# COMP9332
# Network Routing & Switching

# OSPF

www.cse.unsw.edu.au/~cs9332

# Recap

- Routing algorithm versus routing protocol
- Types of routing algorithms
  - Distance vector
  - Path vector
  - Link state
- Organization of Internet routing
  - Autonomous systems
  - Interior gateway protocol (IGP)
  - Exterior gateway protocol (EGP)

# Internet routing protocols

Three standardized Internet routing protocols:

| Routing protocol | Based on routing algorithm | IGP/EGP |
|---|---|---|
| Routing Information Protocol (RIP) | Distance vector | IGP |
| Open shortest path first (OSPF) | Link state | IGP |
| Border Gateway Protocol (BGP) | Path vector | EGP |

Note: there are also propriety routing protocols

# This lecture

- Routing algorithm: Link state
- Open shortest path first (OSPF)
  - Link state routing protocol
- Advanced topic:
  - Intra-domain traffic engineering using OSPF

# *RIP or Distance Vector Limitations*

- Three main limitations:
  - Network diameter limited to 15
  - No alternative to shortest path (load balancing not possible)
  - Slow convergence
- Link state routing (or OSPF) removes all these limitations, plus it supports *quality of service*

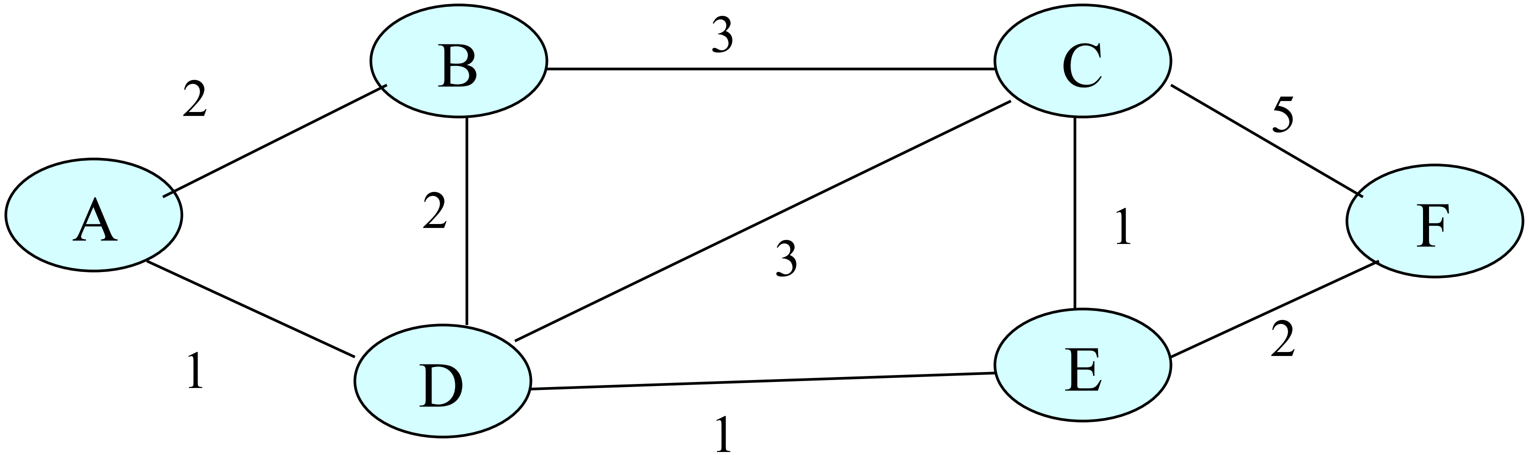
# Link state routing

- The network is given by a graph
  - A set of nodes
  - A set of edges/links
- A different cost can be assigned to each link
- The aim of link state routing is to find the least cost path or shortest path

# *Graph derivation from link state database*

# Link State

- "Link state" refers to whether a link is up or down
  - Convention: Finite cost for a link which is up




Link states of router D are:

D → A cost = 1
D → B cost = 2
D → C cost = 3
D → E cost = 1

# Key ideas behind link state routing (1)

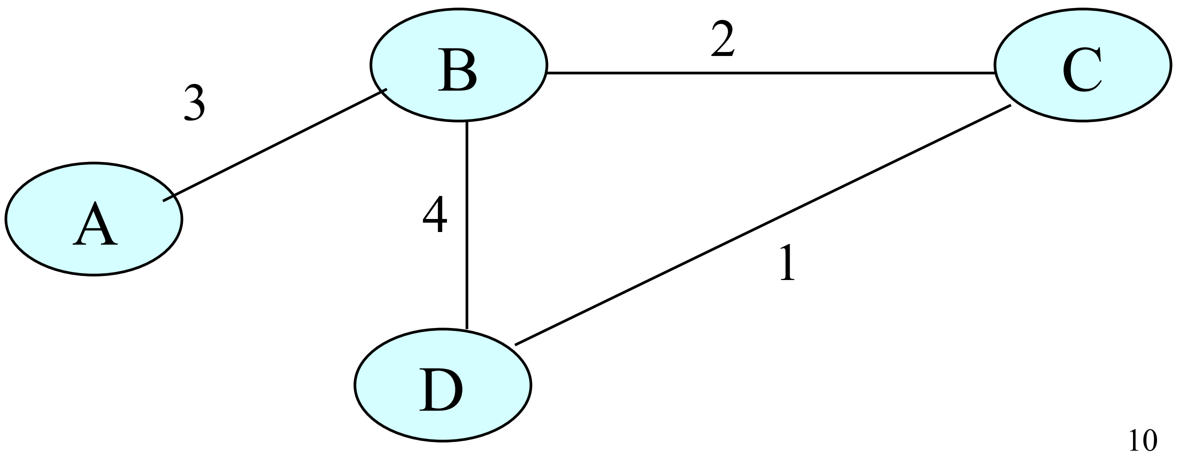
- The link states of each router is distributed throughout the network
  - As a result, each router has the complete topology of the network
- Exercise: Given the following link states:
  - A↔B, cost = 3
  - B↔C, cost = 2
  - C↔D, cost = 1
  - D↔B, cost = 4

  Can you draw the graph for the network?

# Solution

- The given link states are:
  - A↔B, cost = 3
  - B↔C, cost = 2
  - C↔D, cost = 1
  - D↔B, cost = 4

The graph is:



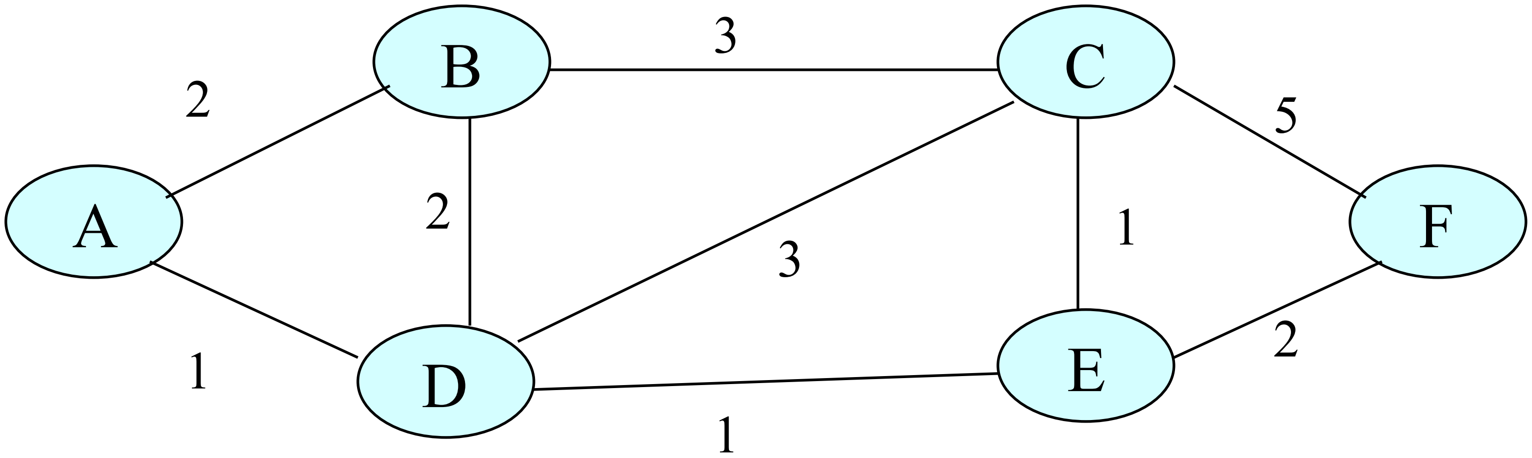
# Computing Shortest Paths

# Key ideas behind link state routing (2)

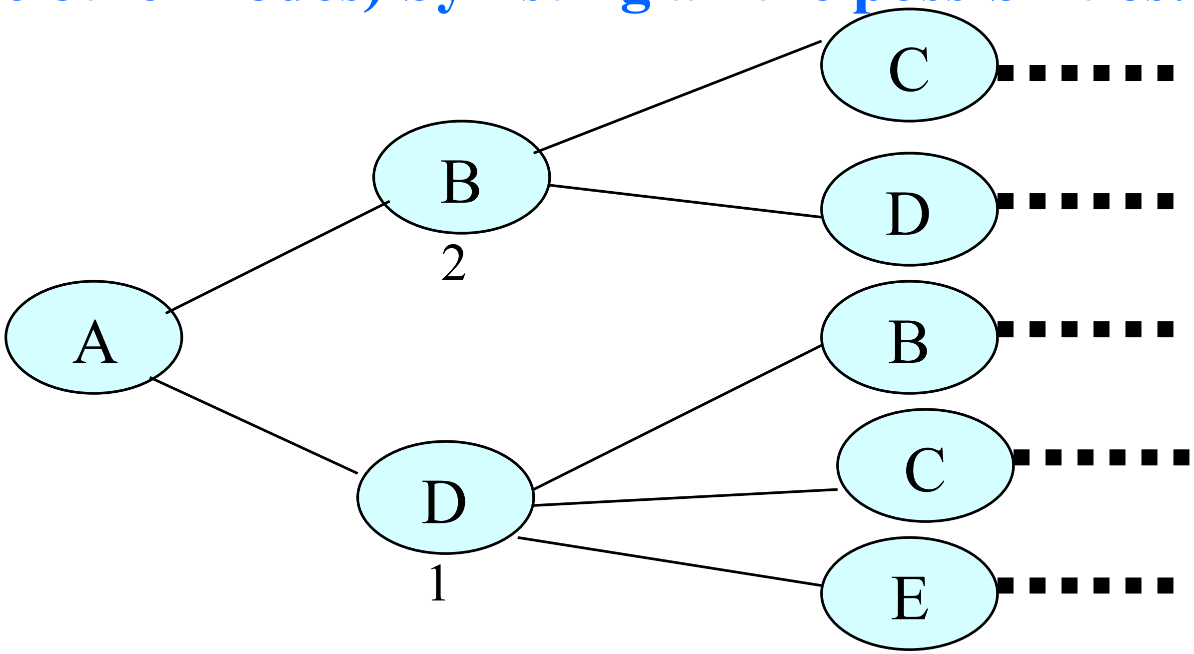
- By using the complete topology information
  - Nodes, Links, Link costs

  Each router computes the shortest path from itself to all the other nodes (destinations)

Given that router A knows the network topology is:




Router A can compute the shortest paths to B, C, D, E and F (all the other nodes) by listing all the possibilities:




Can it be done more efficiently?

