# EITF25
# Internet Routing

Jens A Andersson

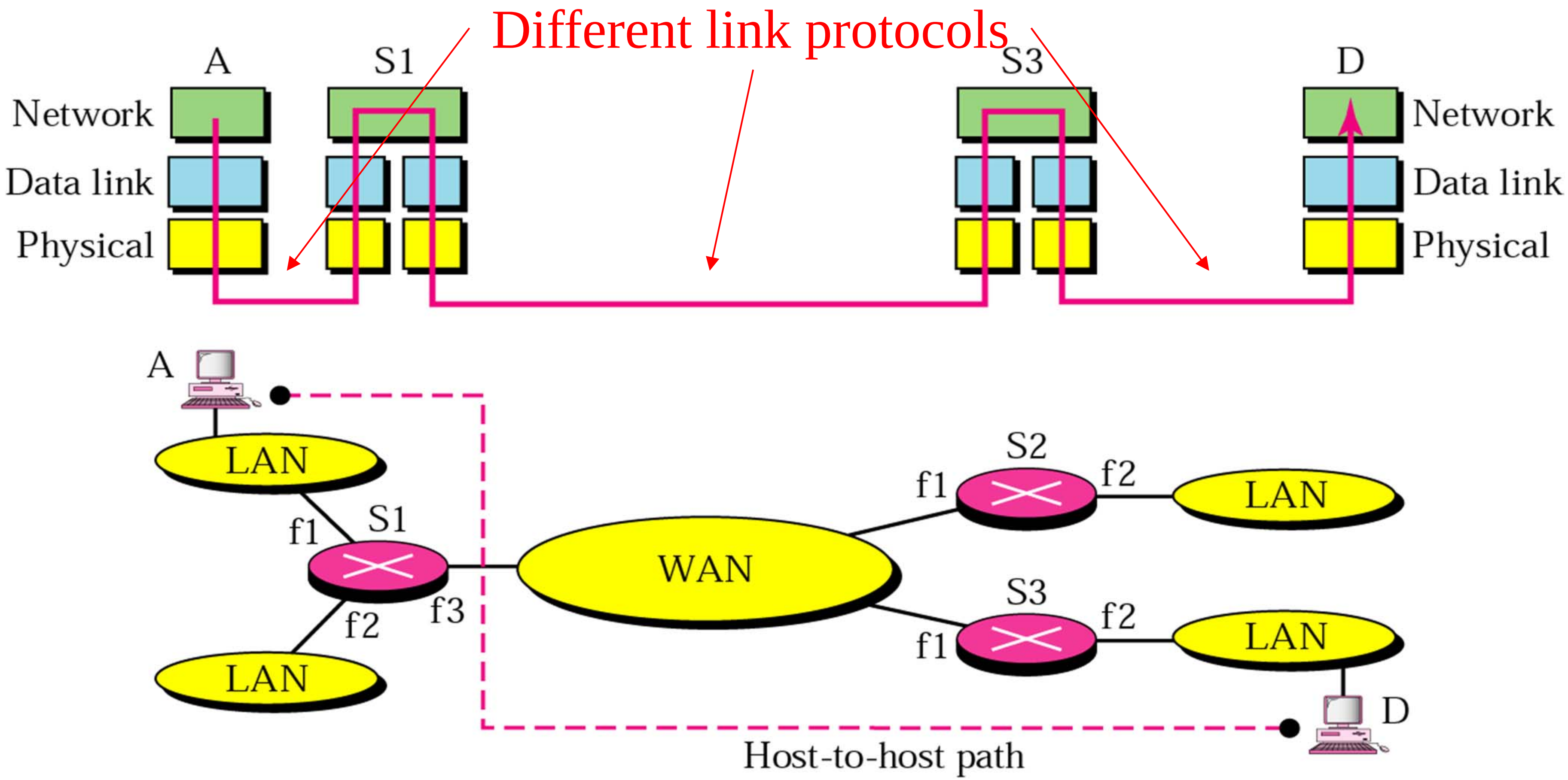
# Study Guide

- Kihl & Andersson:
  - Ch 8, 9.3 – 9.4
- Stallings:
  - Ch 19.1 & 19.2
- Forouzan 5th ed
  - Ch 20.1 – 20.3, 21.1 – 21.2

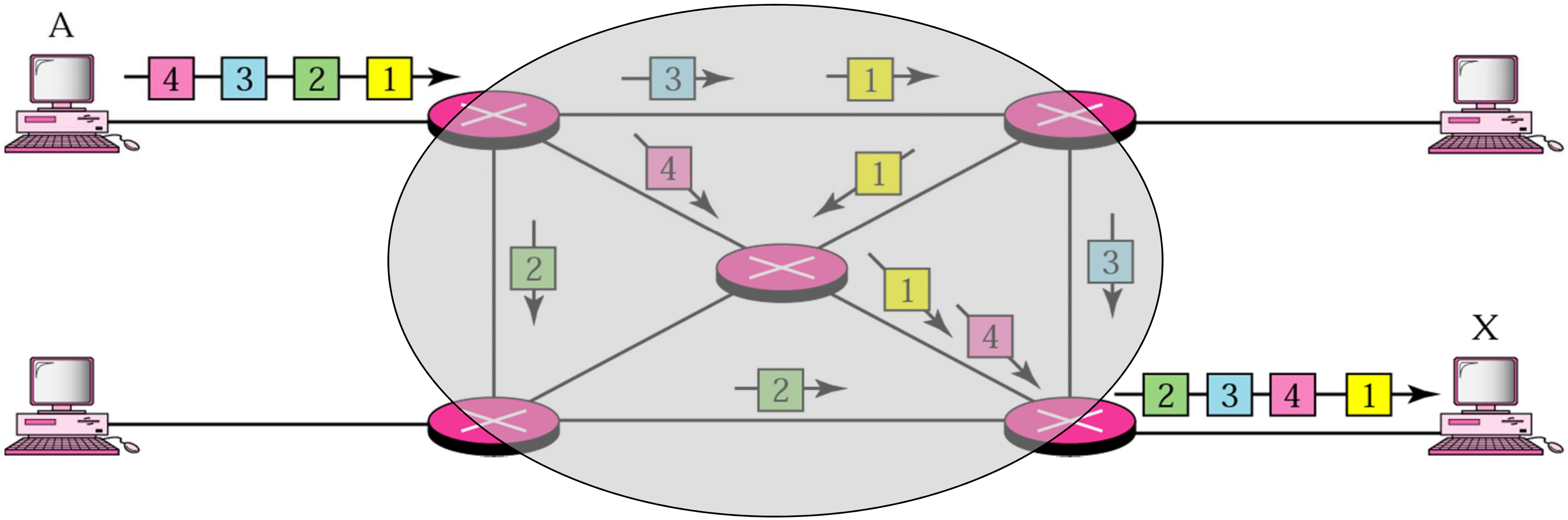
# Routing

- The Routing Concept
- Unicast Routing
- Multicast Routing (short overview)

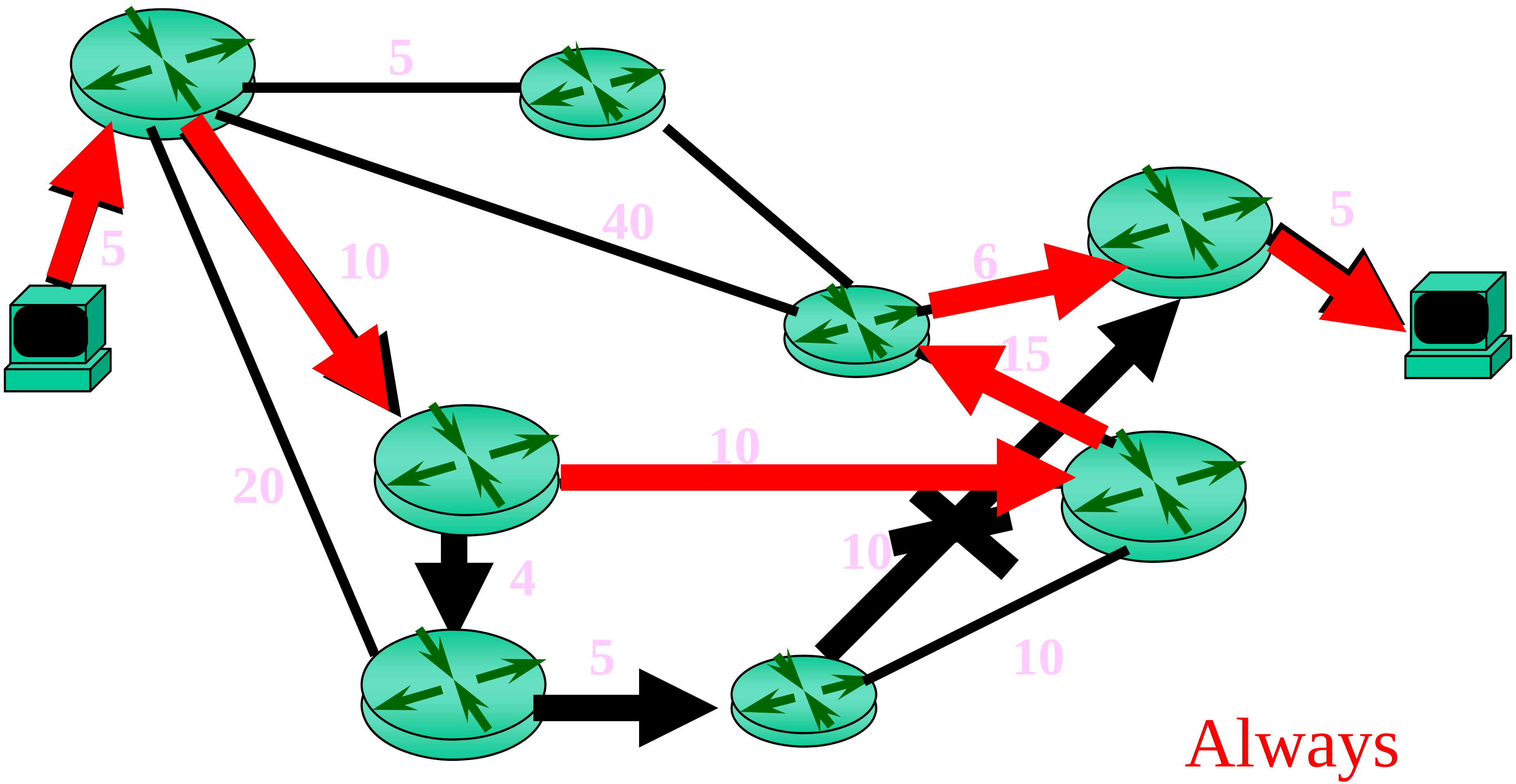
# Network Layer (Layer 3)



