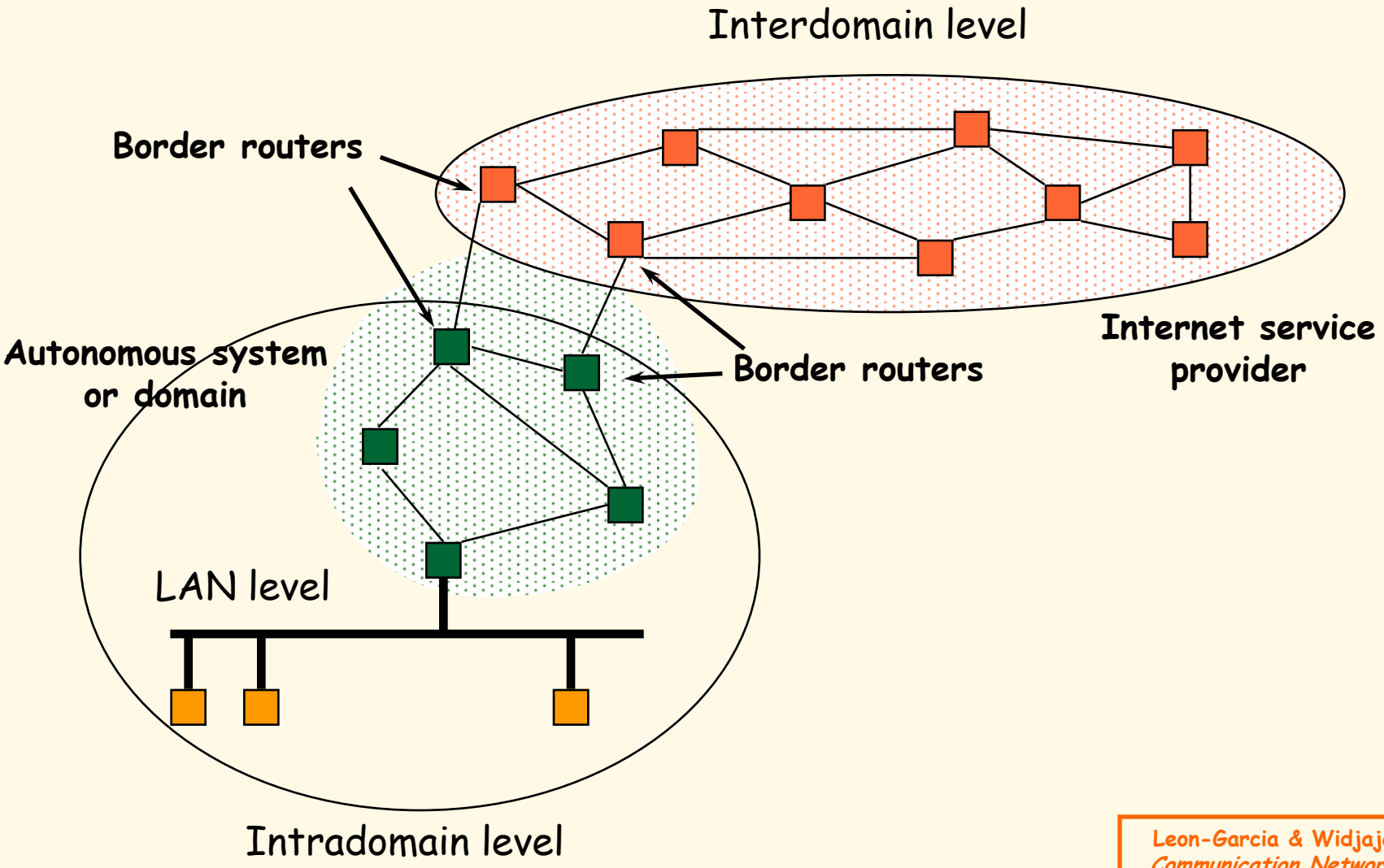
Distance Vector Routing

Metropolitan Area Network (MAN)