# Dijkstra's Algorithm

- Published in 1958
- Given:
  - The graph topology
  - Cost of each link
  - A source node
- Dijkstra's algorithm computes
  - the least cost path from the source to all other nodes

# Dijkstra's Algorithm- summary

- It finds these shortest paths in order of increasing cost
- At each iteration
  - A shortest path is determined
  - Generates more candidate paths
    - » One hop extension from the shortest path just found
  - Hopeless candidates are removed

# *Dijkstra's algorithm - notation*

- $C(i,j)$ : link cost from node *i* to *j*
- $D(v)$ : cost of the path from the source to destination v as of this iteration of the algorithm
  - = the cost of the current least cost path to v
- $p(v)$ : previous node (neighbour of *v*) along the current least-cost path from source to *v*
- N: set of nodes whose least-cost path from source is definitely known
- Source node = *A*

# Dijkstra's Algorithm (pseudocode)

```
N={A}
for all nodes v
        if v is adjacent to A
                then D(v) = c(A,v)
                     p(v) = A
        else D(v) = infinity
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))
        If the value of D(v) has changed, set p(v) = w
Until all nodes in N
```

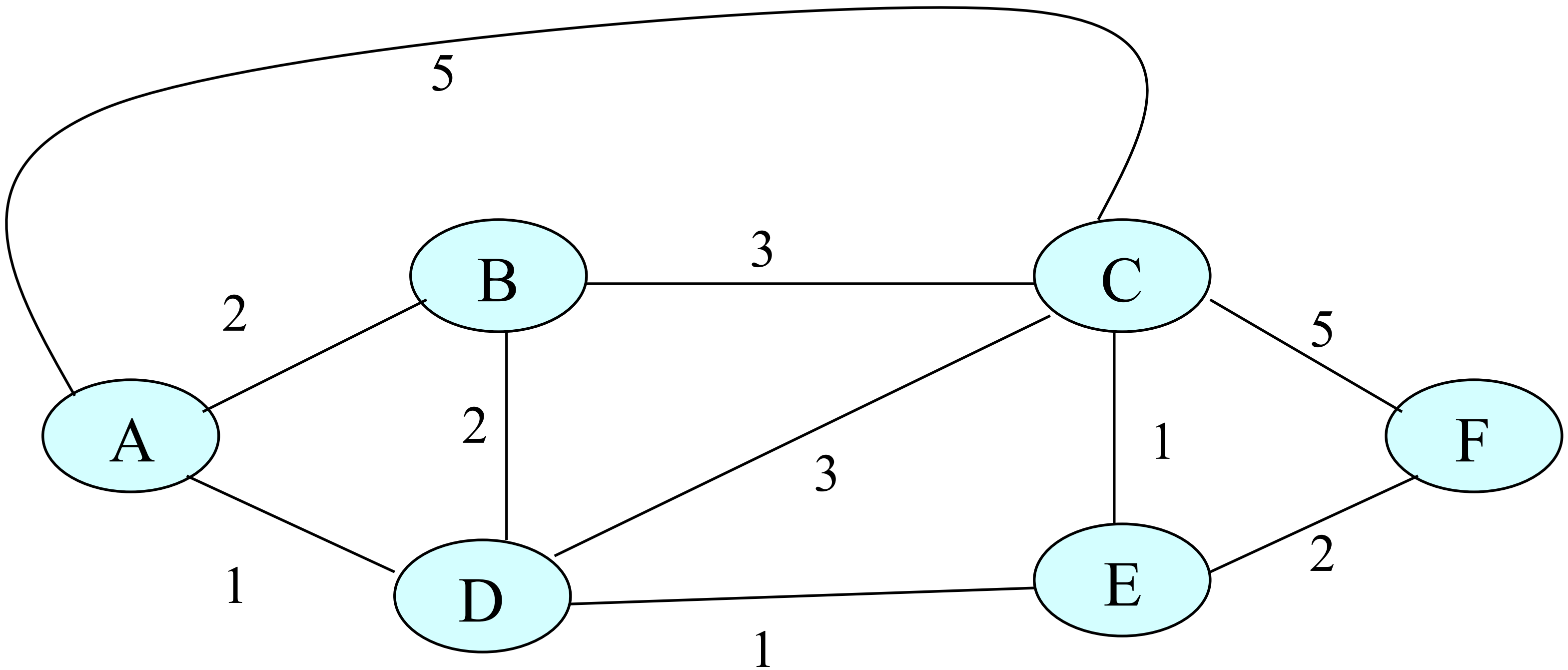
Initialisation

Loop

# Exercise

- Consider the network shown on the next slide
- Compute the least-cost path from A to all possible destinations using Dijkstra's algorithm

# Example Network Topology



Step 0: Initialisation
Fill in the table for step 0

| Step | N | D(B),p(B) | D(C),p(C) | D(D),p(D) | D(E),p(E) | D(F),p(F) |
|---|---|---|---|---|---|---|
| 0 | | | | | | |
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |

Loop the first time:

- 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
  by D(v) = min( D(*v*), D(*w*) + c(*w*,*v*) )
  If the value of D(v) has changed, set p(v) = w

Exercise: Complete step 1

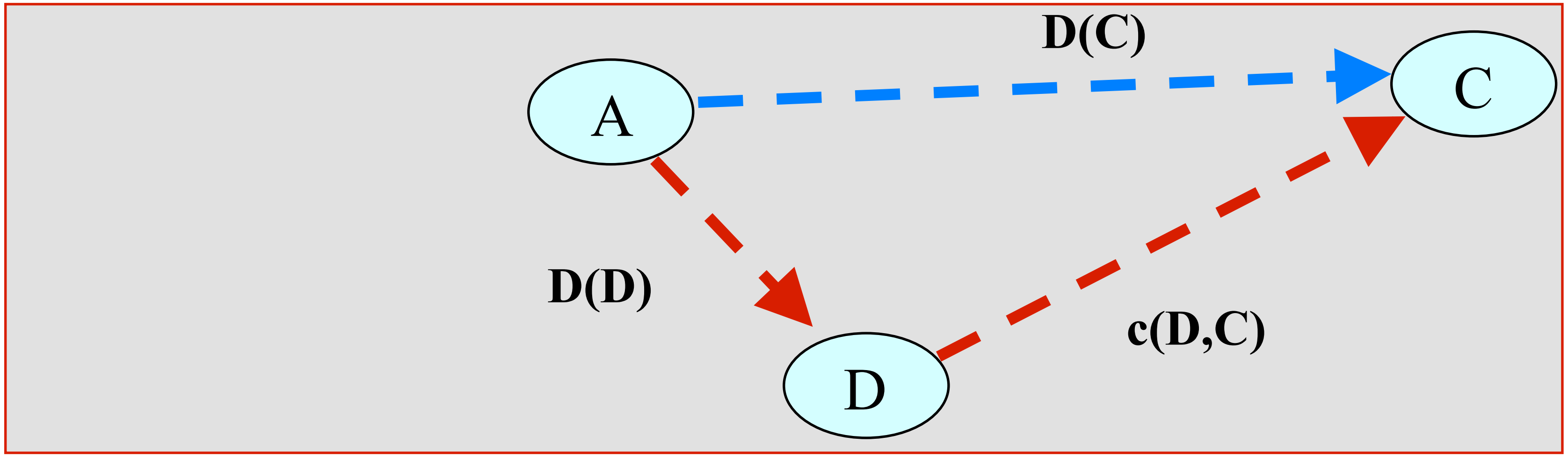
- What is *w* for this iteration?
- What are the nodes that are adjacent to w and not in N?

| Step | N | D(B),p(B) | D(C),p(C) | D(D),p(D) | D(E),p(E) | D(F),p(F) |
|---|---|---|---|---|---|---|
| 0 | A | 2,A | 5,A | 1,A | inf | inf |
| 1 | | | | | | |

- **w = D**
- **The 1st shortest paths known is A→D with cost 1**
- **In this step, we are sure that we have found the shortest path to w (which is D here)**

| Step | N | D(B),p(B) | D(C),p(C) | D(D),p(D) | D(E),p(E) | D(F),p(F) |
|---|---|---|---|---|---|---|
| 0 | A | 2,A | 5,A | 1,A | inf | inf |
| 1 | AD | 2,A | 4,D | 1,A | 2,D | inf |

- Nodes adjacent to w (=D) but not in N = {B,C,E}
- Generate candidate paths A → w (= D) → ?
- Also eliminate hopeless candidates: for C, we compare
  - D(C) = 5 (A → … → C) and
  - D(D) + c(D,C) = 1 + 3 = 4 (A → … → D → C)
- Update D(C)=4 and p(C) = D




| Step | N | D(B),p(B) | D(C),p(C) | D(D),p(D) | D(E),p(E) | D(F),p(F) |
|---|---|---|---|---|---|---|
| 0 | A | 2,A | 5,A | 1,A | inf | inf |
| 1 | AD | 2,A | 4,D | 1,A | 2,D | inf |

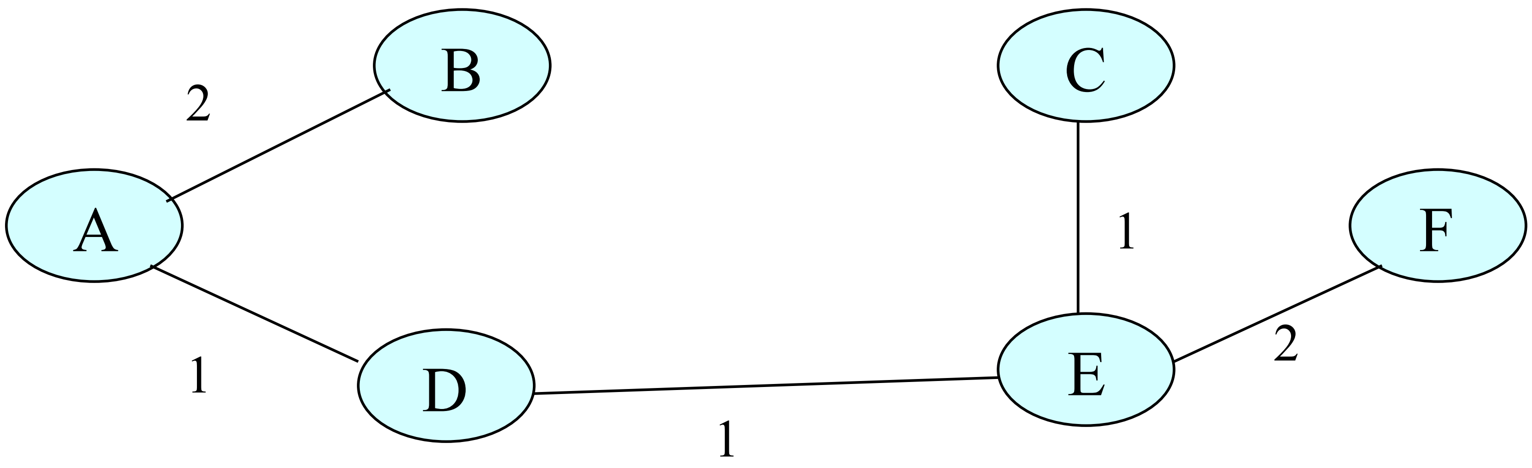
# The final result

| Step | N | D(B),p(B) | D(C),p(C) | D(D),p(D) | D(E),p(E) | D(F),p(F) |
|---|---|---|---|---|---|---|
| 0 | A | 2,A | 5,A | 1,A | inf | inf |
| 1 | AD | 2,A | 4,D | 1,A | 2,D | inf |
| 2 | ADE | 2,A | 3,E | 1,A | 2,D | 4,E |
| 3 | ADEB | 2,A | 3,E | 1,A | 2,D | 4,E |
| 4 | ADEBC | 2,A | 3,E | 1,A | 2,D | 4,E |
| 5 | ADEBCF | 2,A | 3,E | 1,A | 2,D | 4,E |