# Datagrams in the network



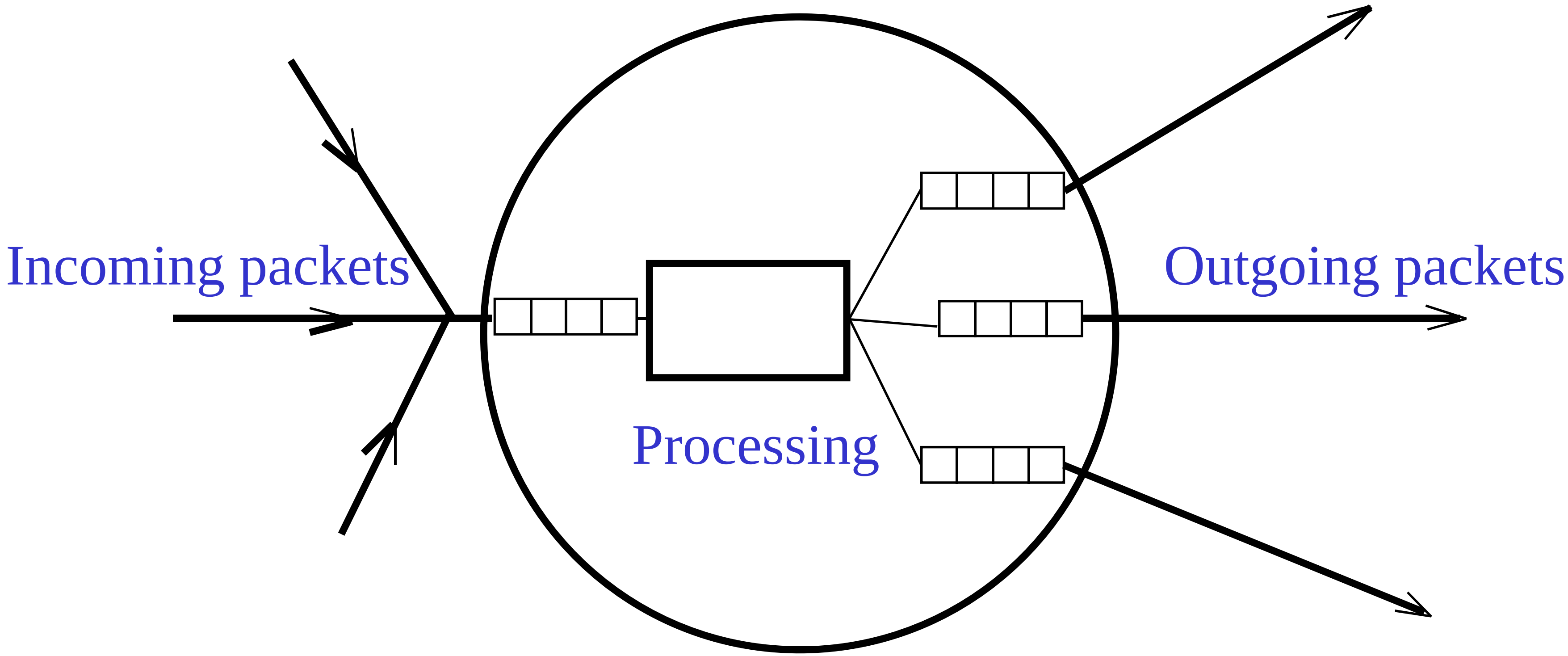
# Task: Select Best Path



# Router

- A router is a type of internetworking device that passes data packets between networks, based on Layer 3 or Network Layer addresses.
- A router has the ability to make intelligent decisions regarding the best path for delivery of data on the network.
- Routing is performed based on network identities, not host identities
  - Last step (delivery to host) performed with ARP

# Schematic View of a Router



# Routing Technologies

- No “intelligence”
- Centralized
- Distributed

# No “intelligence”: Flooding

- Output all packets/datagrams
  - On all ports/interfaces/links
  - Except ingress port/interface/link
- Problem?
  - Looping packets
  - Two solutions
    - Avoided with TTL counter
    - Remember processed datagrams

# Centralized routing technology

- Database and algorithm centralized
  - Network equipment informs central function
- Packet forwarding distributed
  - of course!?

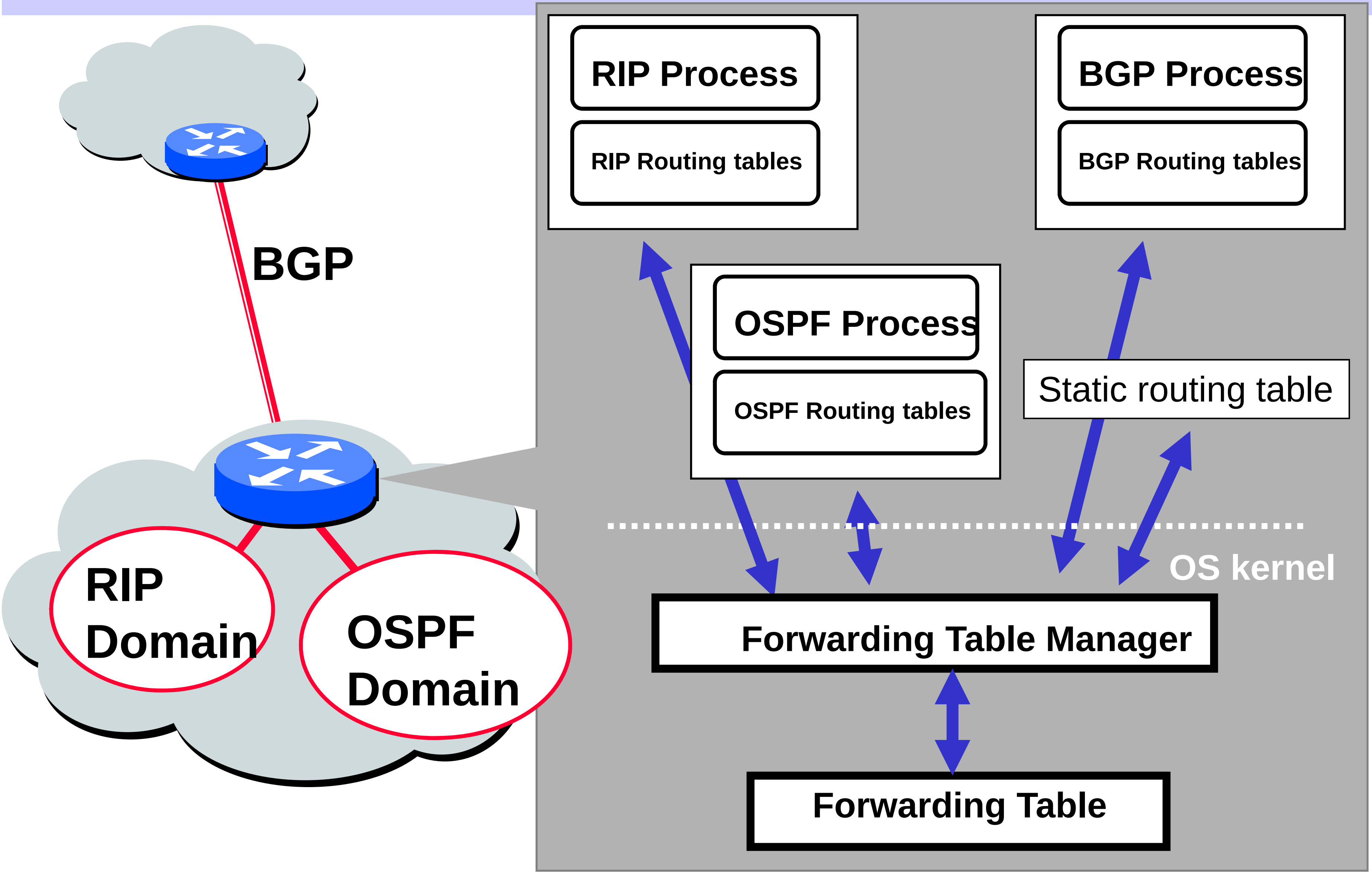
# Distributed routing technology

- Routing process distributed among all routers
- Two approaches
  - Distance Vector
    - Each nodes best-path information is distributed to its neighbours
    - Best end-to-end paths found from comparison of all next-hop choices
    - Simple, low demands on processor and memory
  - Link State
    - Local topology information is flooded to all nodes
    - Best end-to-end paths are computed locally at each router
    - More complicated, high demands on processor and memory