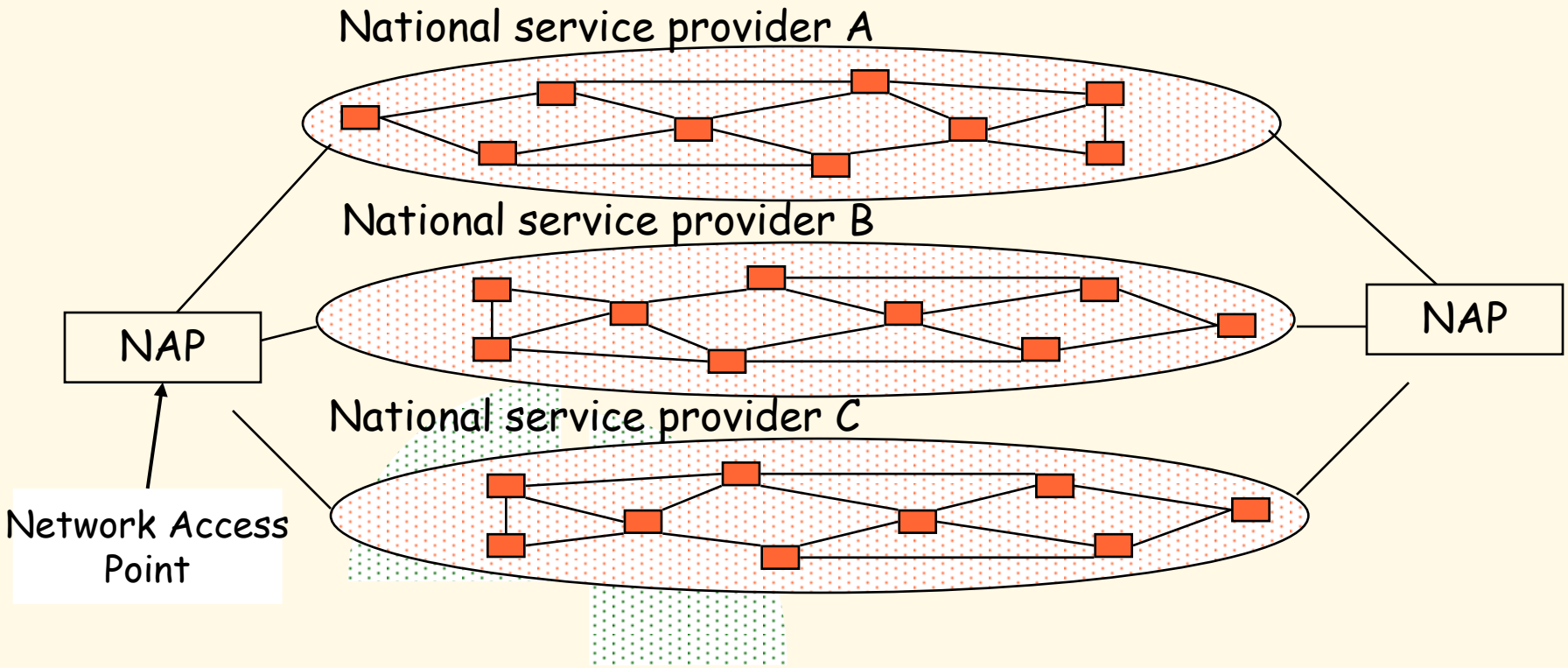
Leon-Garcia & Widjaja: *Communication Networks*

Wide Area Network (WAN)