# Interpretation

| Step | N | D(B),p(B) | D(C),p(C) | D(D),p(D) | D(E),p(E) | D(F),p(F) |
|---|---|---|---|---|---|---|
| 5 | ADEBCF | 2,A | 3,E | 1,A | 2,D | 4,E |

**The shortest path tree rooted at source A**:



# Routing Table of A

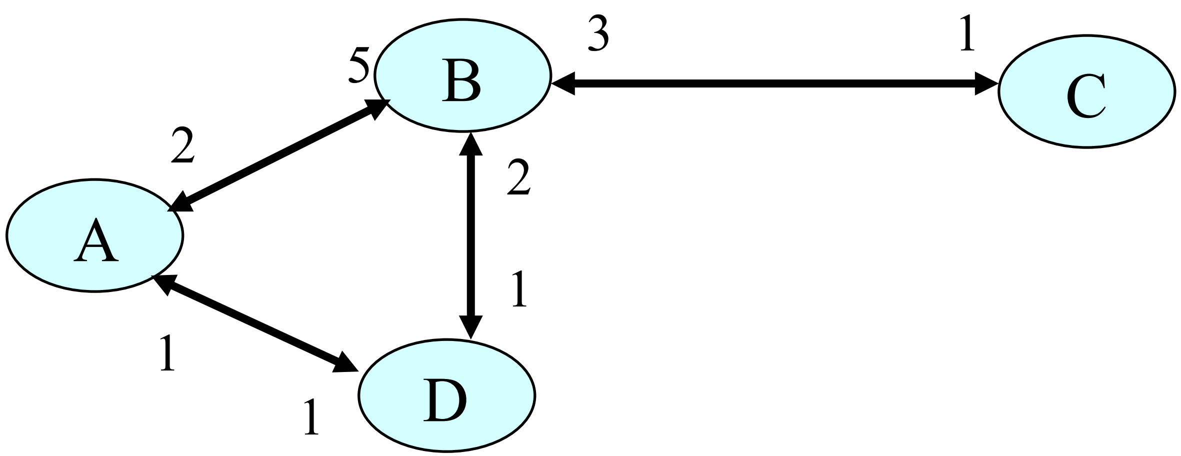
| Destination | Cost | Next Hop |
|---|---|---|
| B | 2 | Direct |
| C | 3 | D |
| D | 1 | Direct |
| E | 2 | D |
| F | 4 | D |

# The order in which the shortest paths are obtained

| Step | N | D(B),p(B) | D(C),p(C) | D(D),p(D) | D(E),p(E) | D(F),p(F) |
|---|---|---|---|---|---|---|
| 0 | A | 2,A | 5,A | 1,A | inf | inf |
| 1 | AD | 2,A | 4,D | 1,A | 2,D | inf |
| 2 | ADE | 2,A | 3,E | 1,A | 2,D | 4,E |
| 3 | ADEB | 2,A | 3,E | 1,A | 2,D | 4,E |
| 4 | ADEBC | 2,A | 3,E | 1,A | 2,D | 4,E |
| 5 | ADEBCF | 2,A | 3,E | 1,A | 2,D | 4,E |

# Undirected versus directed graph

- Dijkstra's algorithm can also be applied to directed graph where cost in both directions are different



# OSPF Routing Protocol

# Open shortest path first (OSPF)

- Developed in the late 1980s to overcome the limitations of RIP
  - What are the limitations of RIP?
- Open = non-proprietary
- An implementation of link state algorithm
  - Link state – a generic routing algorithm
  - OSPF – a specific routing standard

# OSPF Features

- Can simplify physical topology to less complex logical topology (*adjacency reduction*)
- Can divide the whole network into several smaller networks (the AREA concept)
- Supports different Type of Service (TOS)
  - *A* link has multiple link costs
    - » Link cost for TOS #1
    - » Link cost for TOS #2
  - This features helps supporting *quality of service*, but usually not used though
- Unlike RIP/BGP, it does not use TCP/UDP
  - OSPF sends its messages directly over IP

# Key ideas behind OSPF (1)

- Each router maintains its link states and distributes them throughout the network
  - As a result, each router has the complete topology of the network
- Each router computes the shortest path to all destinations by applying Dijkstra's algorithm to the topology knowledge that it has

# Key ideas behind OSPF (2)

- When a router detects a change in link state, it sends out a link state advertisement which is distributed throughout the network
  - So that each router has the up-do-date network topology
- Each router re-computes the shortest paths using the updated topology knowledge

# Key components of OSPF

- A router needs to know its link state (needs to know its neighbours)
  - Greeting neighbours with HELLO message
- A method to distribute link states (neighbour connectivity information) throughout the network (flooding)
- An algorithm to compute the shortest path to all destinations
  - Dijkstra's algorithm

# HELLO protocol

# 2-way communication (adjacency)

- Each router sends out a HELLO packet at regular interval identifying itself
  - HELLO messages are sent to AllSPFRouters Multicast group address 224.0.05
- HELLO also includes a list of neighbours from which the router has received a HELLO
- 2-way communication (adjacency) with a neighbour is established when a router finds its ID in the HELLO sent out from that neighbour
  - Each adjacency is a link in the topology graph
- Once adjacency is established, both neighbours synchronise their link state databases (more on this later)

# Maintaining link state

- If a router has not heard from its neighbour for a long time, it declares that neighbour dead (the adjacency no longer exists)
- The adjacency may be established again in the future if appropriate HELLO messages are received
- Change in link state (Up → Down, Down → Up) is propagated throughout the network by means of *flooding*

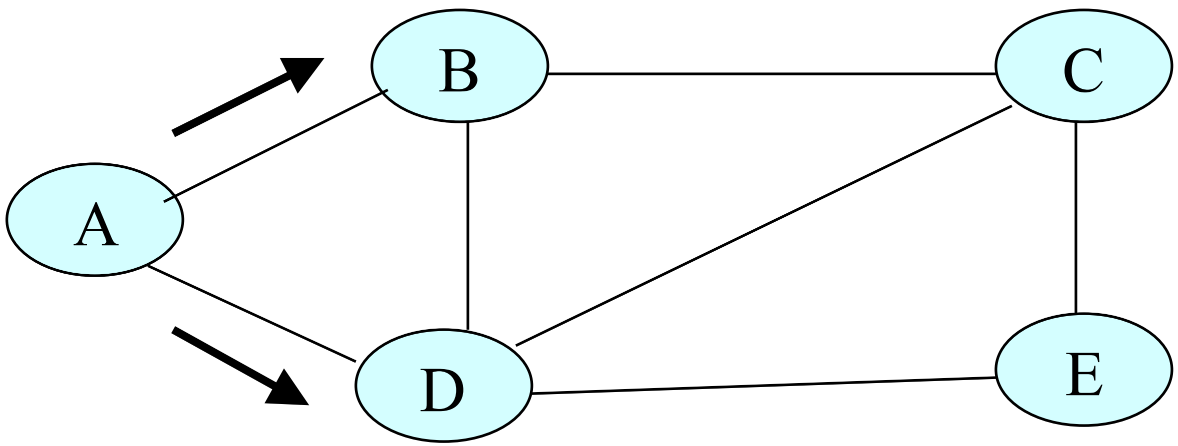
# *Flooding the link states*

# Flooding (1)

- A method to distribute a message throughout the entire network
- Rule 1: If you haven't received the message before, send it to all your neighbours except the one that you receive the message from
- Rule 2: If you have received the message before, discard the message
  - Need a method to check whether it has received the message before

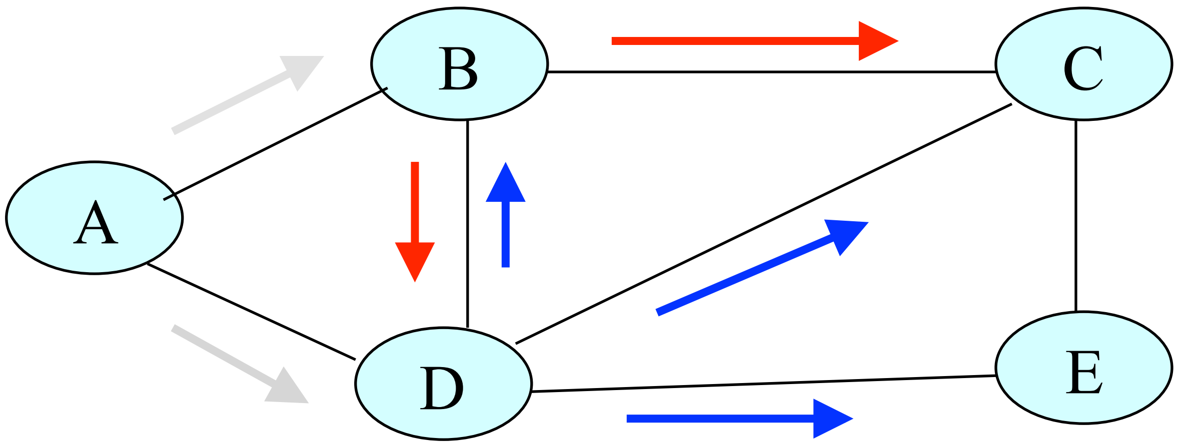
# Flooding - example (1)

- A initiates flooding and sends the message to its neighbours, i.e., B and D



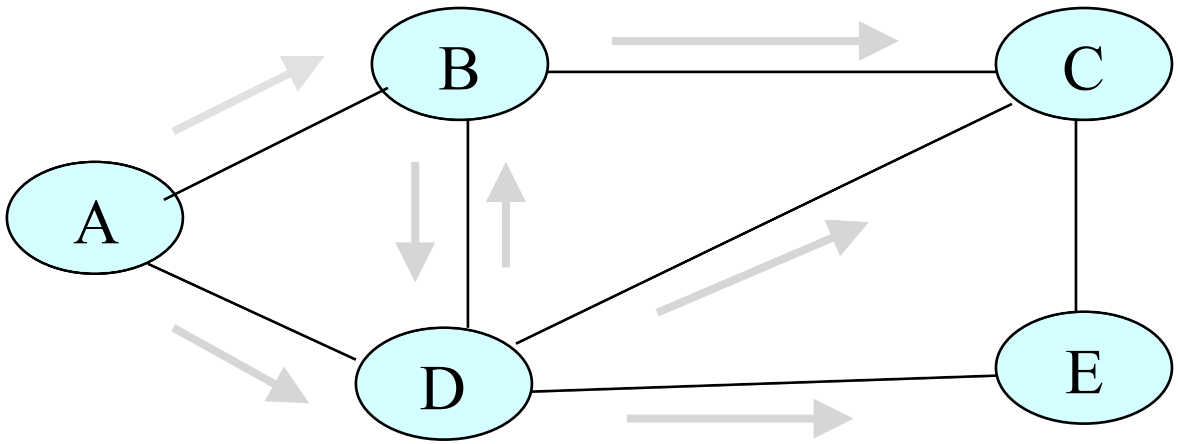
# Flooding - example (2)