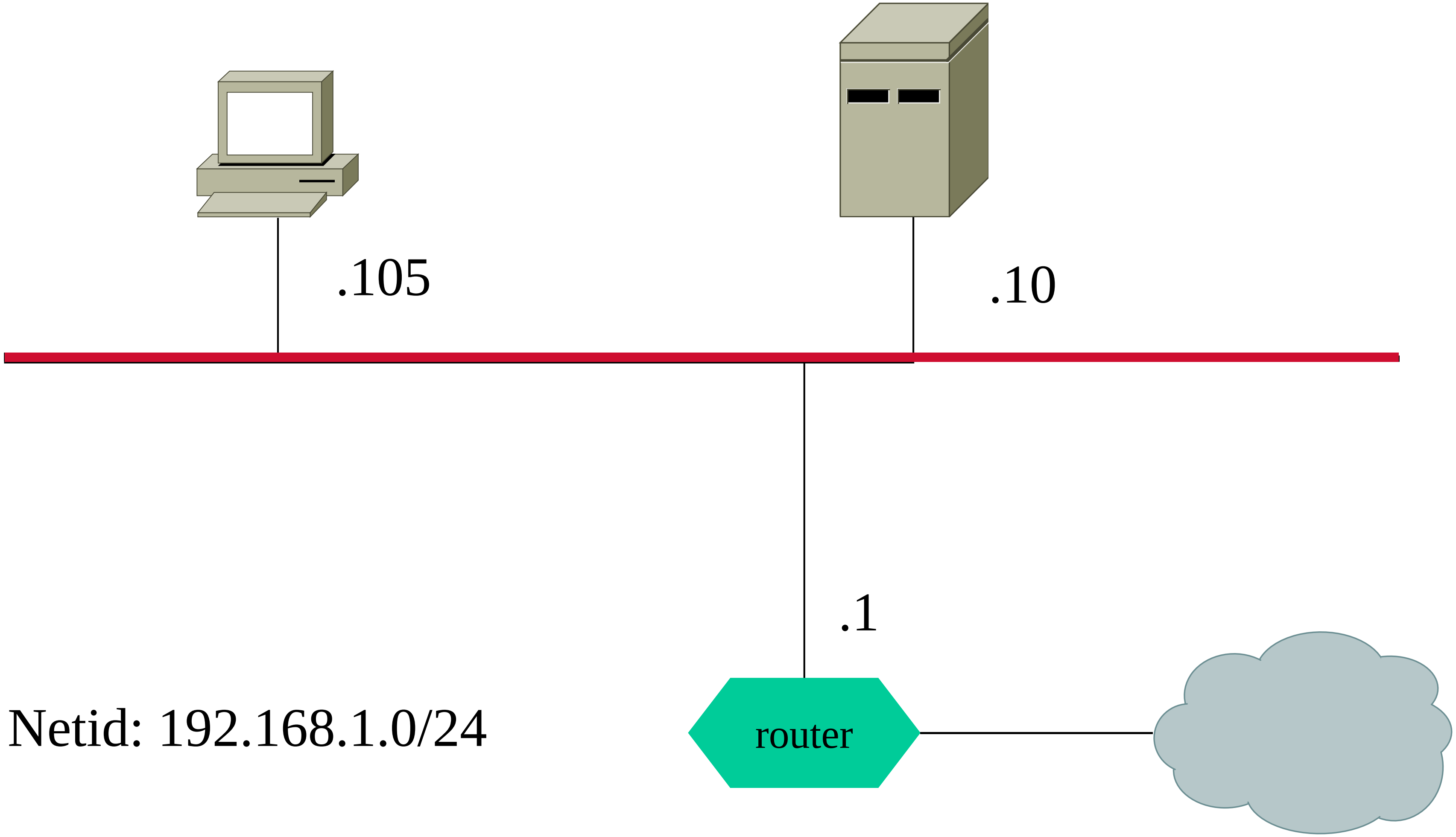
# Path Cost, Link Metric

- Possible metrics
  - hop count
  - inverse of the link bandwidth
  - delay
  - dynamically calculated
  - administratively assigned
  - combination

# Routing Tables and Forwarding Table

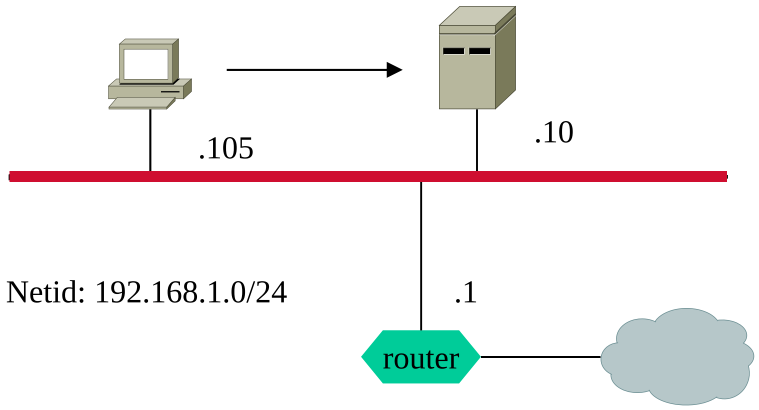


# Local routing & ARP (1)




Netid: 192.168.1.0/24

# Local Routing & ARP (2)




Send datagram to 192.168.1.10!

Is destination on same subnet?

*Sender compares own net-id with net-id of destination.*

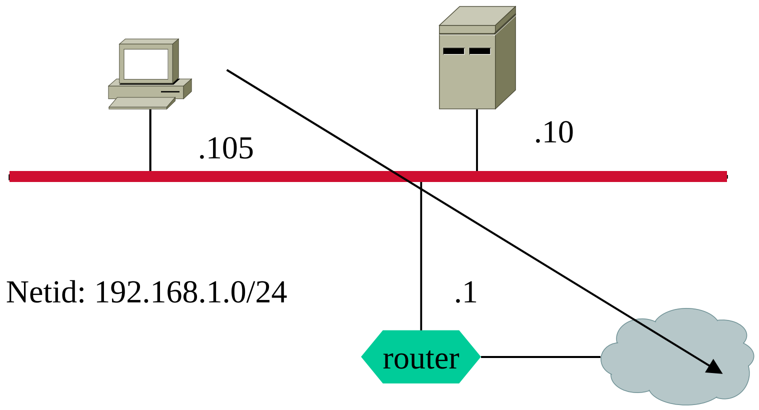
in this case YES

Is MAC-address of destination in ARP-cache?

if YES, use it

if NO, use ARP

# Local Routing & ARP (3)




Send datagram to 10.0.100.35!

Is destination on same subnet?

*Sender compares own net-id with net-id of destination.*

in this case NO

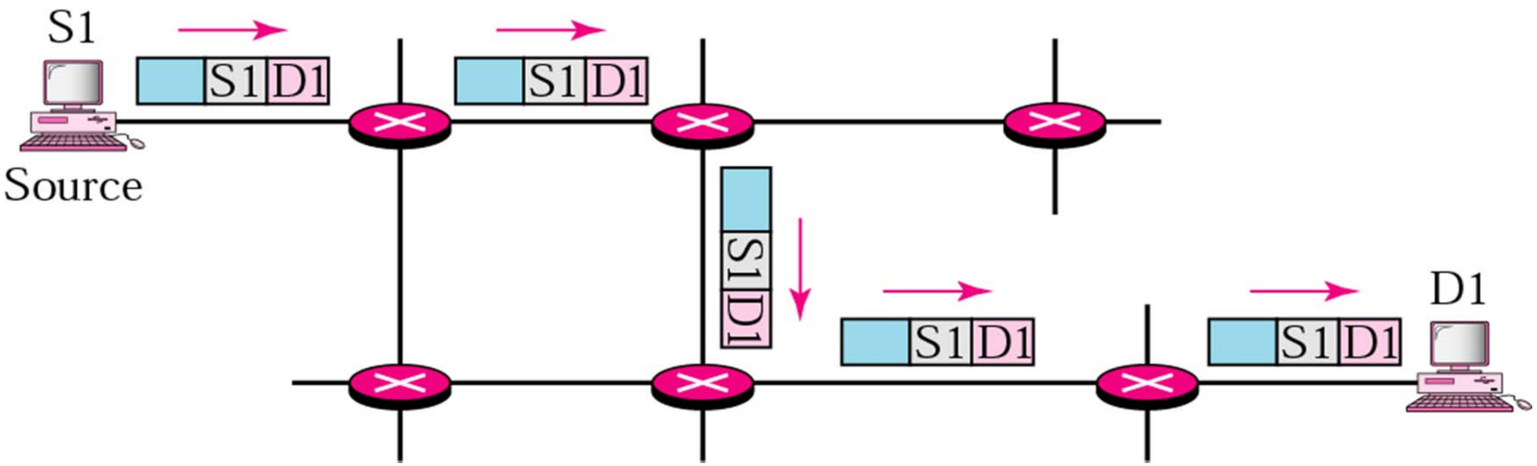
Is network address of default gateway known?

If YES, is MAC-address of default gateway in ARP-cache?

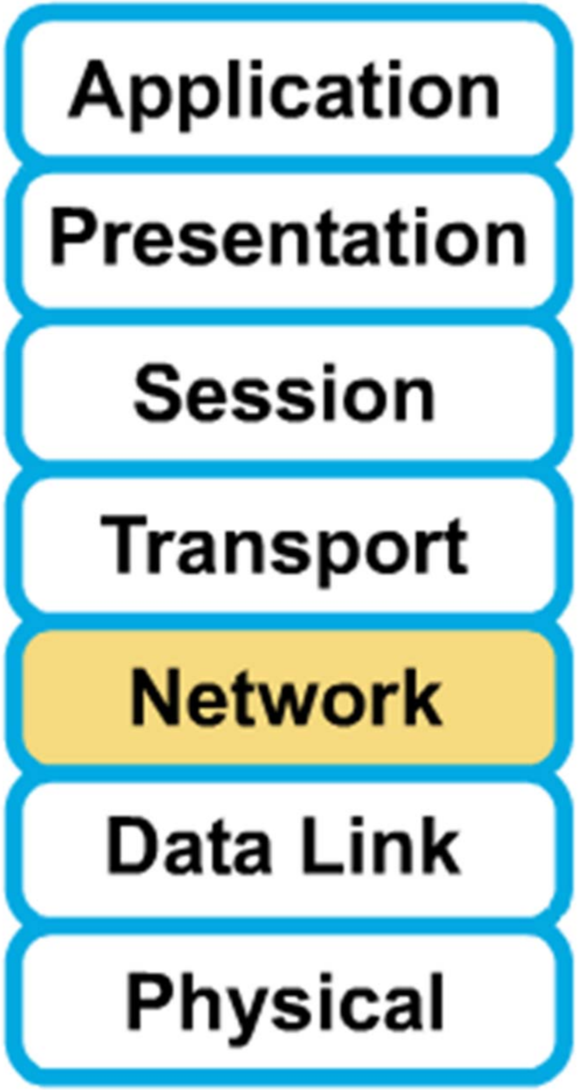
if YES, use it

if NO, use ARP

# Unicast Routing

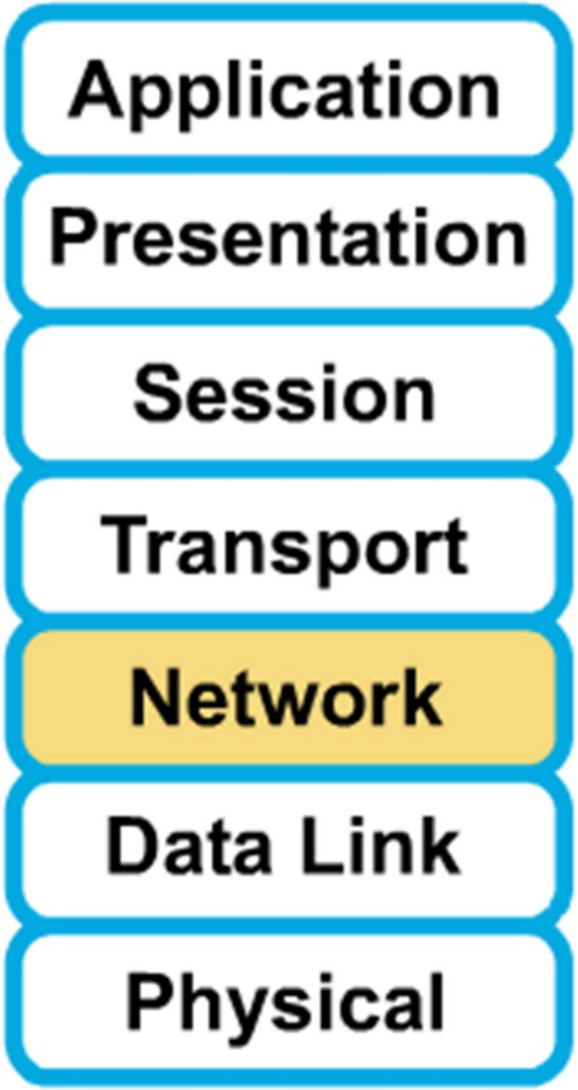


# Routable protocols




- Protocols that provide support for the network layer are called routed or routable protocols.
- IP is a network layer protocol, and because of that, it can be routed over an internetwork.