Leon-Garcia & Widjaja:
Communication Networks

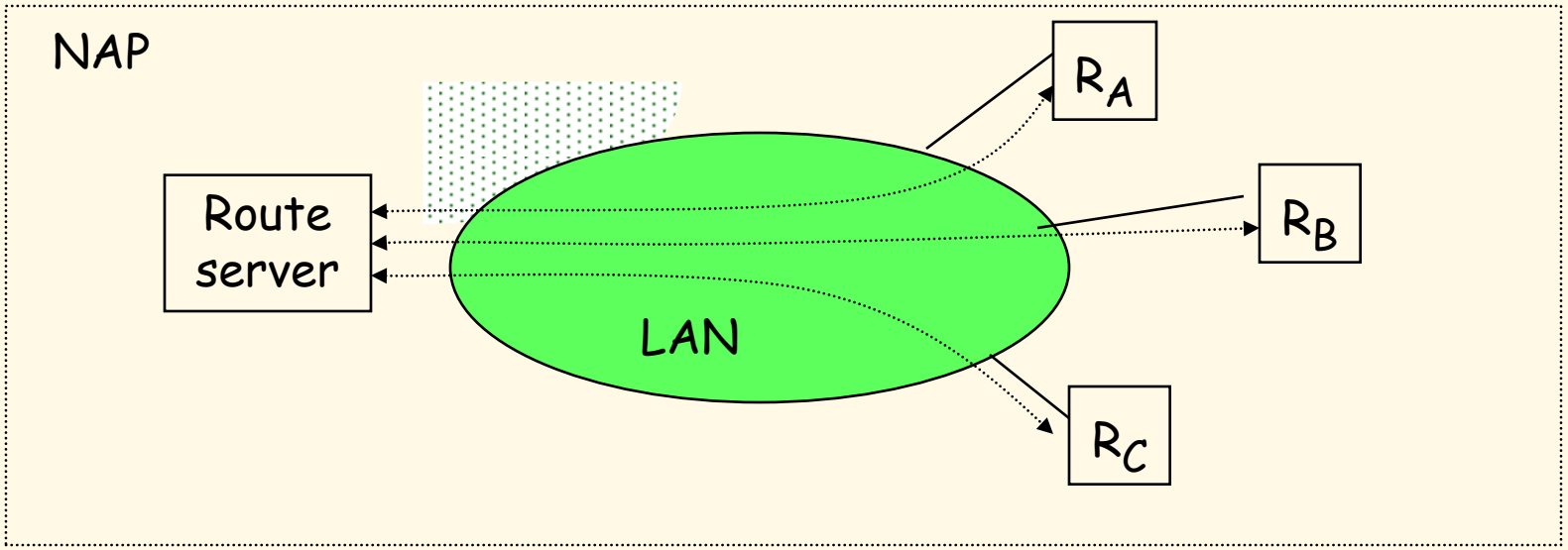
Modern Internet Backbone



National Internet Service Providers

Leon-Garcia & Widjaja: *Communication Networks*

Network Access Point



Leon-Garcia & Widjaja: *Communication Networks*

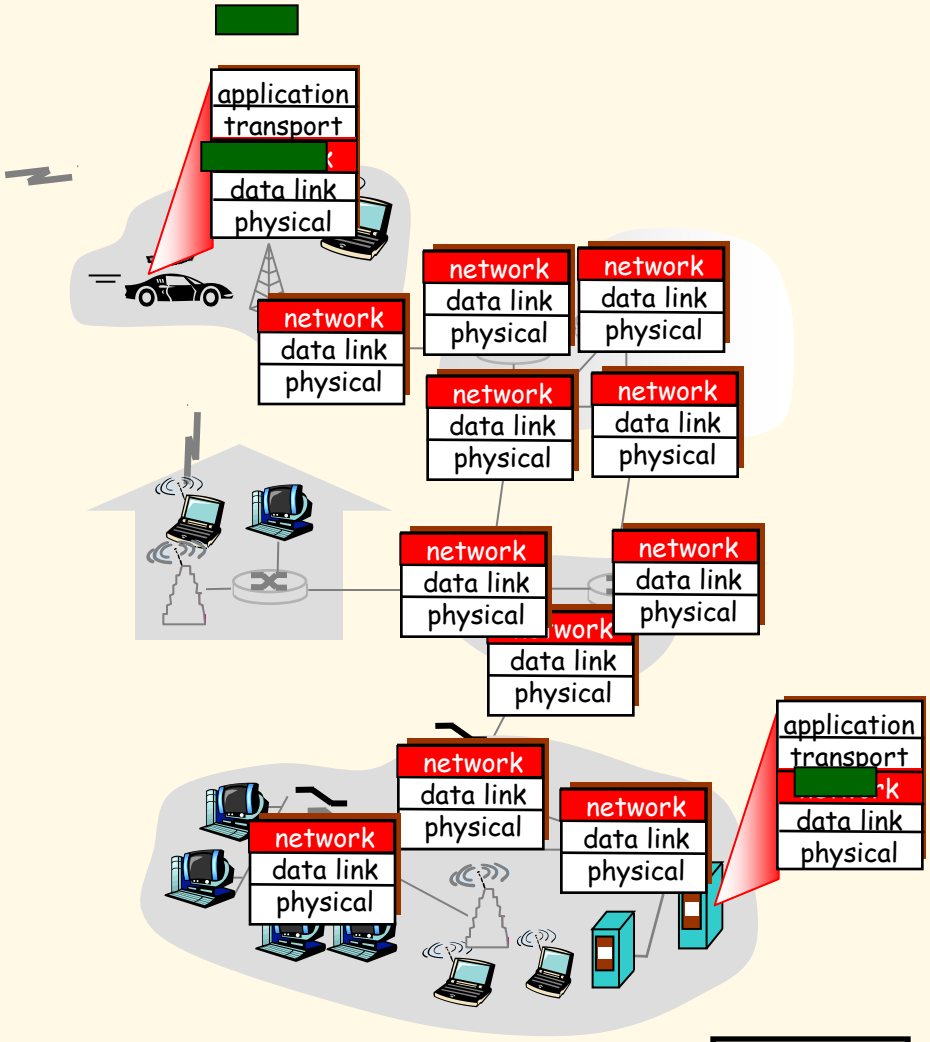