- B forwards the message to C and D (but not A)
- D forwards the message to B, C and E (but not A)
- Duplicate messages received at B and D are discarded



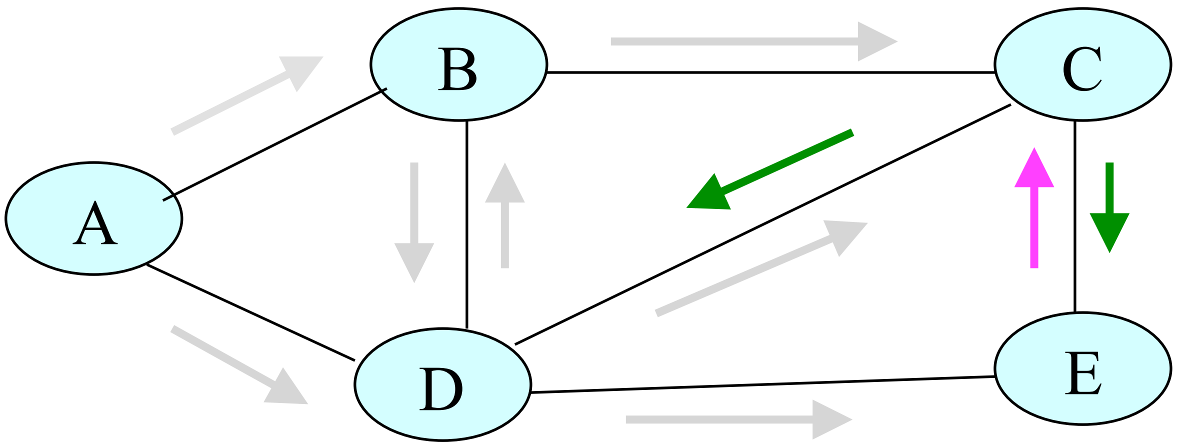
# Flooding - example (3)

- Assume that C receives the message from B first
- Who will C forward this message to? B? D? E?



# Flooding - example (4)

- C receives the message from B before receiving it from D
- C sends the message to D and E; E forwards the message it received from D to C
- Duplicate messages at B, C and E are discarded. Flooding complete.



# Flooding (2)

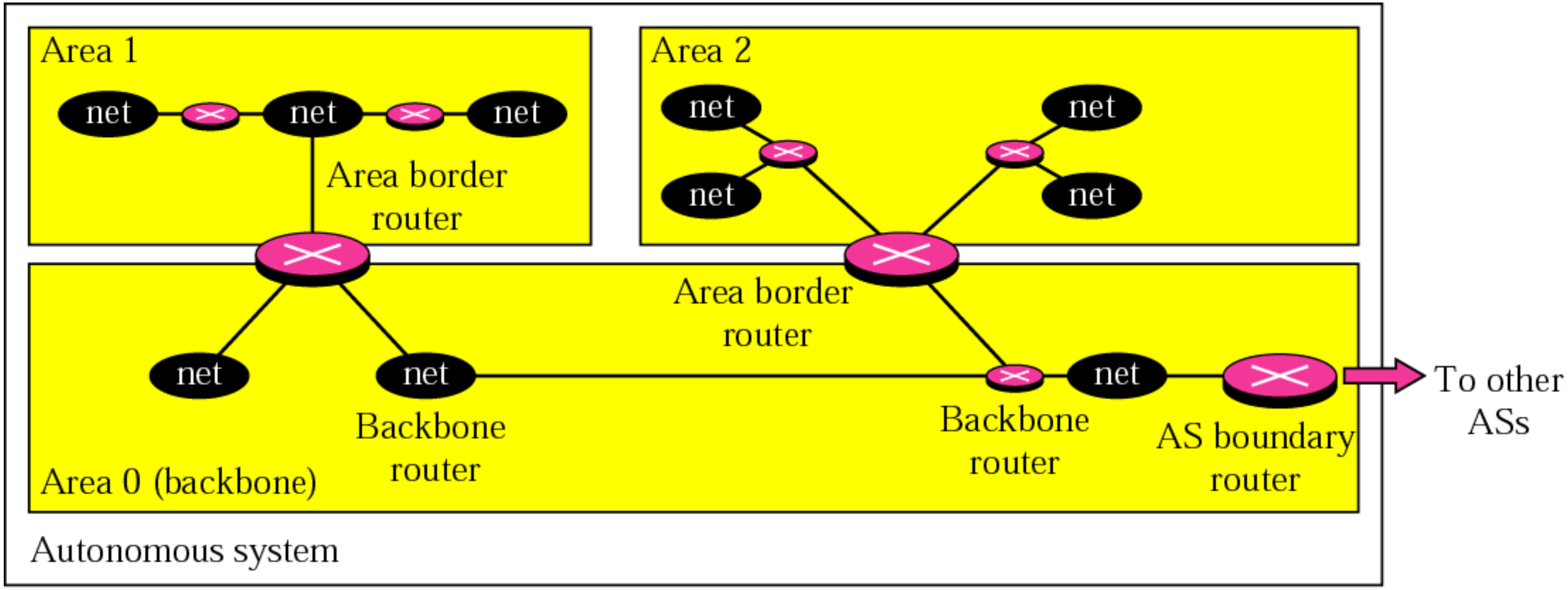
- Message is acknowledged
  - This provides reliability
  - Why doesn't OSPF use TCP for reliability?
- Duplicate detection
  - The link state advertisement (LSA) message header contains a sequence number
  - If a router receives two LSA messages which have the same sequence number in their headers, are they necessarily duplicates?

# The AREA Concept

# *Achieving Scalability using Routing Areas*

- An autonomous system is divided into smaller OSPF routing areas
  - Each area is identified by a 32-bit area ID
- The link state messages are only flooded within an area
- There's always a backbone area (also known as area zero)
- All other areas are connected to the backbone area using *area border routers*
- Knowledge of an area's topology is hidden from the routers in other areas

# Areas in an autonomous system



# OSPF router types

- Intra-area routers
  - Maintain only topology within its area
- Border area routers
  - Connect multiple areas – one of which must be the backbone
  - Maintain separate topology database and routing table for each attached area
- AS boundary routers
  - Responsible for announcing external link information within the AS

# Why OSPF areas?

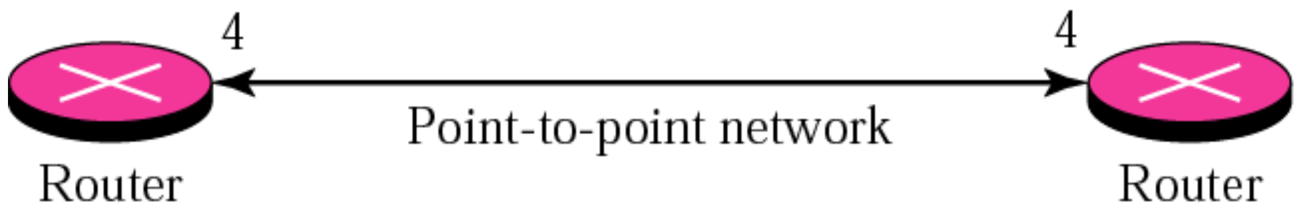
- Reduce the size of topology database
  - Each area only knows the topology of its area
- Reduce the number of link state updates
  - Link states do not propagate across areas
- Reduce the amount of processing
  - Less link state messages to process
  - Smaller topology → faster route computation

# Link state advertisement (LSA)

- OSPF defines different types of link state advertisements (LSAs) depending on the location of the router or network
- We first need to learn about different link types in OSPF
  - Point-to-point link
  - Transient link
  - Stub link
  - Virtual link - but we won't look at them

# Point-to-point link

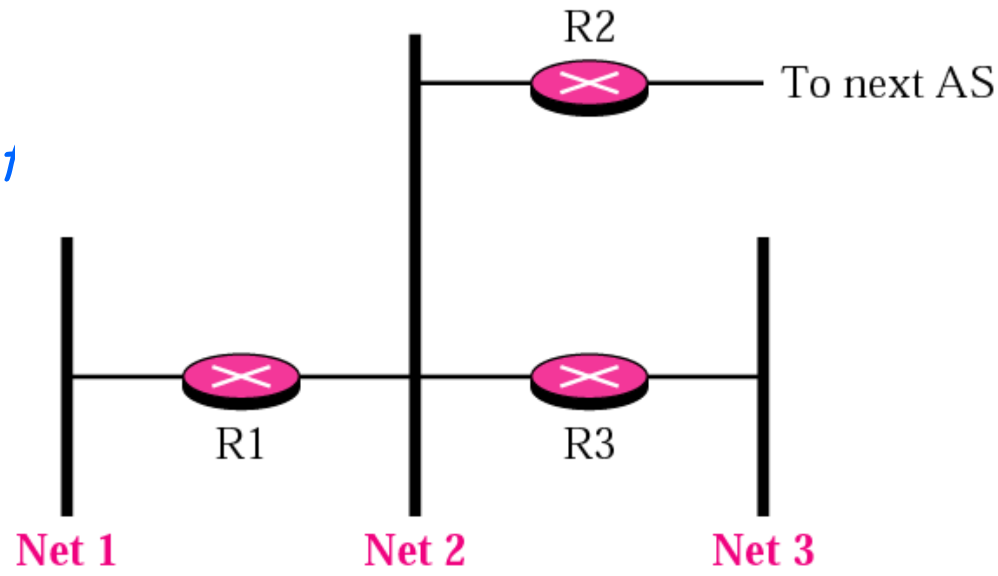
- Point-to-point network directly connects two routers




**Note: The links are directed.**

# Transient link

- A transient link is a network with N (>1) routers attached to it (*Net 2 is an example of transient links*)



# Adjacency reduction using designated routers (DRs)

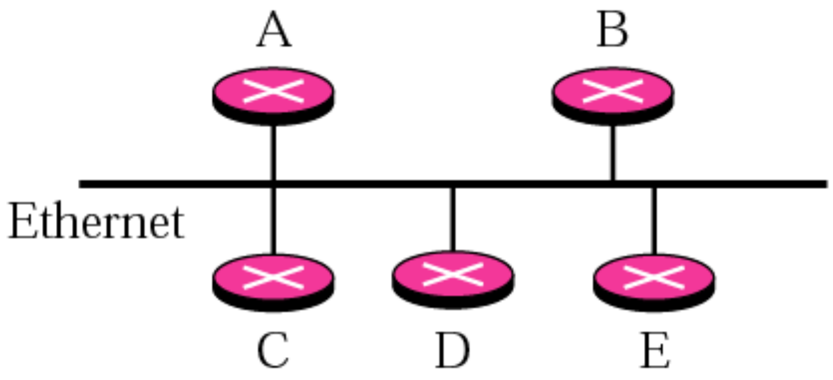
# Physical Topology without DR

- Let N = number of routers attached to a transient link (a broadcast network, such as Ethernet)
- Each of the N routers becomes a neighbour of each other
  - N(N-1)/2 adjacencies!
  - N(N-1)/2 databases to synchronize!
  - Highly complex network topology with many nodes and links in the graph → shortest path computation overhead is high each time there is a change in the database