# Routing protocols

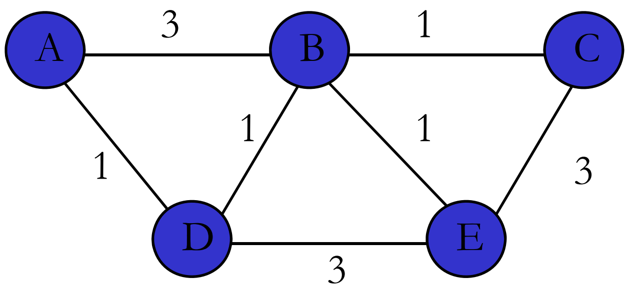


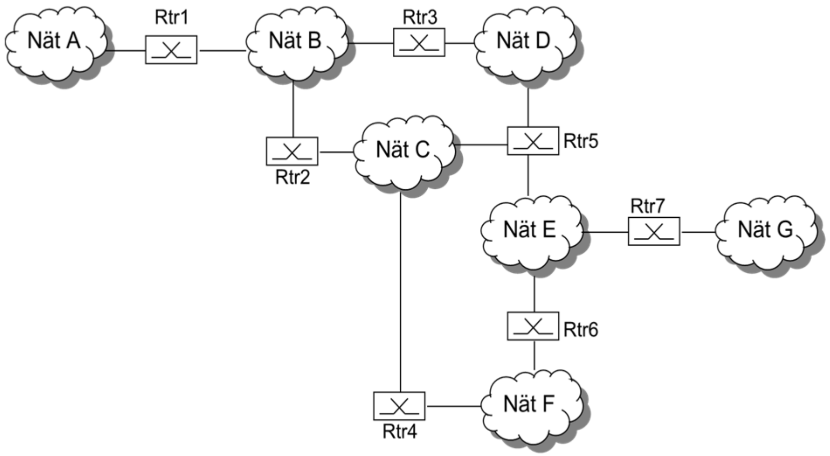

- Routing protocols determine the paths that routed protocols follow to their destinations.
- Routing protocols enable routers that are connected to create a map, internally, of other routers in the network.

# Routing

- The art of building least-cost trees
  - From sender to receiver
  - From each nod to every other node
- Three principles
  - Distance Vector
  - Link State
  - Path Vector
    - Policy-based routing

# An Internet and its graphical representation



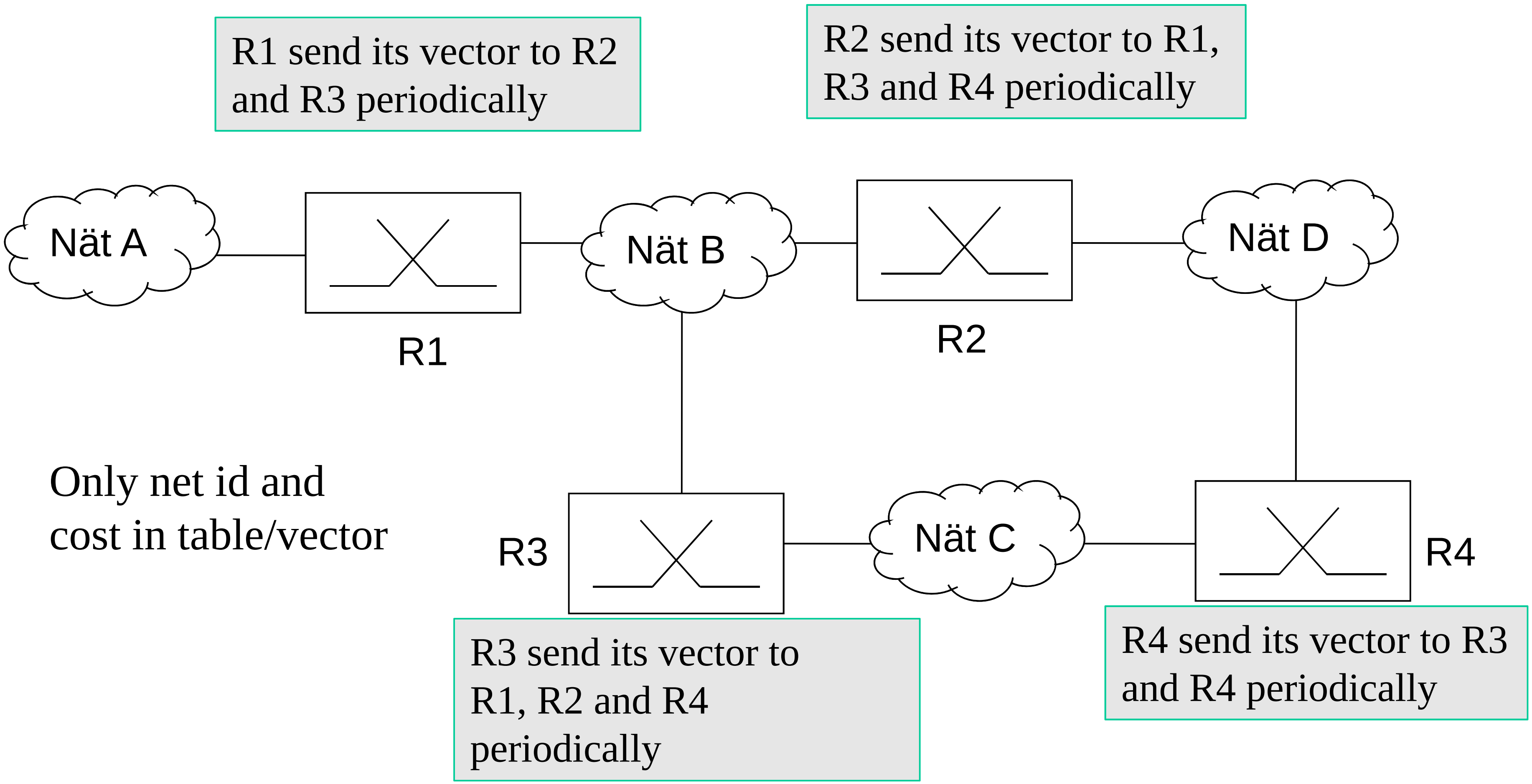




- Graphs are built by
  - Links connects nodes
  - Nodes are ”destinations”
- In routing two differnet node types:
  - Nets i.e. destinations
  - Routers connection nets, next hop
- The application of spanning tree algoritms can get fuzzy …

# A distance vector

| Net-id | Cost |
|---|---|
| - | - |
| - | - |
| - | - |
| - | - |
| - | - |

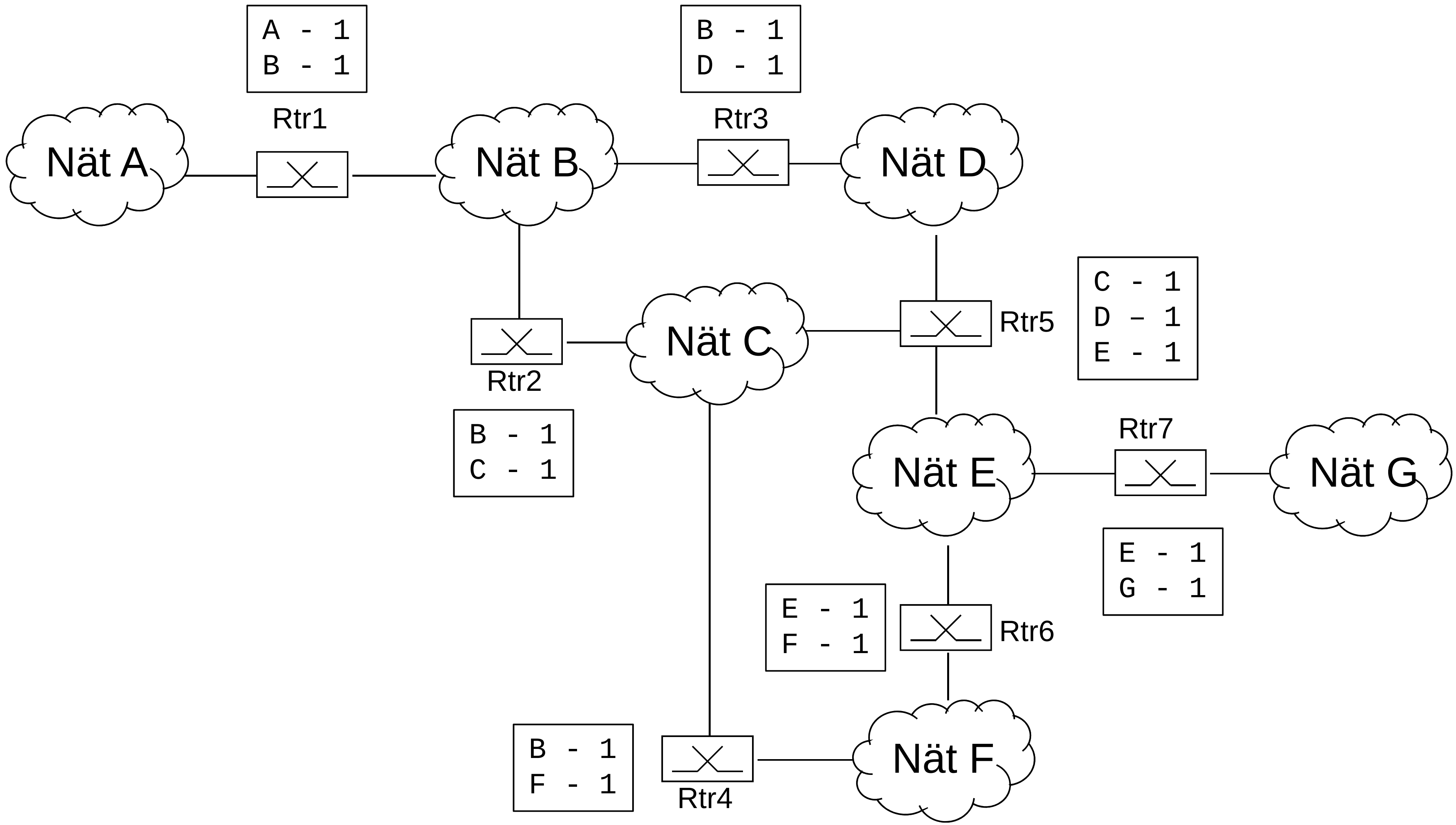
# Distance Vector: Principle



# Distance vector: principle

- **Send** all known best paths **to neighbours**
  - Periodically
  - At changes
- **Update** routing tables with
  - New nodes
  - Change of cost or path
- ”Global knowledge spread locally”