Network Layer Routing

Computer Networks

Distance Vector Routing

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

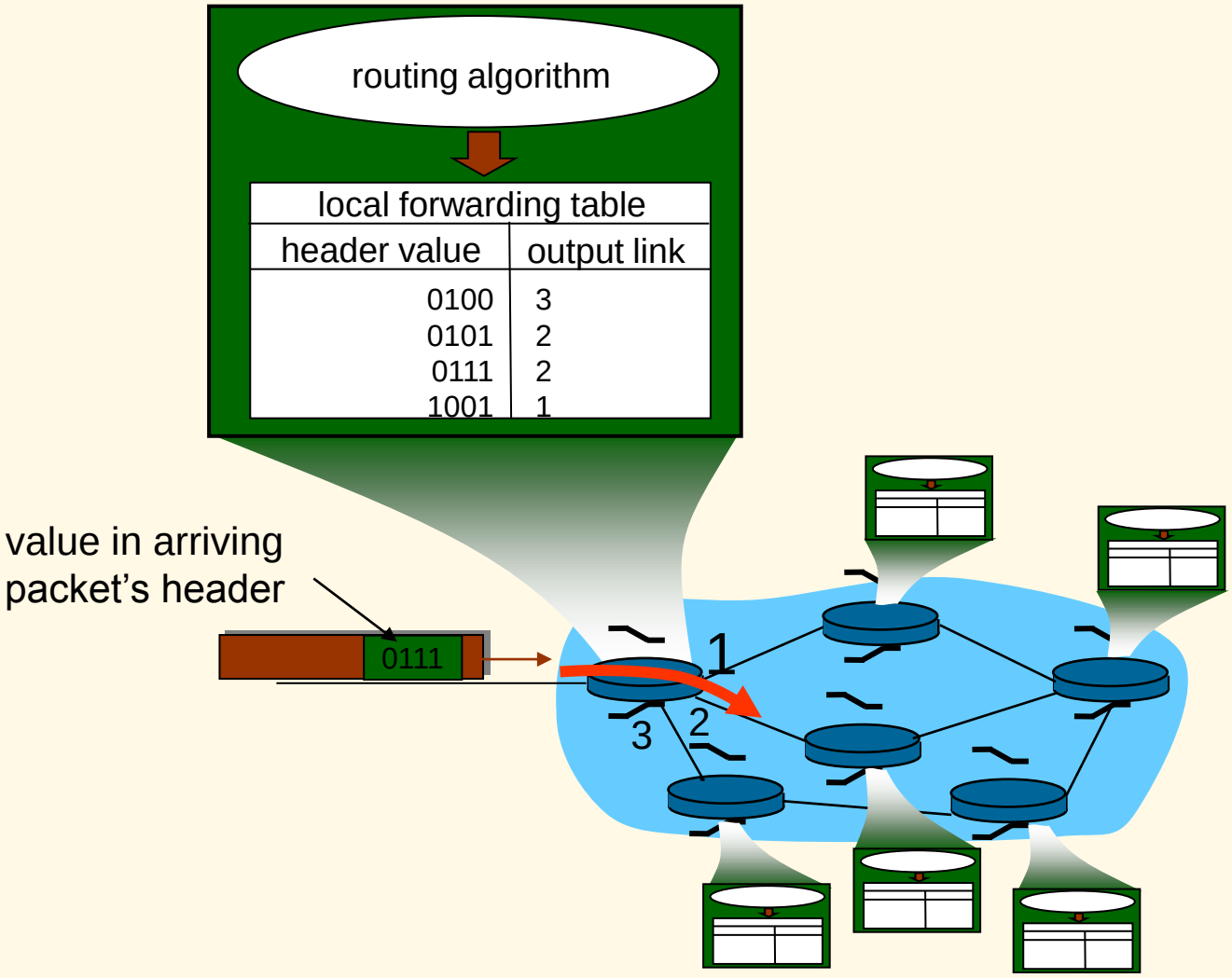
Two Key Network Layer Functions

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

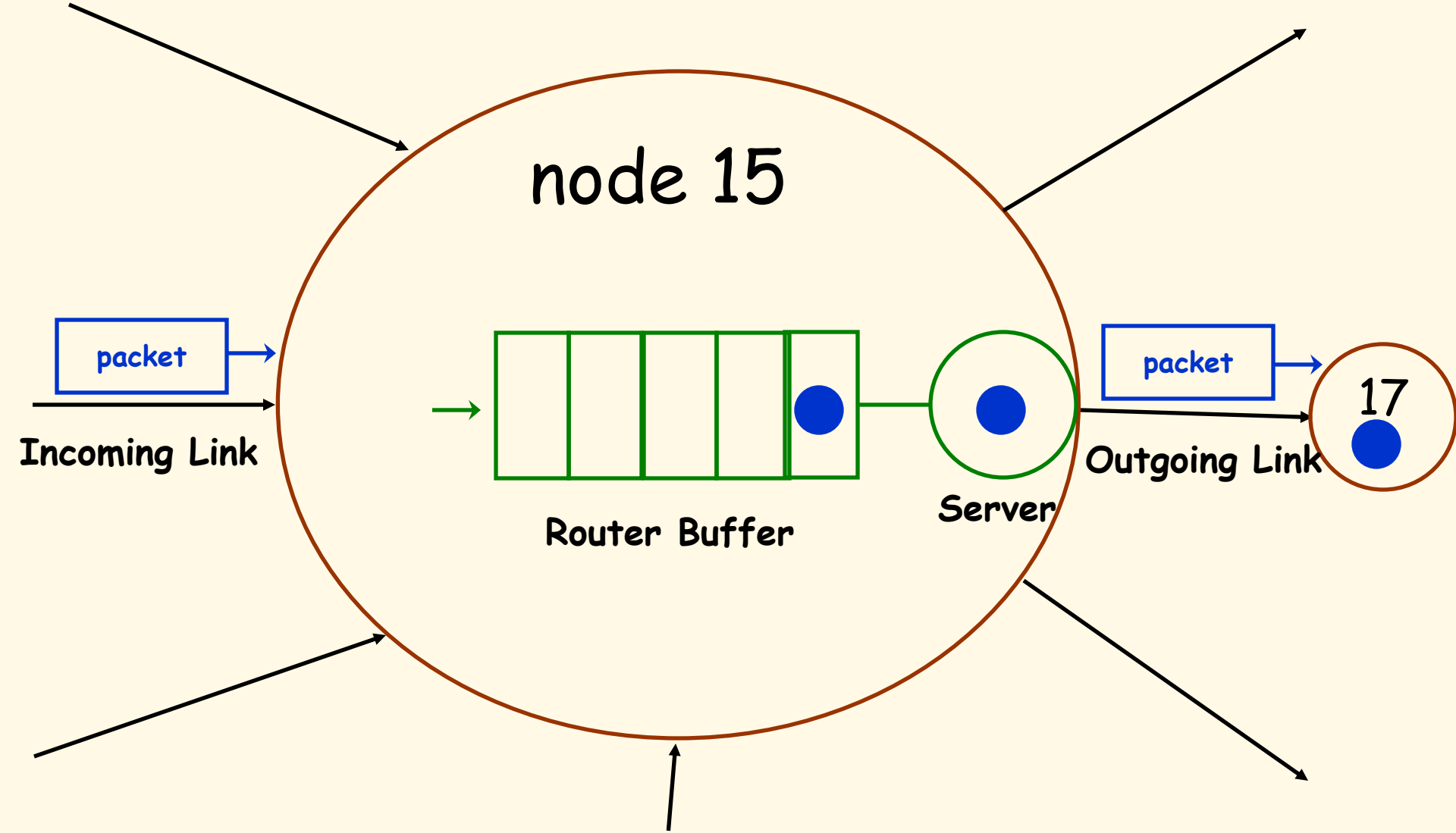
analogy:

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

Interplay between Routing and Forwarding

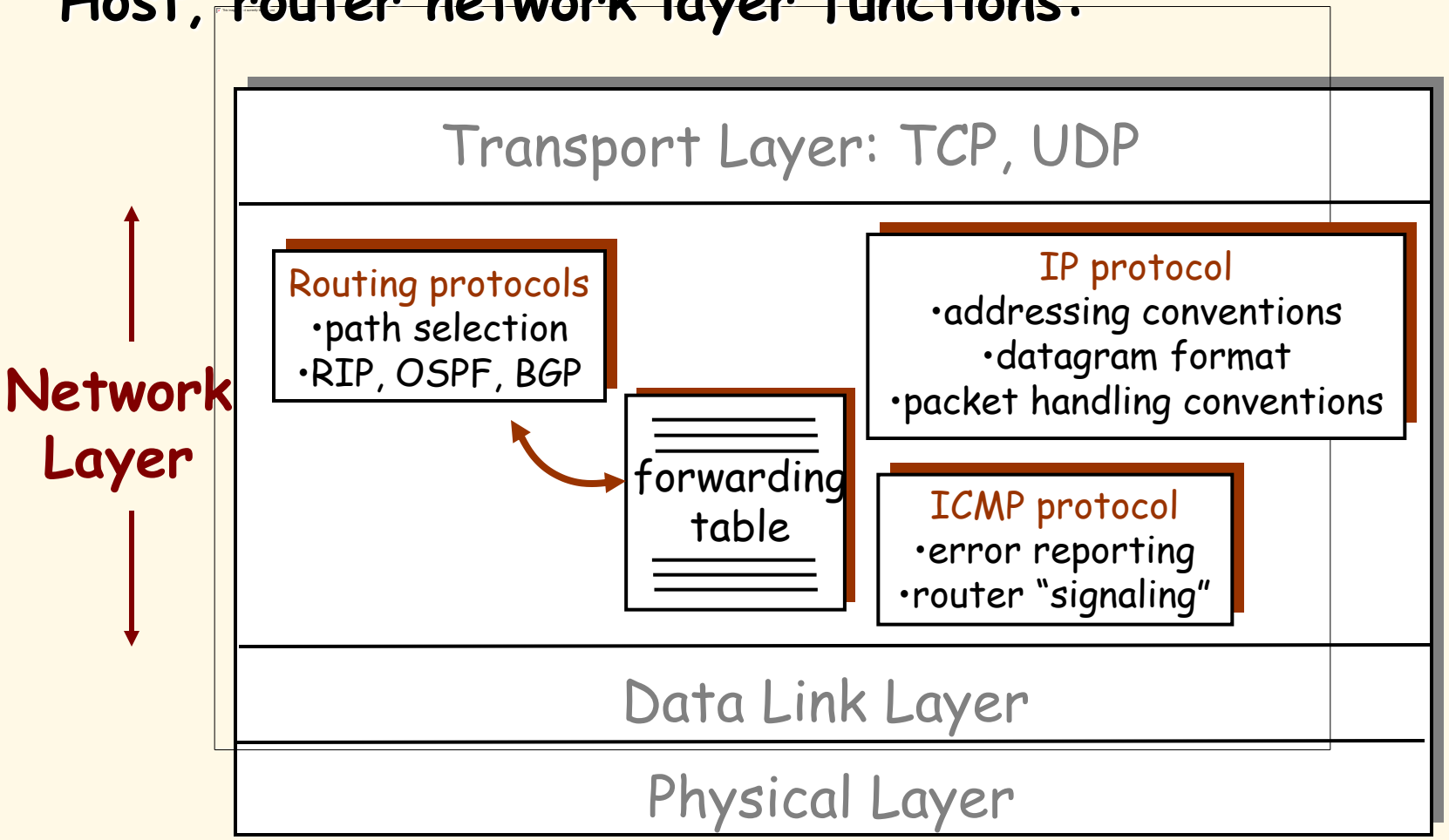


Router Node


node 15
packet
Incoming Link
Router Buffer
Server
packet
Outgoing Link
17

The Internet Network Layer

Host, router network layer functions:



Quick Routing Overview

Computer Networks

Distance Vector Routing

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 Classification

Adaptive Routing

based on current measurements of traffic and/or topology.