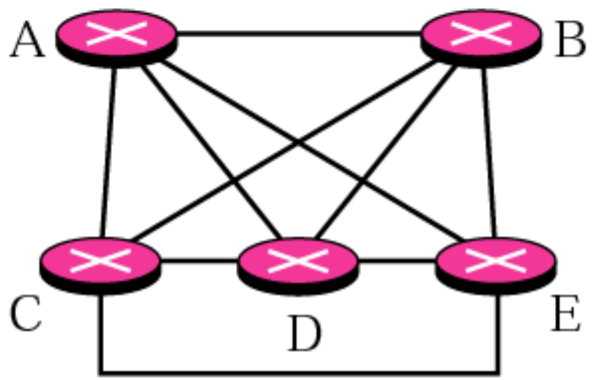
# *Using DR to reduce database complexity*

- One of the routers, called DR, assume the role of the broadcast network
- All routers form adjacencies with the network (DR) in a hub-like topology (only N links)
- Routers do not form adjacencies between themselves (although they are physically connected)
- Thus, we obtain a logical topology, which is simpler (less number of links) than the original physical topology

# Example of a Transient Link with 5 Routers

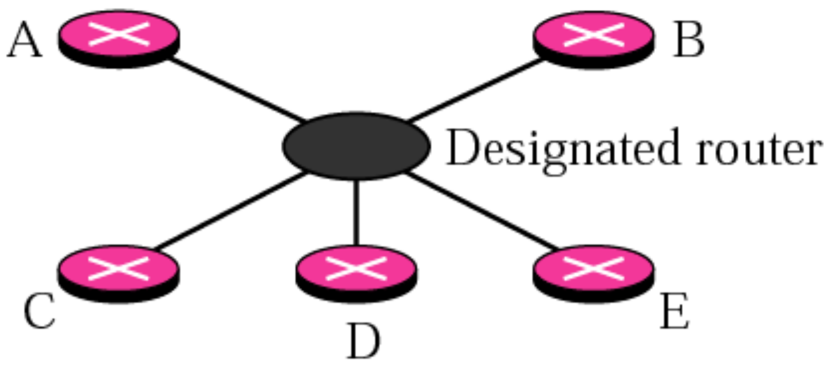



a. Transient network




b. ~~Unrealistic representation~~

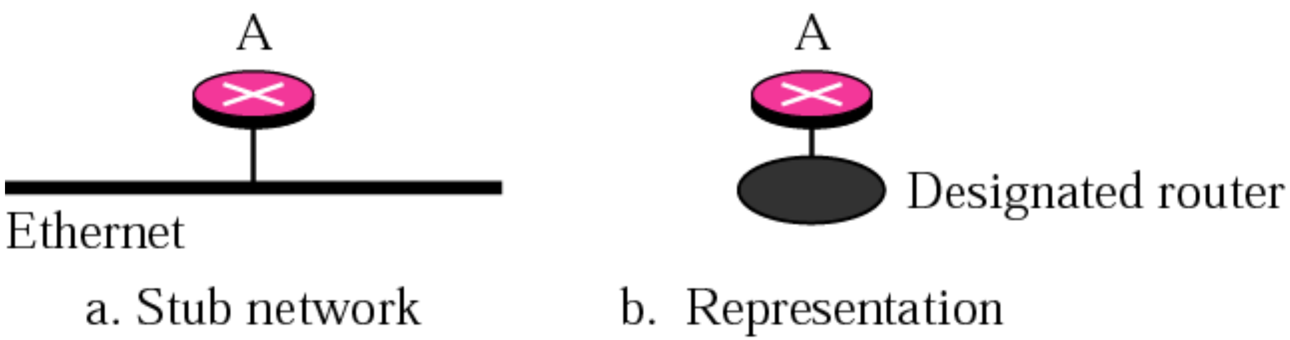
OSPF links without DR
(physical topology)



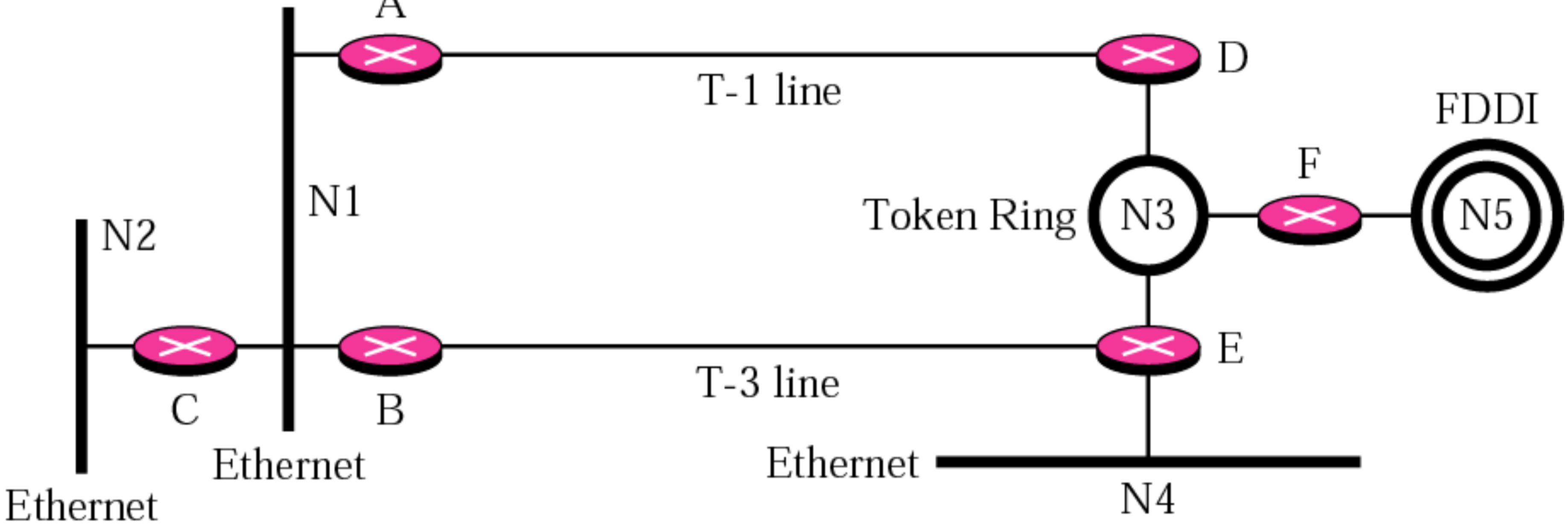

c. ~~Realistic representation~~

OSPF links with DR
(logical topology)