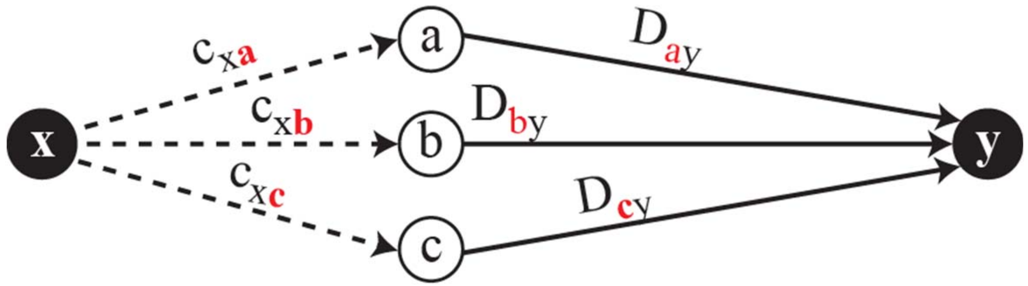
# Exemple based on  hop count



# Bellman-Fords algorithm

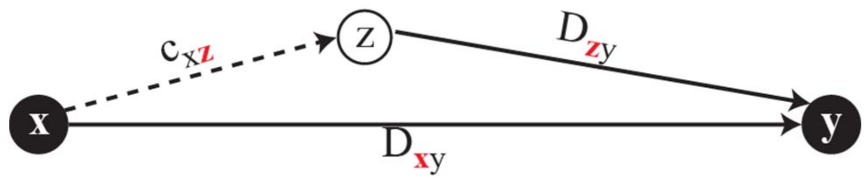
```
(1)    if (advertised destination not in table) then
             update table
(2)    else
(2.a)  if (advertised next-hop = next-hop in table)
then
             replace entry
(2.b)  else
(2.b.i)      if (advertised hop count < hop count in
table) then
             replace entry
(2.b.ii)     else
             do nothing
```

# Alternative: Bellman-Fords equation




a. General case with three intermediate nodes

$$D_{xy} = \min\{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}) \ldots\}$$




b. Updating a path with a new route

$$D_{xy} = min\{D_{xy}, (c_{xz} + D_{zy})\}$$

Note! $D_{xy}$ can change without node z being added!

# Updates

Rtr3

| Original | From Rtr5 +1 | Updated |
|---|---|---|
| B - 1 | C 2 | B – 1 |
| D - 1 | D 2 | C rtr5 2 |
| | E 2 | D – 1 |
| | | E rtr5 2 |

Rtr1

| Original | From Rtr2 +1 | Updated |
|---|---|---|
| A - 1 | B 2 | A - 1 |
| B - 1 | C 2 | B – 1 |
| | | C rtr2 2 |

# Distance Vector, reflections

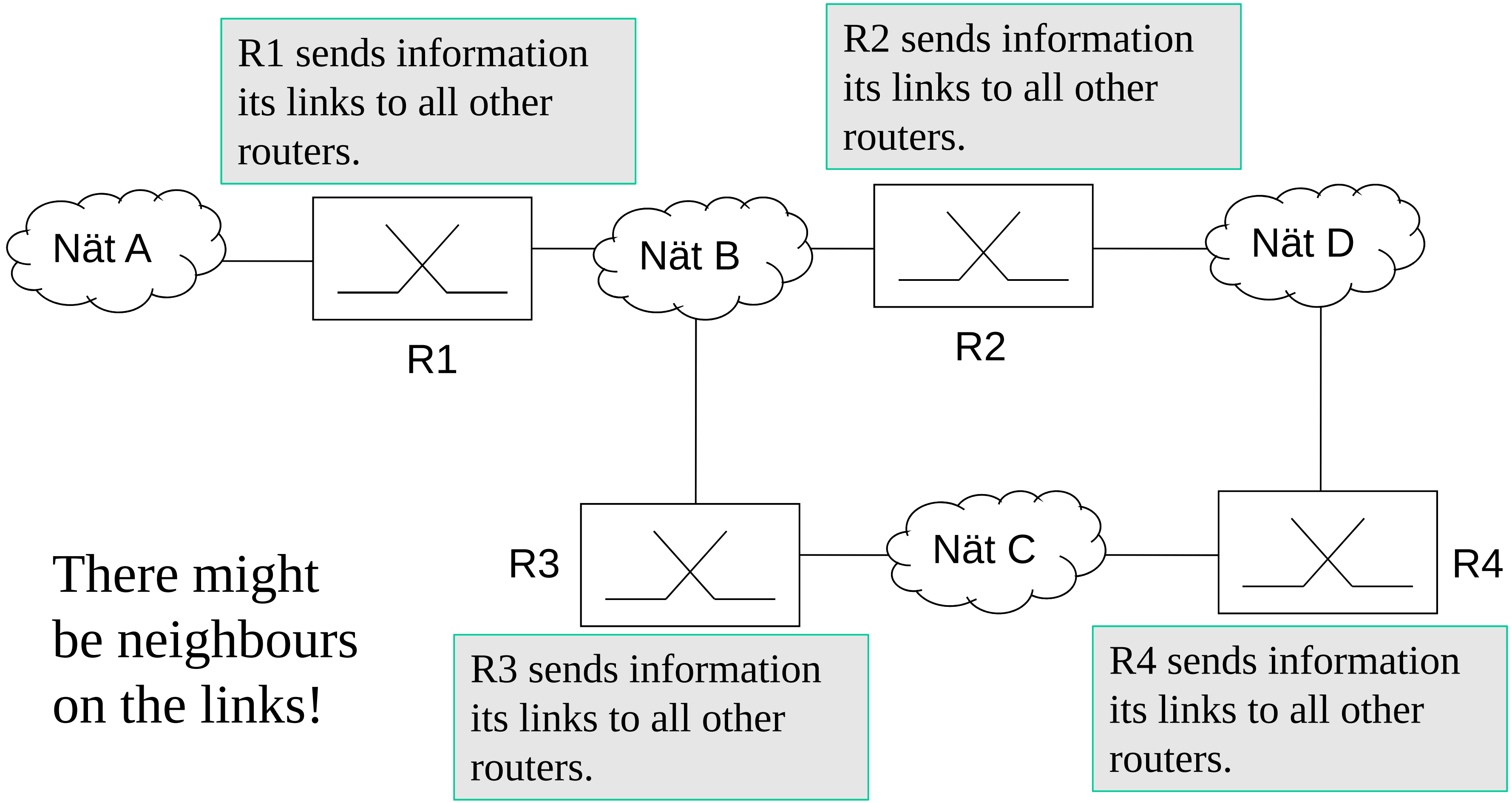
- Periodic updates!?
  - How to find neighbours?
  - How to find out neighbour is lost?
- Problem with loss of links and nodes (behind neighbours).

# More on distance vector

In ETSF10

- Count to infinity
- Two, three node instability
- Split Horizon
- Poison Reverse
- Routing protocol RIP

# Link State: principle




There might be neighbours on the links!

# Link state: principle

- **Local topology** info is **flooded** globally (LSP)
  - Periodically (in practise very seldom, typically every hour)
  - On local changes
- Create database in <u>each</u> node with **all** link states
- Update routing table when new information is added to database (apply Shortest Path First)
- ”Local knowledge spread globally”

# LSP (Link State Packet)

| Advertiser | Network | Cost | Neighbor |
|---|---|---|---|
| . . . . . . . | . . . . . . | . . . . . . . . . . . | . . . . . . . . . . . |
| . . . . . . . | . . . . . . | . . . . . . . . . . . | . . . . . . . . . . . |
| . . . . . . . | . . . . . . | . . . . . . . . . . . | . . . . . . . . . . . |

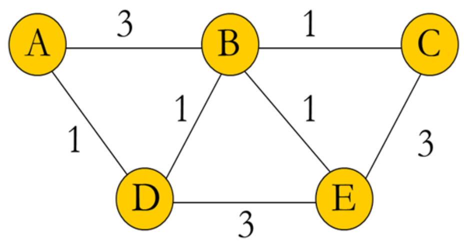
# Link State Databas, exemple

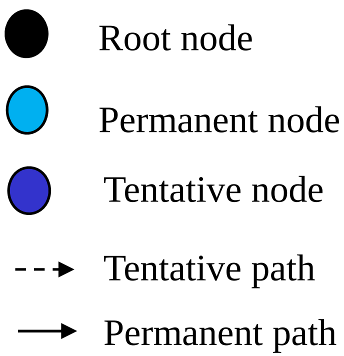
| Advertiser | Network | Cost | Neighbor |
|---|---|---|---|
| A | 14 | 1 | B |
| A | 78 | 3 | F |
| A | 23 | 2 | E |
| **B** | **14** | **4** | **A** |
| **B** | **55** | **2** | **C** |
| C | 55 | 5 | B |
| C | 66 | 2 | D |
| **D** | **66** | **5** | **C** |
| **D** | **08** | **3** | **E** |
| E | 23 | 3 | A |
| E | 08 | 2 | D |
| **F** | **78** | **2** | **A** |
| **F** | **92** | **3** | **—** |

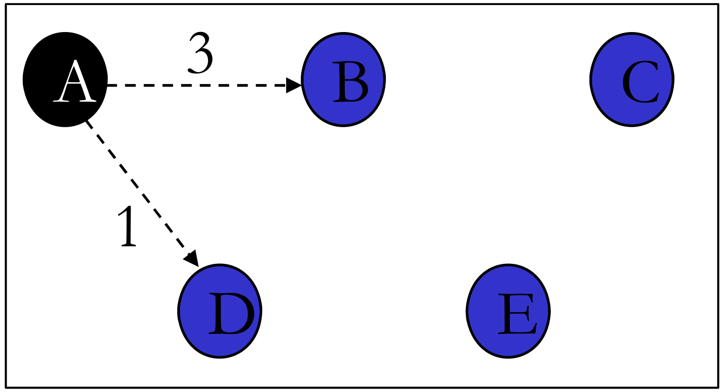
# Dijkstra's algorithm: Shortest Path First