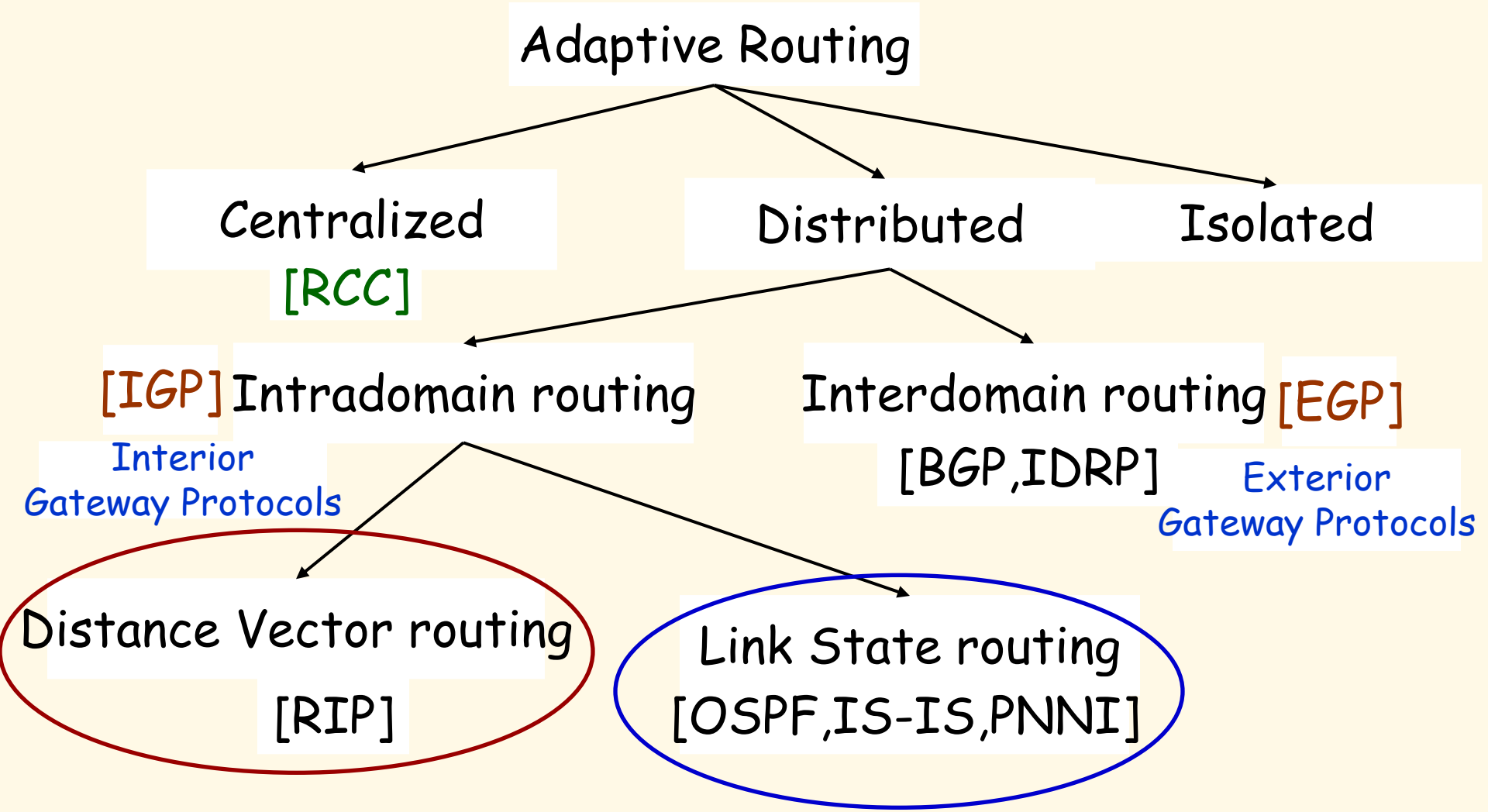
1. centralized
2. isolated
3. distributed

Non-Adaptive Routing

routing computed in advance and off-line

1. flooding
2. static routing using shortest path algorithms

Internetwork Routing [Halsall]



Adaptive Routing Design

Design Issues:

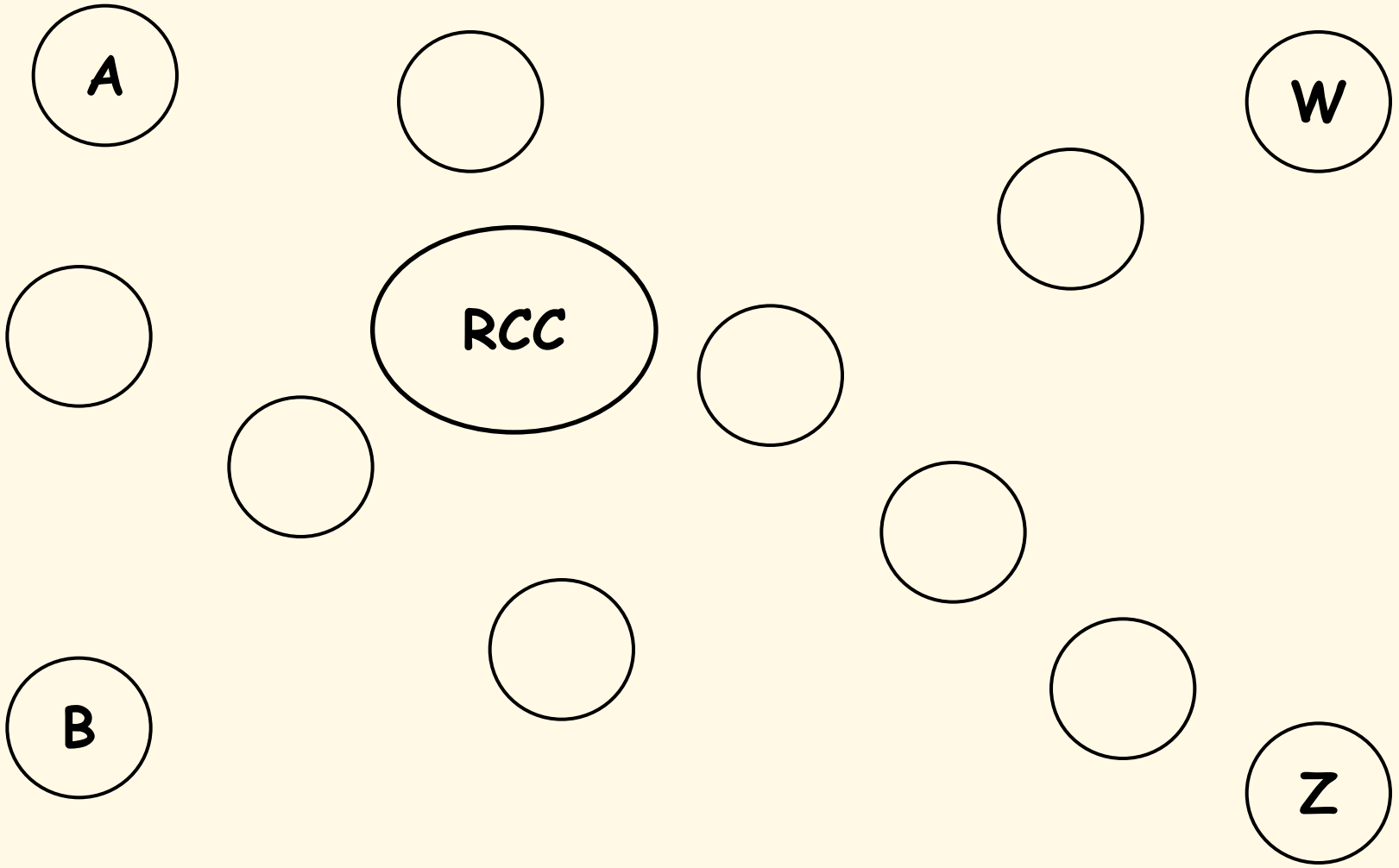
1. How much overhead is incurred due to gathering the routing information and sending *routing packets*?
2. What is the time frame (i.e, the frequency) for sending *routing packets* in support of adaptive routing?
3. What is the complexity of the routing strategy?