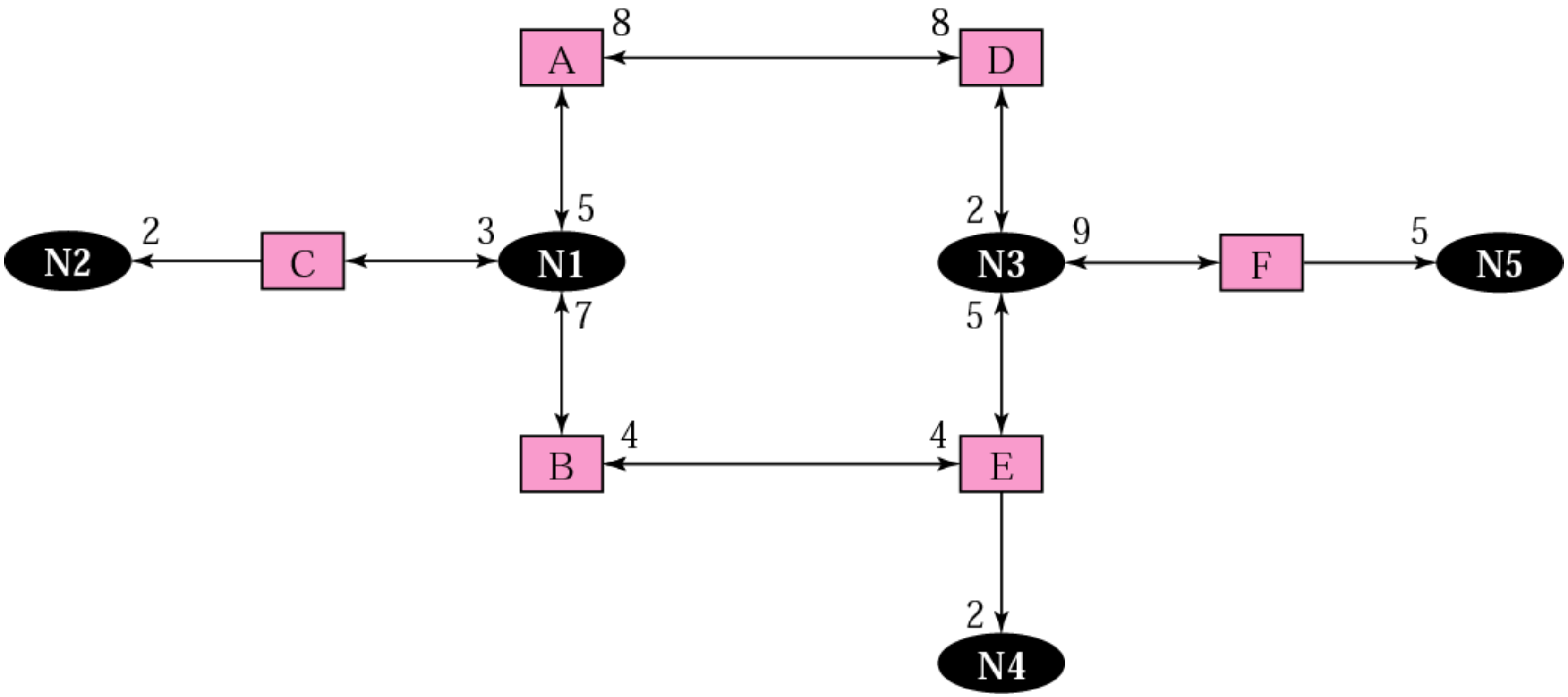
# Stub link


A
Ethernet
a. Stub network
A
Designated router
b. Representation

# Example of an internet



# OSPF representation of an internet





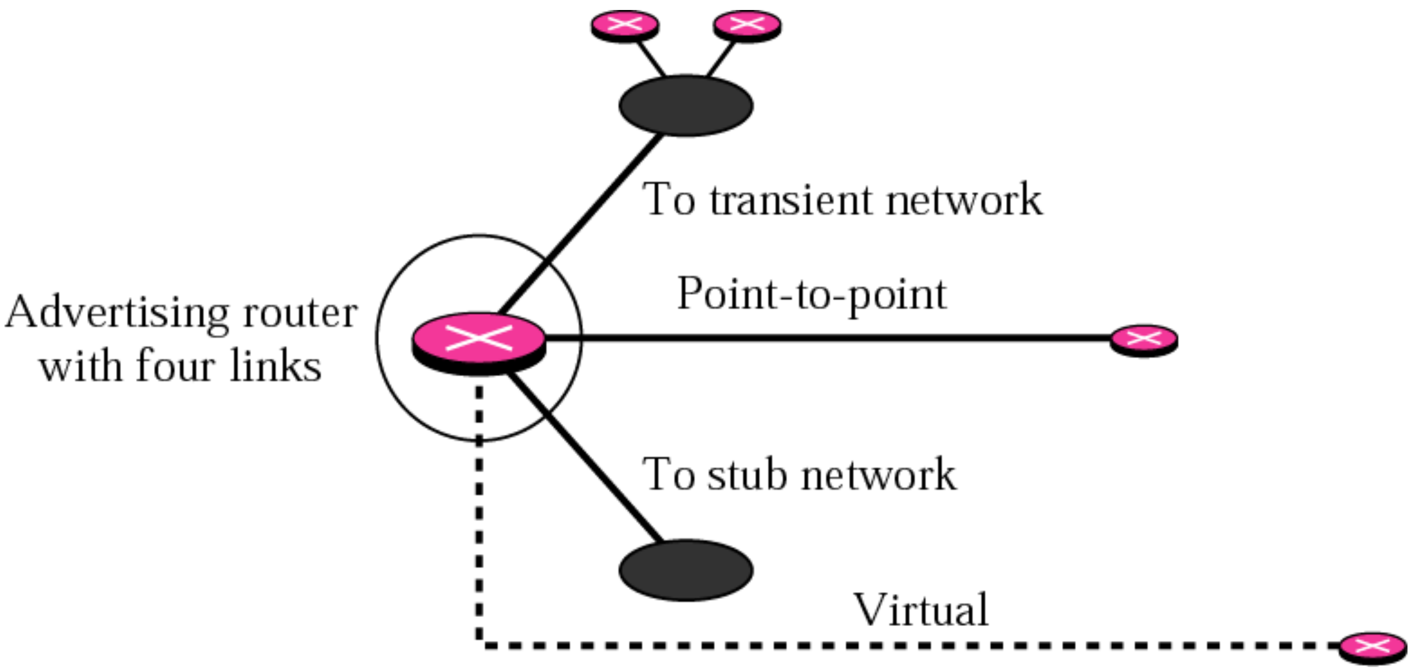

**Note: Network to router link has zero cost.**

# Types of Link State Advertisements (LSAs)

- OSPF defines five types of LSAs
  - Router Link
  - Network Link
  - Summary link to network
  - Summary link to AS boundary routers
  - External link
- A packet format is defined for each type of LSA

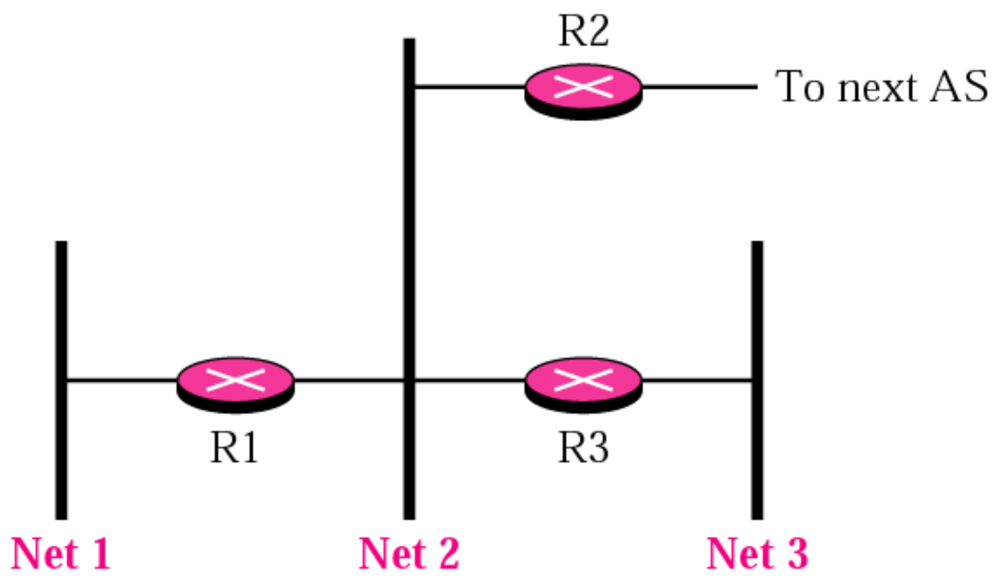
# Router link

- Router link describes the link state from a router to other routers or network



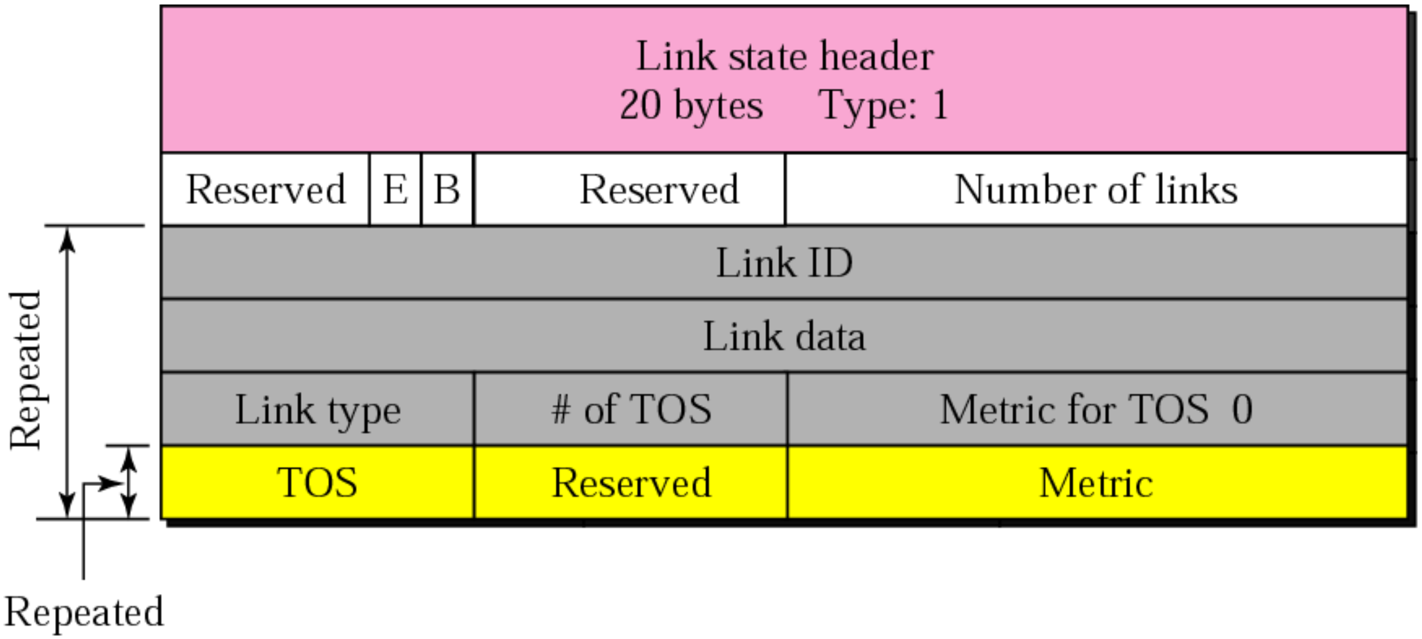
# Router link - example

- All routers advertise router link LSAs
  - R1 has 2 links: Net1 and Net2
  - R2 has 1 link: Net2
  - R3 has 2 links: Net2 and Net3

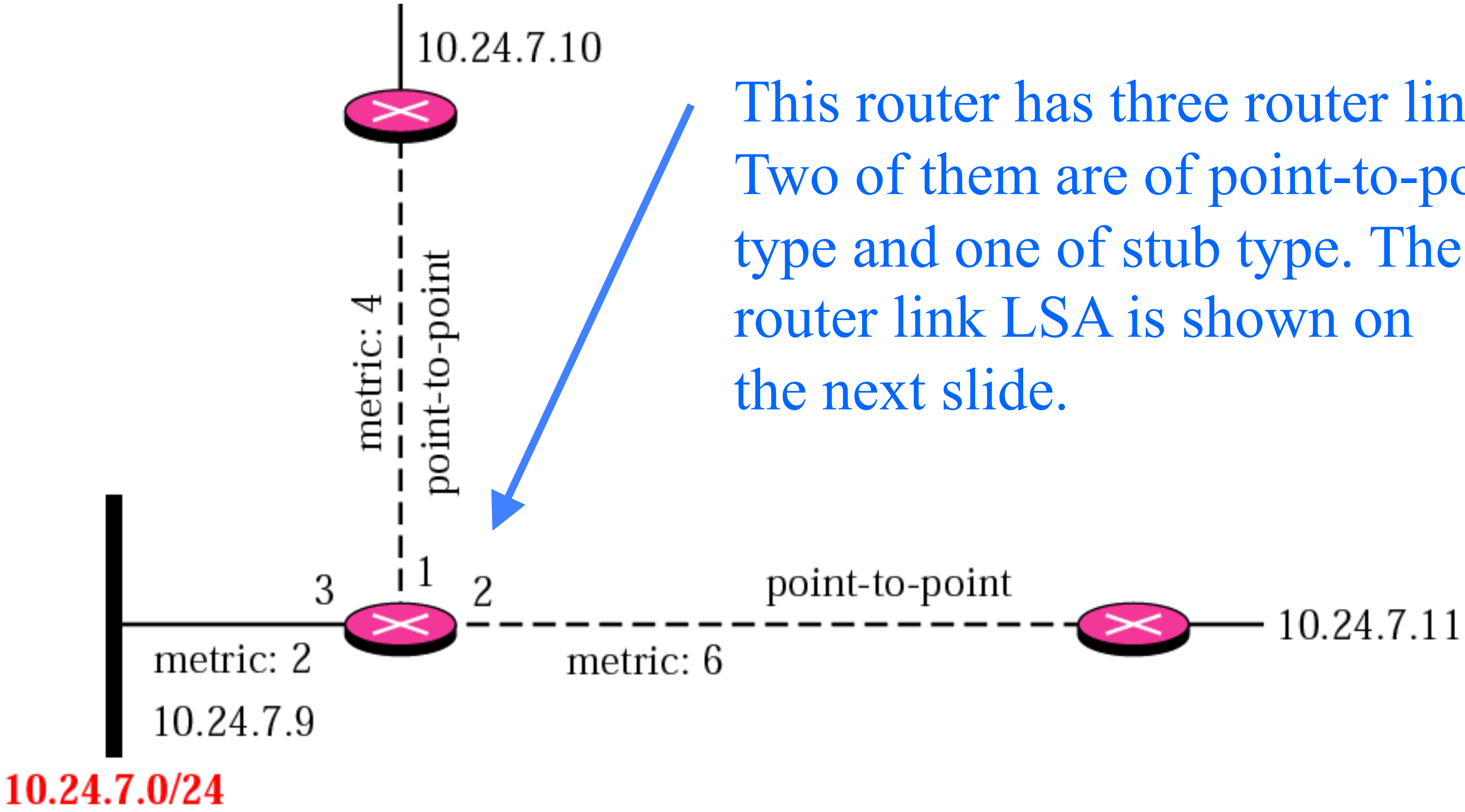


# Router link LSA

- Link ID and Link Data depends on Link Type
  - Type 1: Point-to-point, LinkID=Neighbour router address, LinkData=Interface number
  - Type 3:Stub network, LinkID=Network address, LinkData= Subnet Mask



# Router link LSA - example




This router has three router links. Two of them are of point-to-point type and one of stub type. The router link LSA is shown on the next slide.