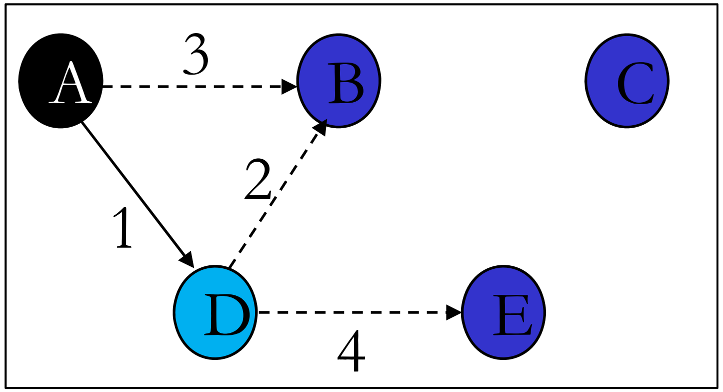
1. Identify the root (the node itself)
2. Attach all neighbor nodes temporarily
3. Make link and node with least cumulative cost permanent
4. Choose this node
5. Repeat 2 and 3 until all nodes are permanent

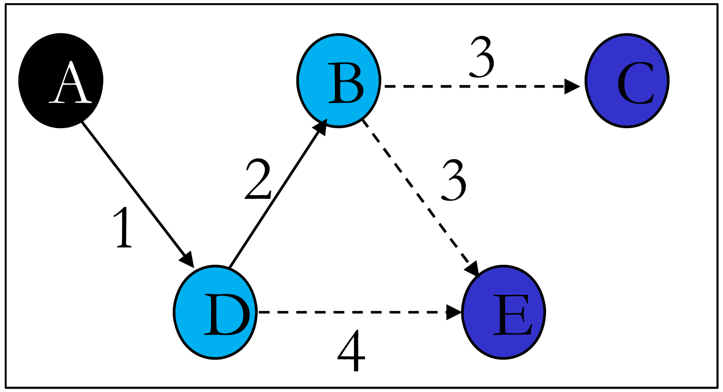
# SPF graph example

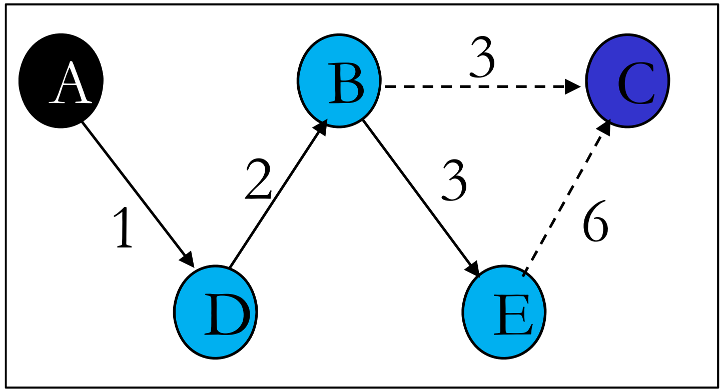


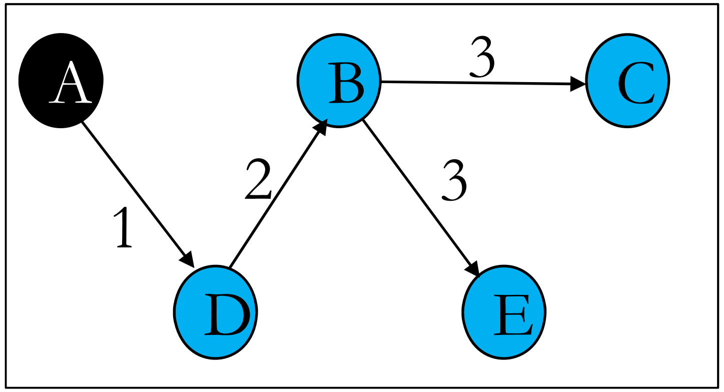



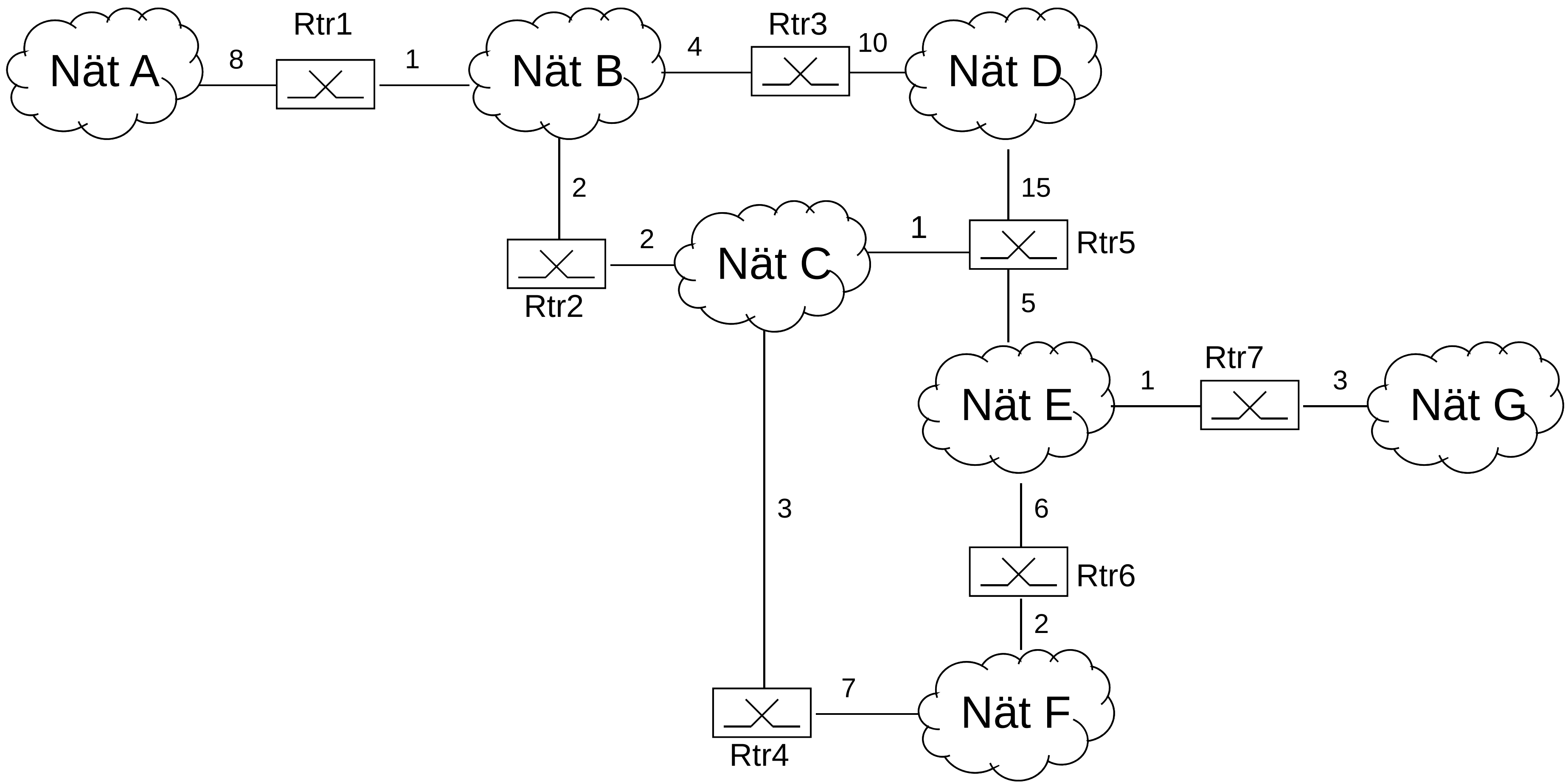
# SPF table example

| Perm.lista | L(A) | L(B) | L(C) | L(D) | L(E) |
|---|---|---|---|---|---|
| $\phi$ | **0** | ∞ | ∞ | ∞ | ∞ |
| {A} | | 3:A | ∞ | **1:A** | ∞ |
| {A,D} | | **2:D** | ∞ | | 4:D |
| {A,D,B} | | | **3:B** | | 3:B |
| {A,D,B,C} | | | | | **3:B** |
| {A,D,B,C,E} | | | | | |

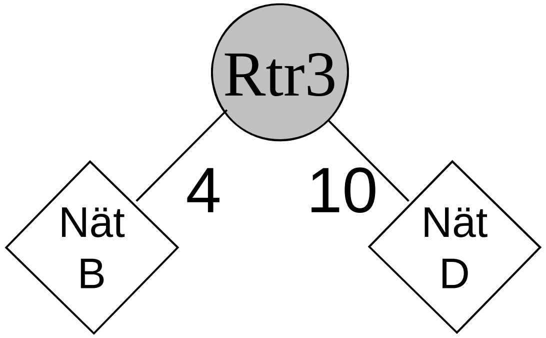
Notation:

Cost:Via

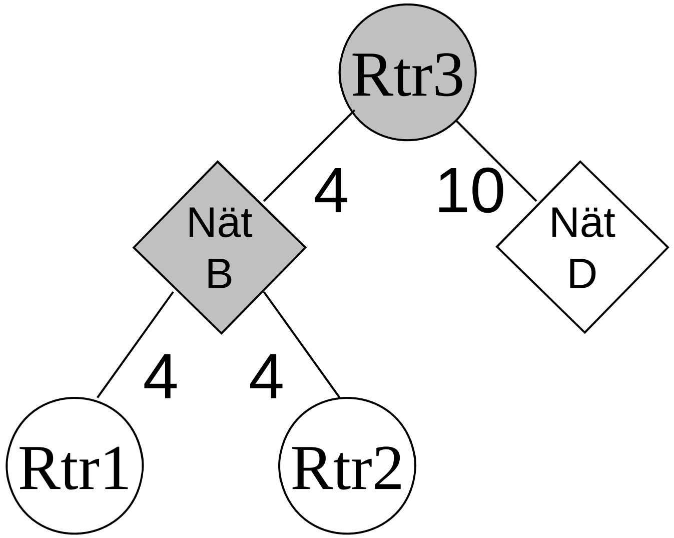
# Link State: an exemple



# SFP: step 1 & 2 for rtr3

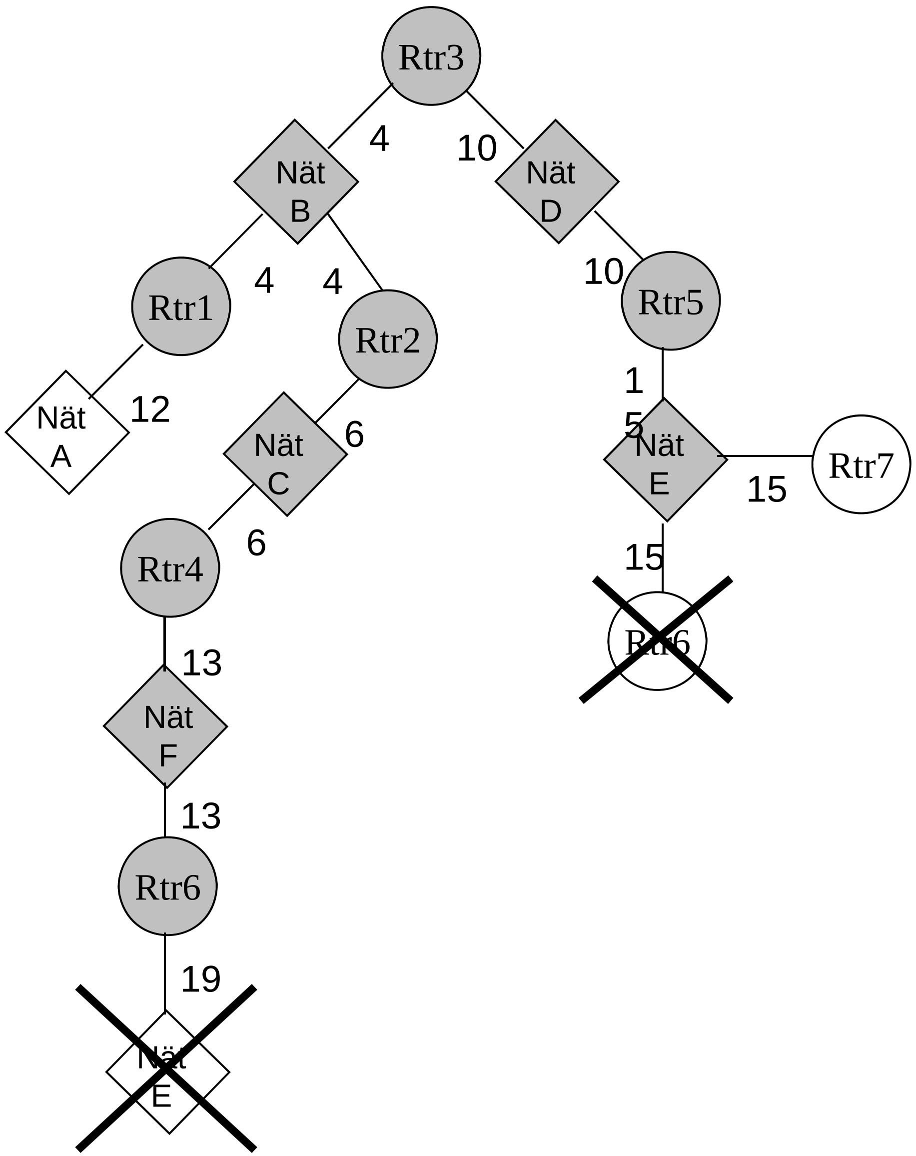



(a)




(b)

# Final SFP tree for rtr3



# Routing table for rtr3

| Nät-id | Next-hop | Kostnad |
|---|---|---|
| A | Rtr1 | 12 |
| B | - | 4 |
| C | Rtr2 | 6 |
| D | - | 10 |
| E | Rtr5 | 15 |
| F | Rtr2 | 13 |
| G | Rtr5 | 18 |

# Link State, reflections

- Periodical updates!?
- Problem with lost links and nodes.
- How to find neighbours?
- How to find out neighbour is lost?

# More on link state

In ETSF10:

- Routing protocol OSPF
- The concept of Areas or how to reduce flooding

# Path Vector Routing

- “Distance Vector routing with add ons”
- In the routing table
  - Destination
  - Next-Hop
  - Path to destination
    - Or rather the path this routing information has taken
- BGP

# Bonus material

- Concept Path Vector Routing
- Introduction to Multicast Routing

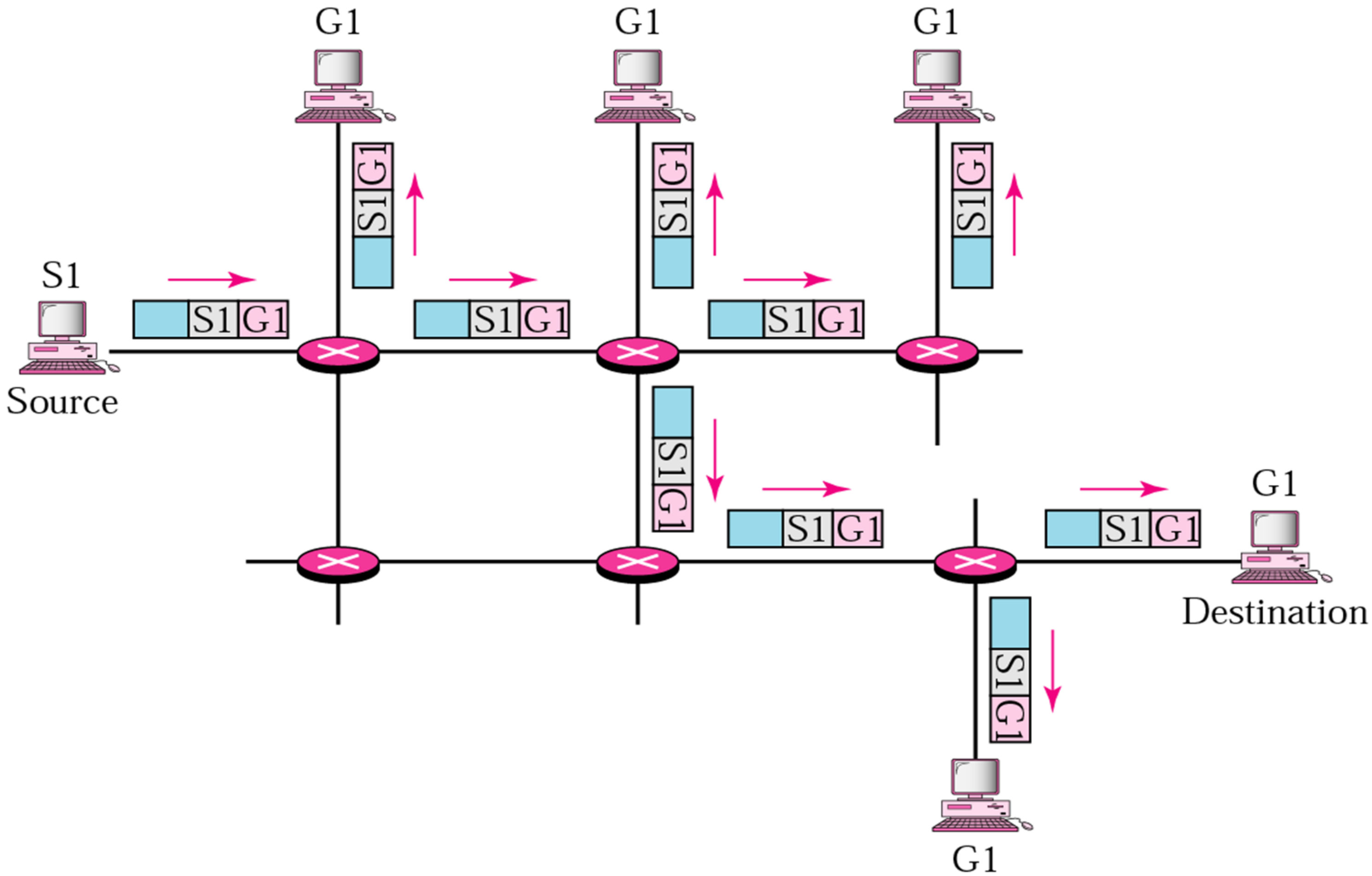
- More on this in ETSF10

# Path Vector Routing Table

| Network | Next Router | Path |
|---|---|---|
| N01 | R01 | AS14, AS23, AS67 |
| N02 | R05 | AS22, AS67, AS05, AS89 |
| N03 | R06 | AS67, AS89, AS09, AS34 |
| N04 | R12 | AS62, AS02, AS09 |