Adaptive Routing

Basic functions:

1. Measurement of pertinent network data.
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.

Centralized Routing



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

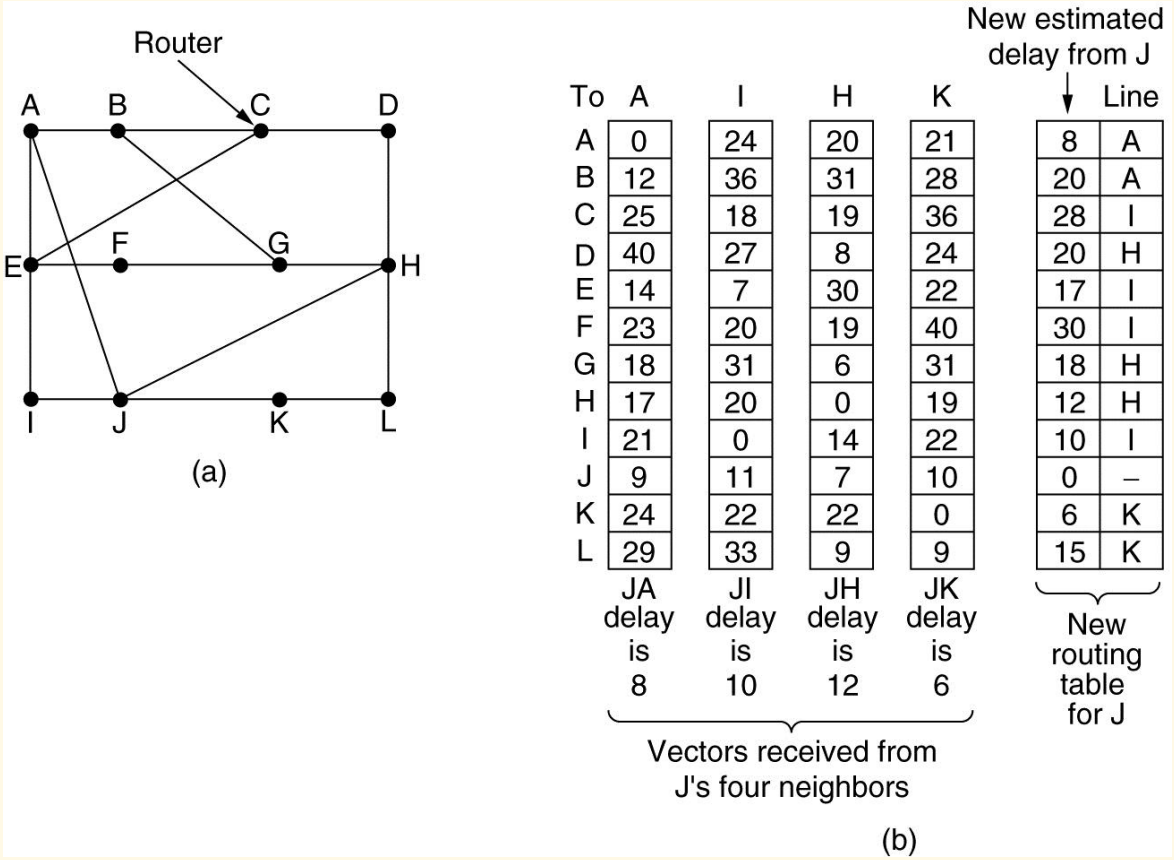


Figure 5-9.(a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.

Tanenbaum

Distance Vector Routing

{Kurose & Ross version}

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Define

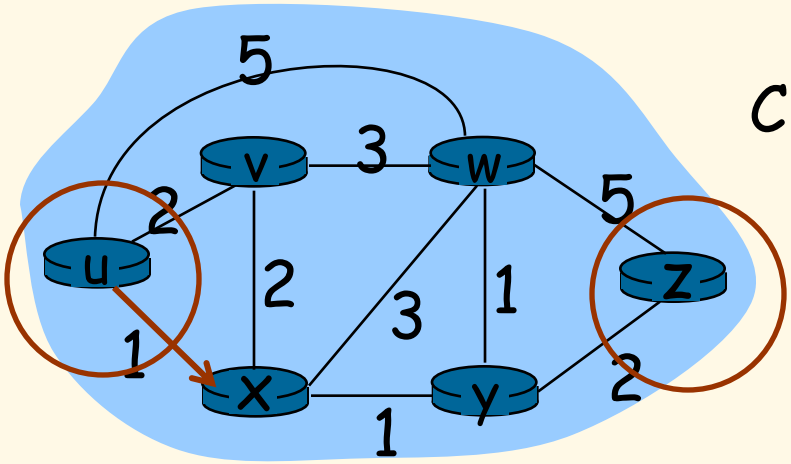
$d_x(y)$:= cost of least-cost path from x to y