# Router link LSA - example (cont'd)

Assumption:
Only 1 TOS.

Note:
The solution in the text assumes multiple TOS

| | | |
|---|---|---|
| OSPF Header Type:4 | | |
| LSA Header Type:1 | | |
| | | 3 |
| 10.24.7.10 | | |
| 1 | | |
| 1 | 1 | 4 |
| 10.24.7.11 | | |
| 2 | | |
| 1 | 1 | 6 |
| 10.24.7.0 | | |
| 255.255.255.0 | | |
| 3 | 1 | 2 |

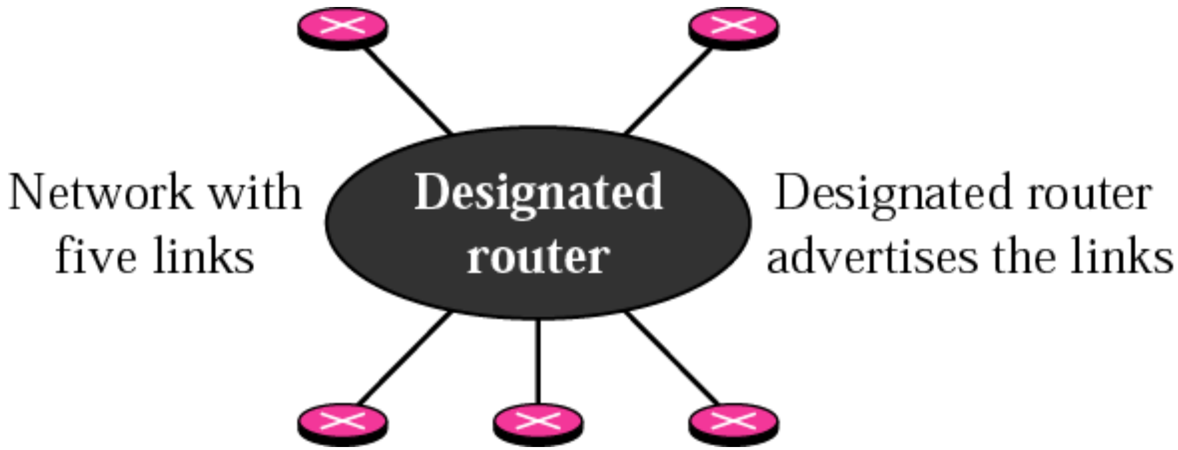
Common for all 3 ads

For interface 1

For interface 2

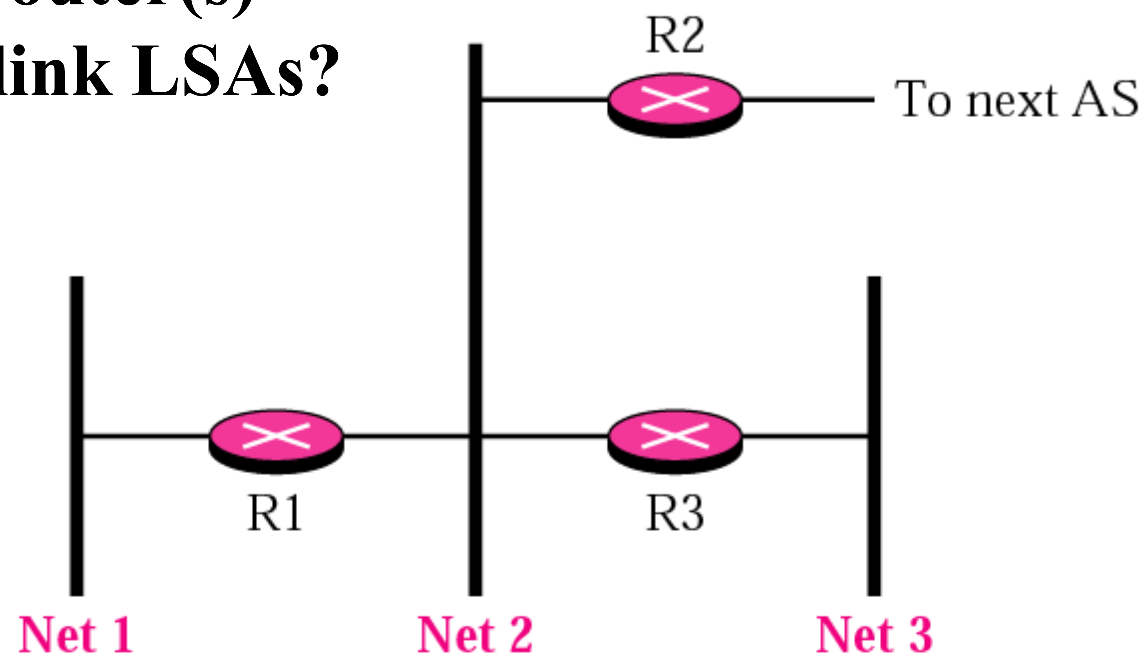
For interface 3

# Network link



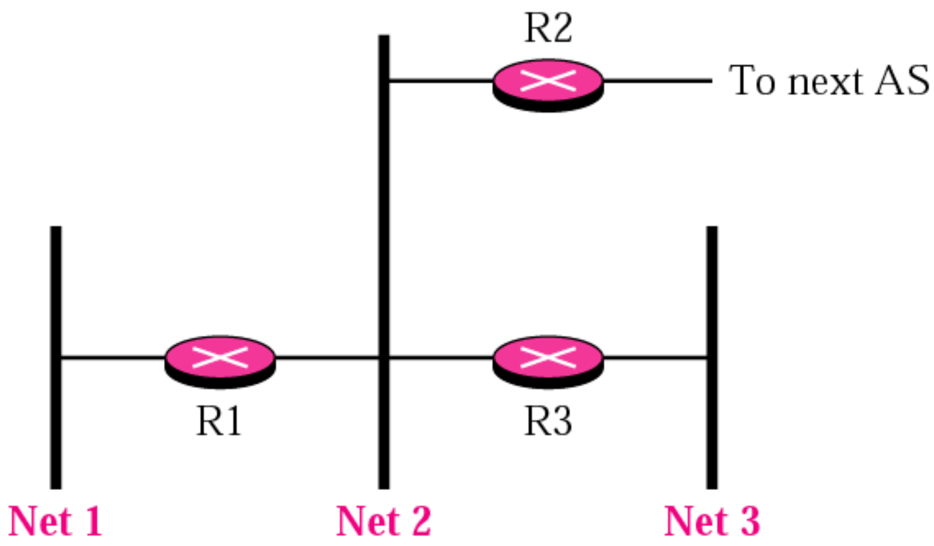
# Network link - example

**Question: Which router(s) send out network link LSAs?**



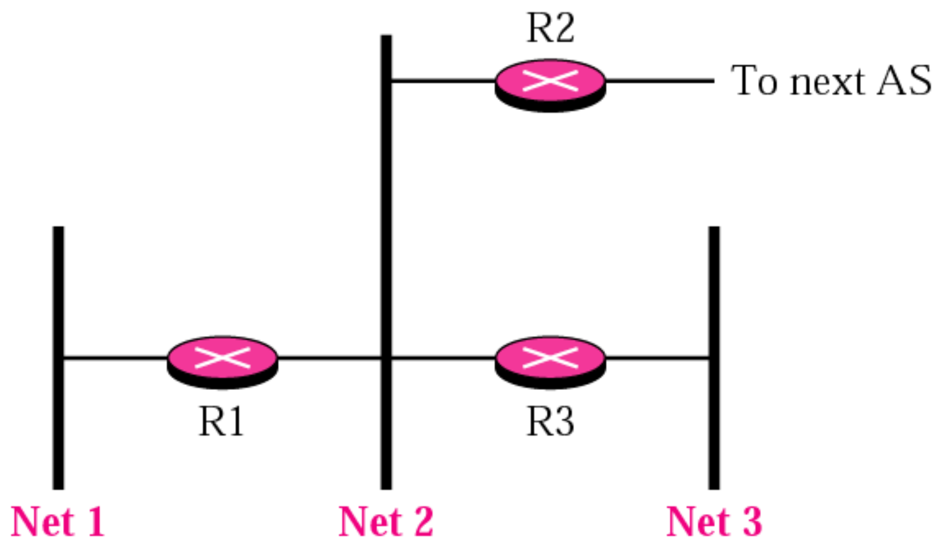
# Network link – example (cont'd)

- All three networks send out network link LSAs
- Advertisement for Net1 is done by R1 because it is the only router attached to the network and therefore the designated router
- Similarly for Net3

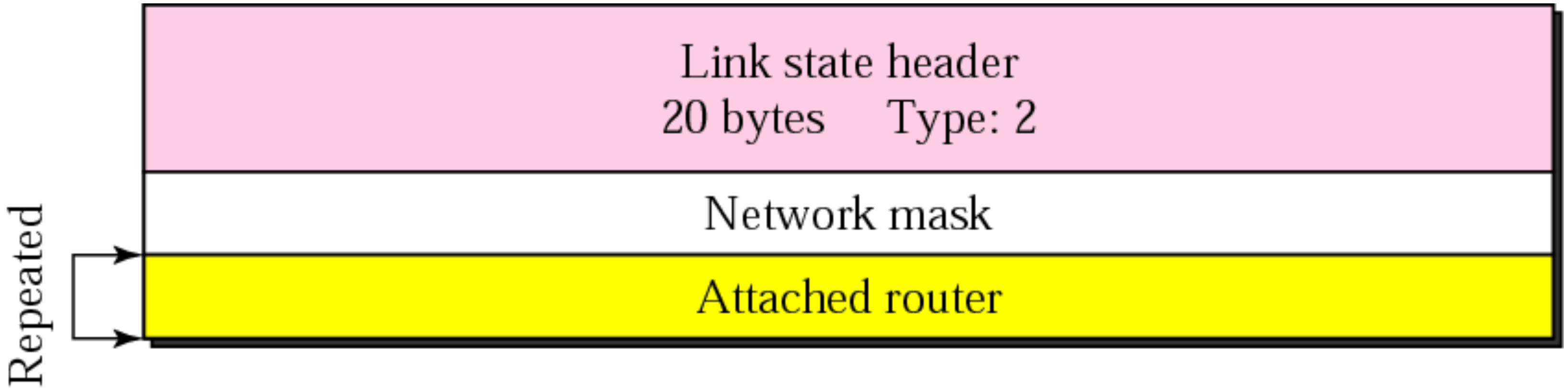


# Network link – example (cont'd)

- Advertisement for Net2 can be done by R1, R2 and R3 depending on which one is chosen as the designated router.