AS = Autonomous System, approx = Organisation

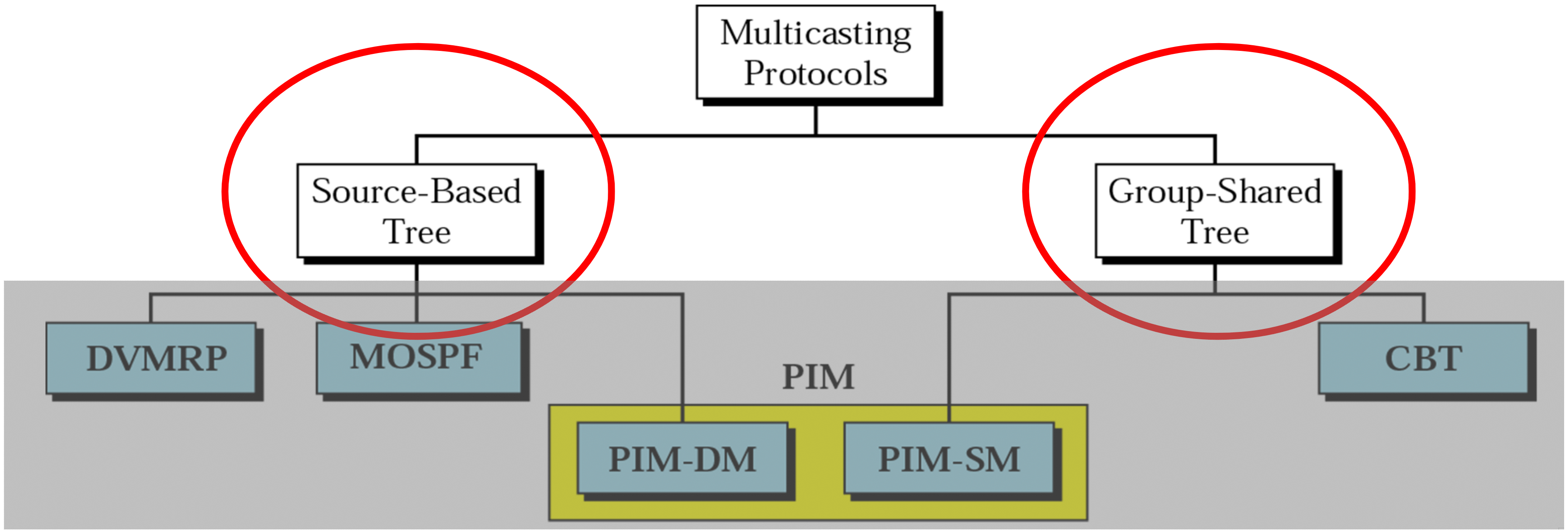
# Multicasting



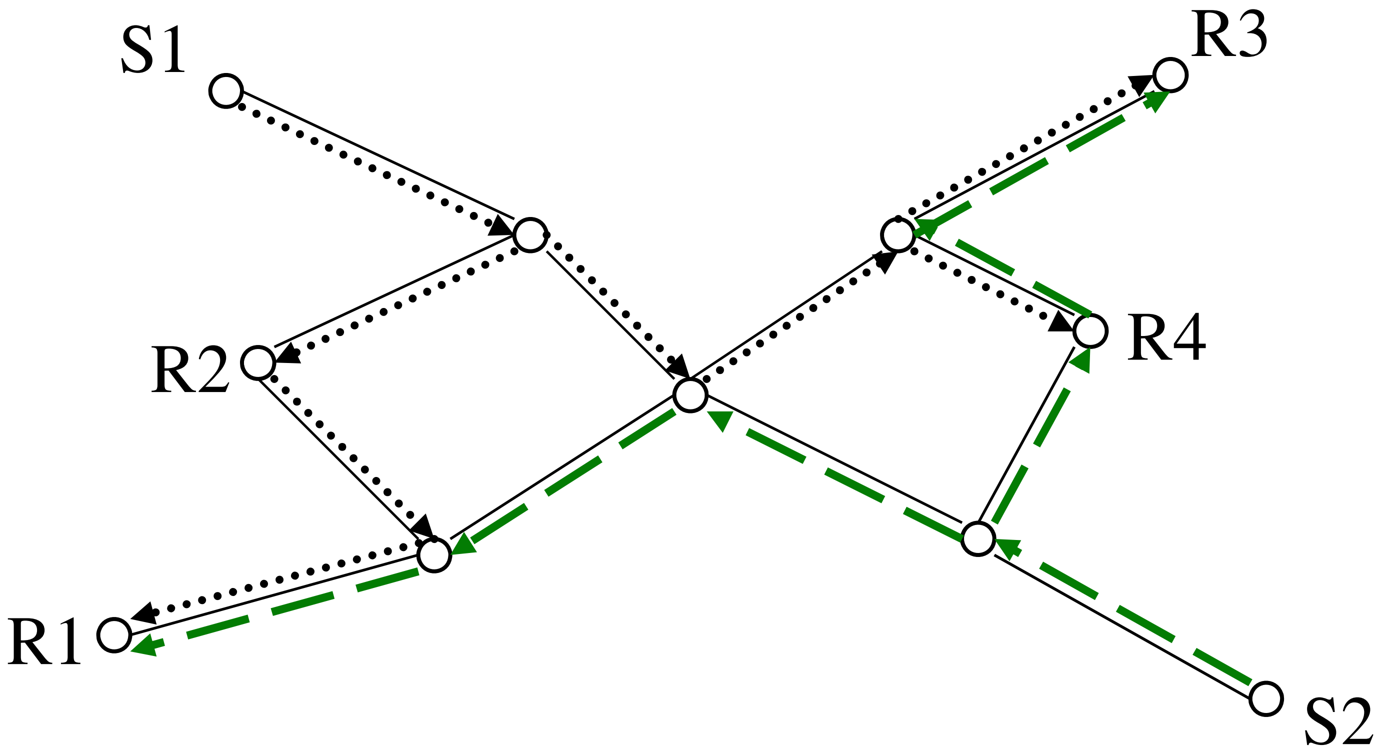
# Basics for Multicast trees

- Destinationen is GROUP
- One and only one copy of each datagram to each group member
- No datagrams to non members
- Avoid looping datagrams
- Optimal paths source – members (*shortest path*)

# Multicast Routing protocols

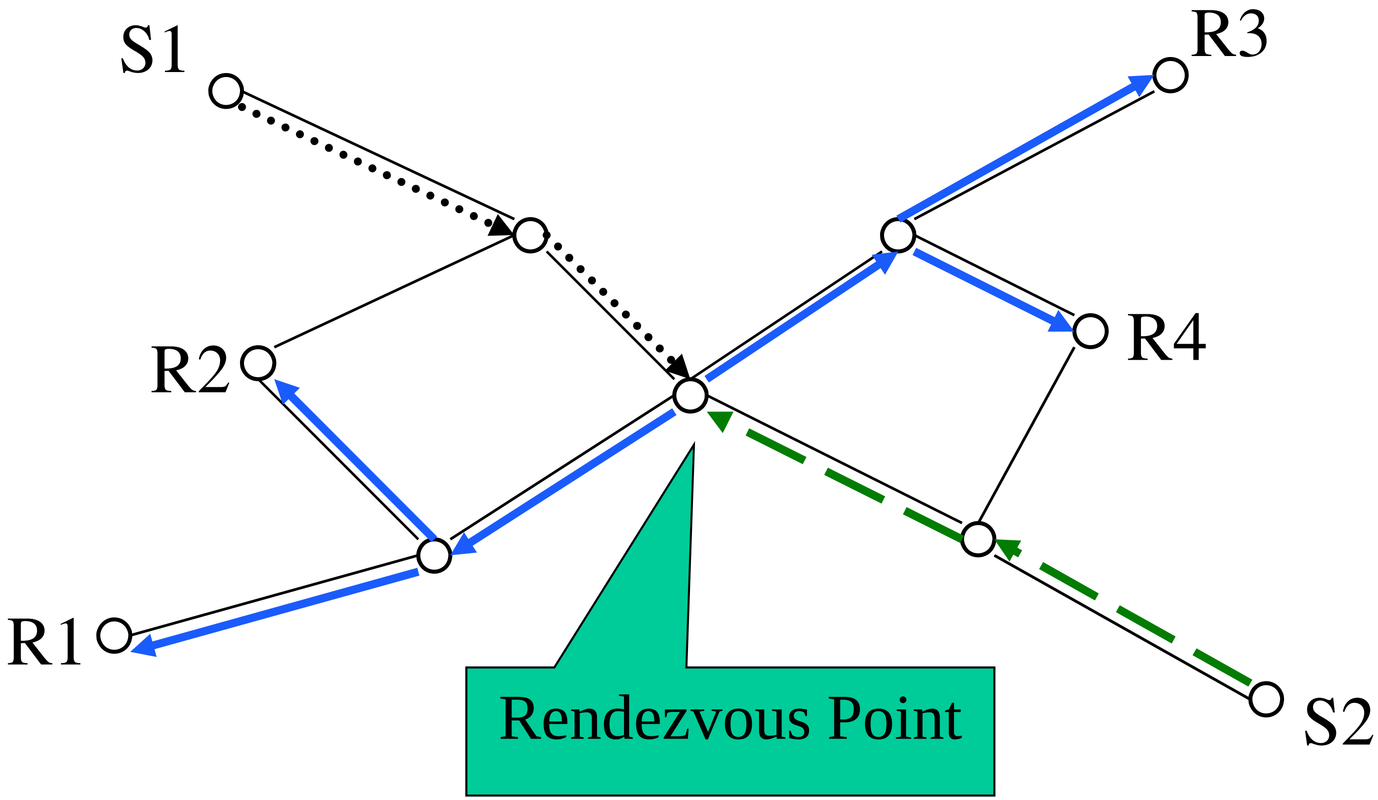


# Source Based Tree



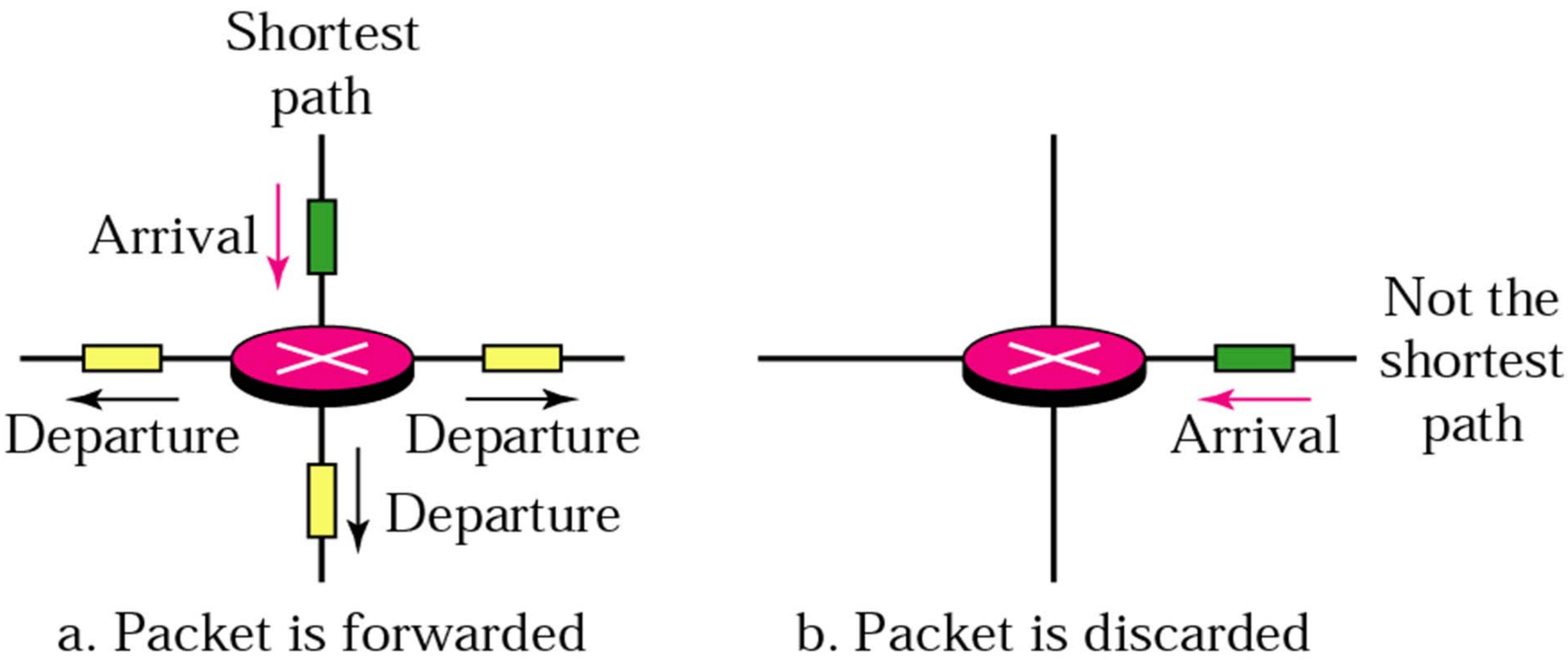

one tree per source

# Group Shared Tree




one tree per group

# Reverse Path Forwarding




Routing per source address!