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 (3)

- $D_x(y)$ = 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 (4)

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:

$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$ 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 (5)

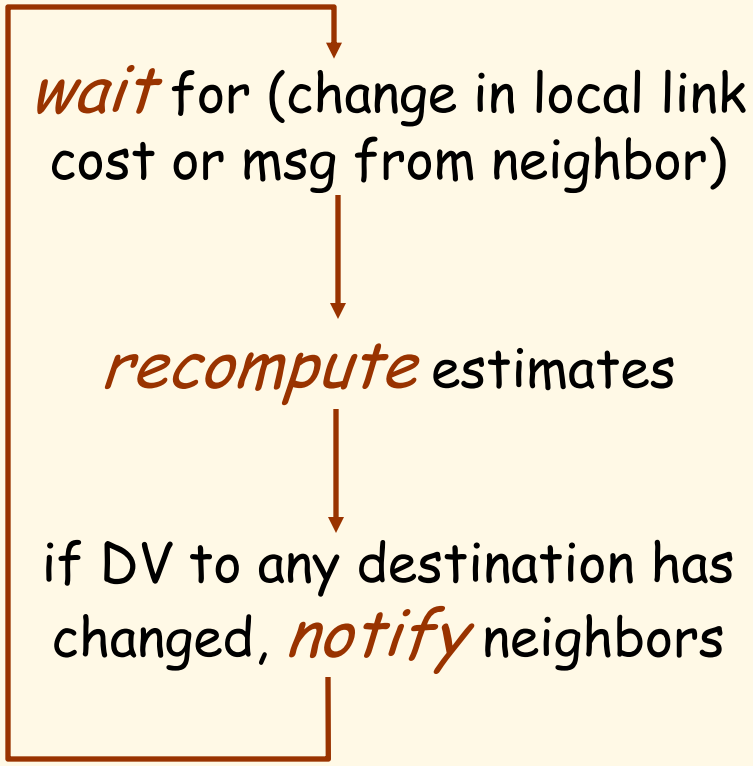
Iterative, asynchronous: each local iteration caused by:

- 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:



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

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

node x table

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

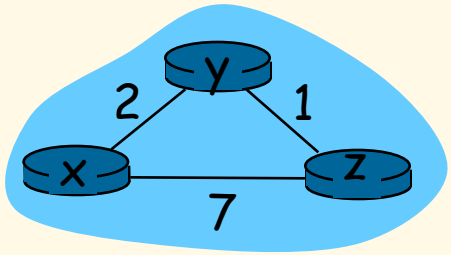
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

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

node z table

from \ cost to	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

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

cost to

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

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cost to

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

cost to

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

node z table

cost to

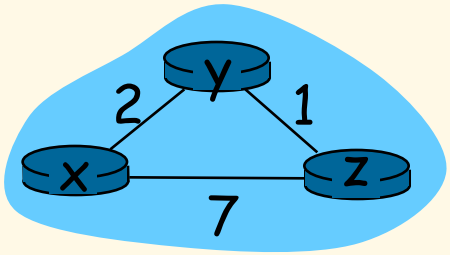
from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

cost to

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

cost to

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

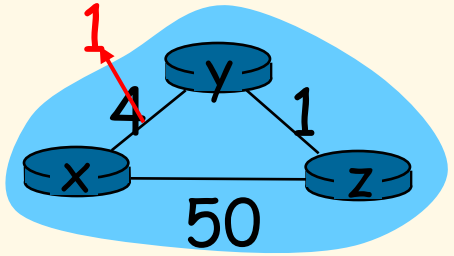


time

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.

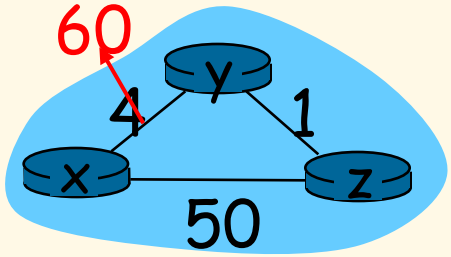
At time t_1, z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbors its DV.

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 text!



Poisoned 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)
- will this completely solve count to infinity problem?

Distance Vector Summary

The Network Layer is responsible for routing and forwarding.

The routing process is used to build forwarding lookup tables.

Forwarding uses the lookup table to move an incoming packet to the correct outgoing link queue.

Routing algorithms use link cost metrics such as hops or delay.

Distance Vector (DV) is an intradomain adaptive routing algorithm that does not scale well.

Distance Vector Summary

DV (originally the old ARPA algorithm) employs the Bellman-Ford shortest path algorithm and currently is used in the RIP, RIP-2, BGP, ISO IDRP and Novell IPX protocols.

DV routers:

- keep distances to ALL intranet routers in a distance vector which is periodically updated and transmitted to each of its neighbors.
- reacts to changes in its neighbors' distance vectors and to topology changes (i.e., nodes and/or links coming up or going down).

In distance vector routing "bad news travels slowly and good news travels quickly".