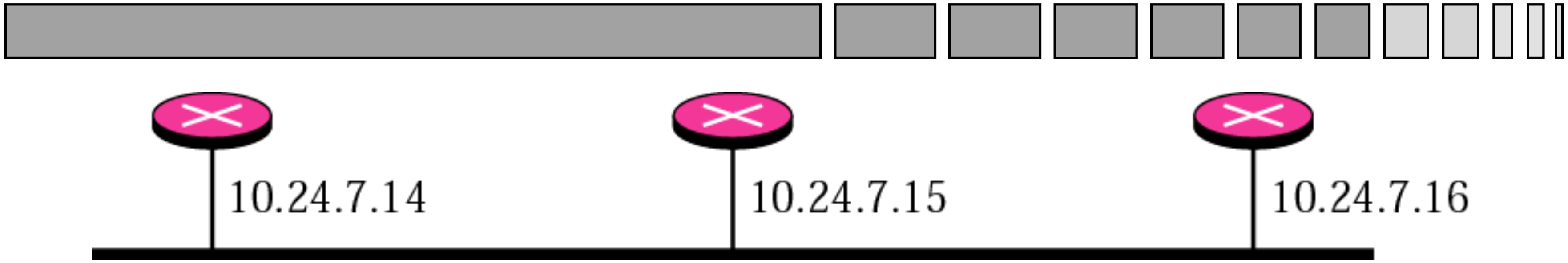


# Network link LSA packet format




**The IP address of the designated router can be found inside the link state header**

# Network LSA - example




The network LSA for the above network is:

| |
|---|
| OSPF Header Type: 4 |
| LSA Header Type: 2 |
| 255.255.255.0 |
| 10.24.7.14 |
| 10.24.7.15 |
| 10.24.7.16 |

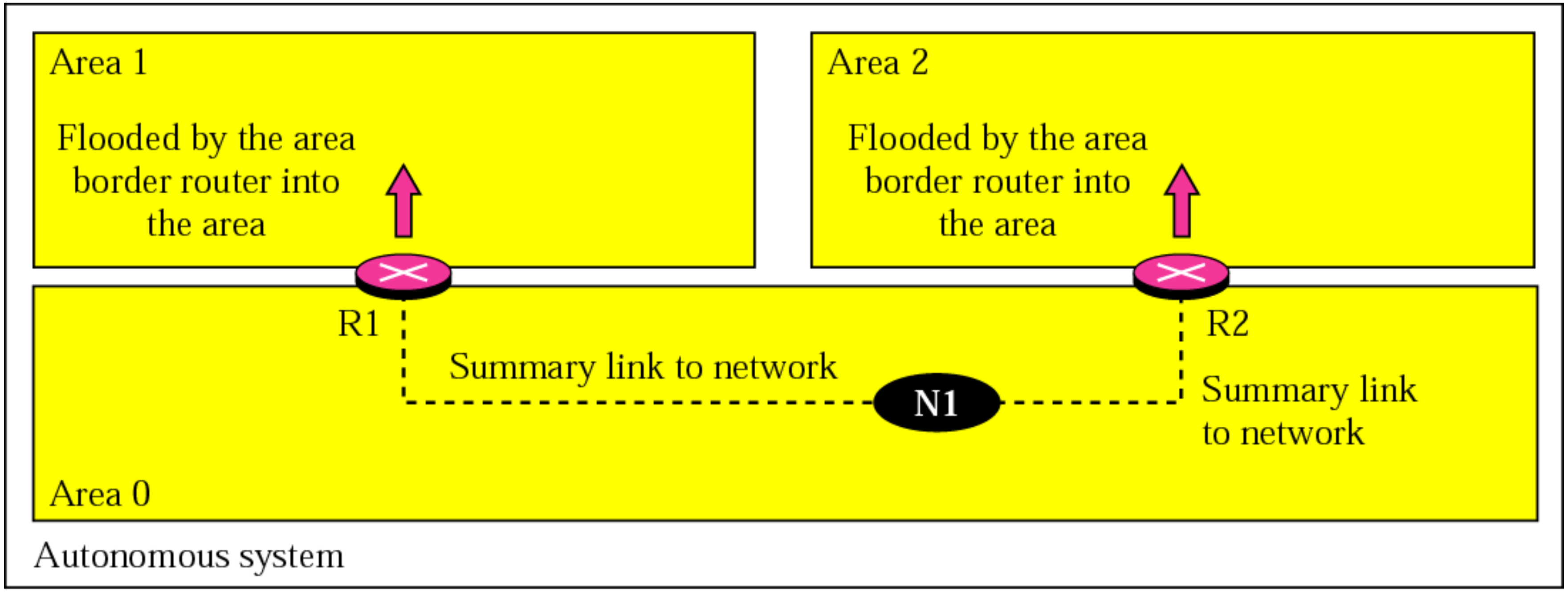
Note: Only one of the routers, the designated router, advertises the network link

# OSPF LSA types

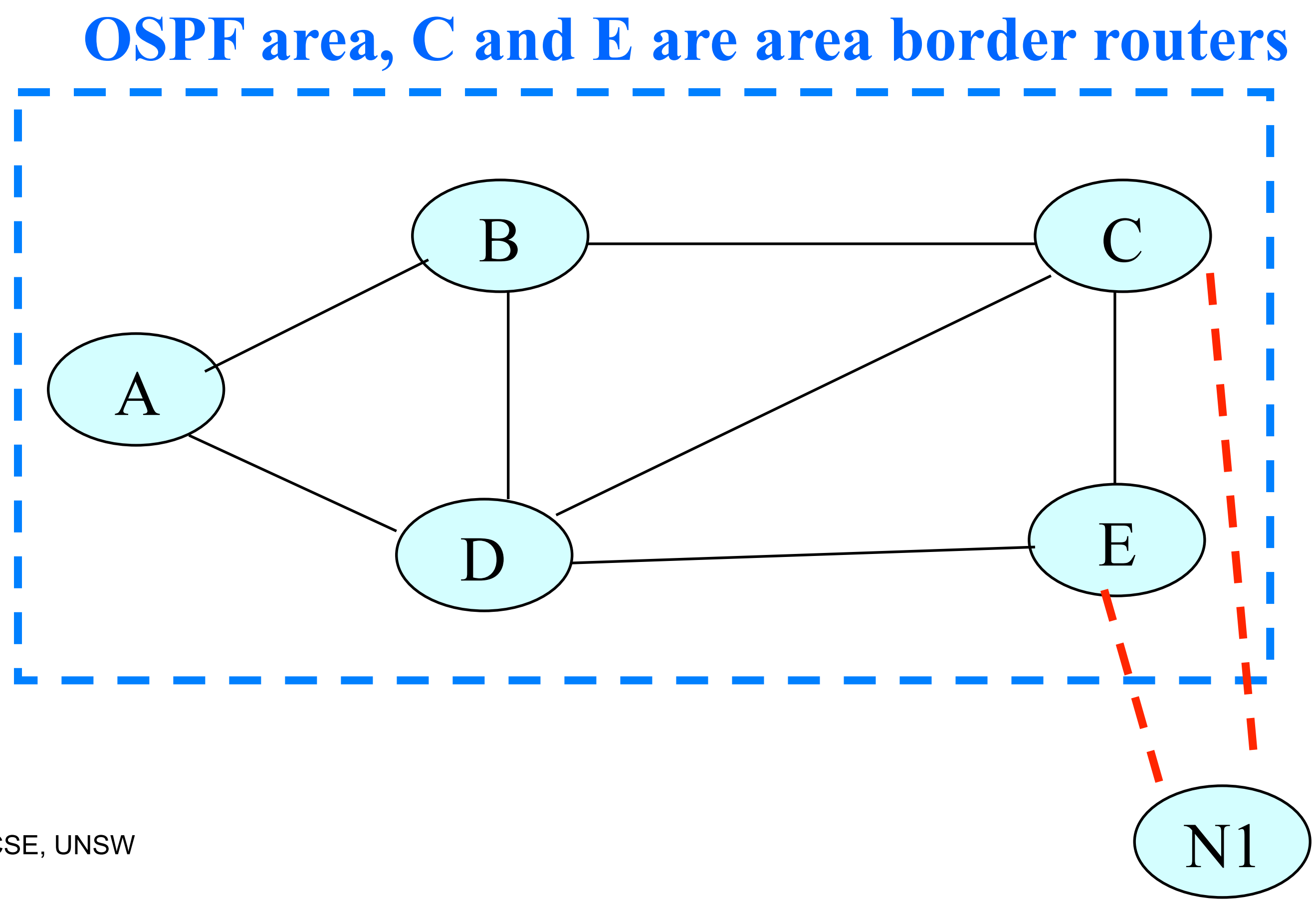
- There are five types of OSPF LSAs
  - Router link
  - Network link
  - Summary link to network
  - Summary link to AS boundary router
  - External link
- From router link and network link LSAs, the routers in an area obtain complete topology in the area
- Routers also need information on other areas

# Summary link to network LSA

- Provides link state for networks in all other OSPF areas within the AS



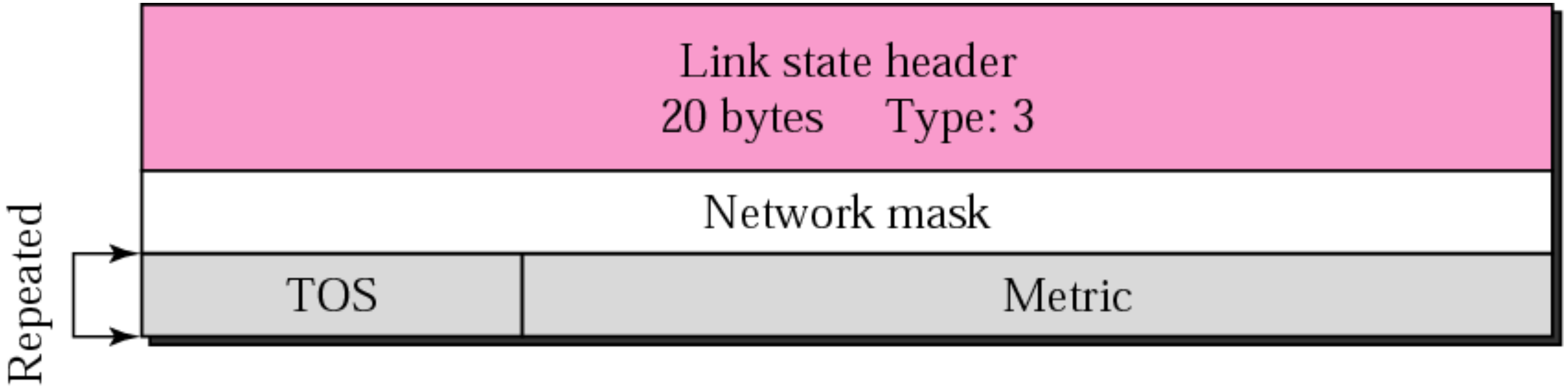
# Summary link to network LSA - view from intra-area routers




**Routers C & E advertises N1**

**To the interior routers, network N1 appears to be attached to routers C & E**

# Summary link to network LSA - packet format

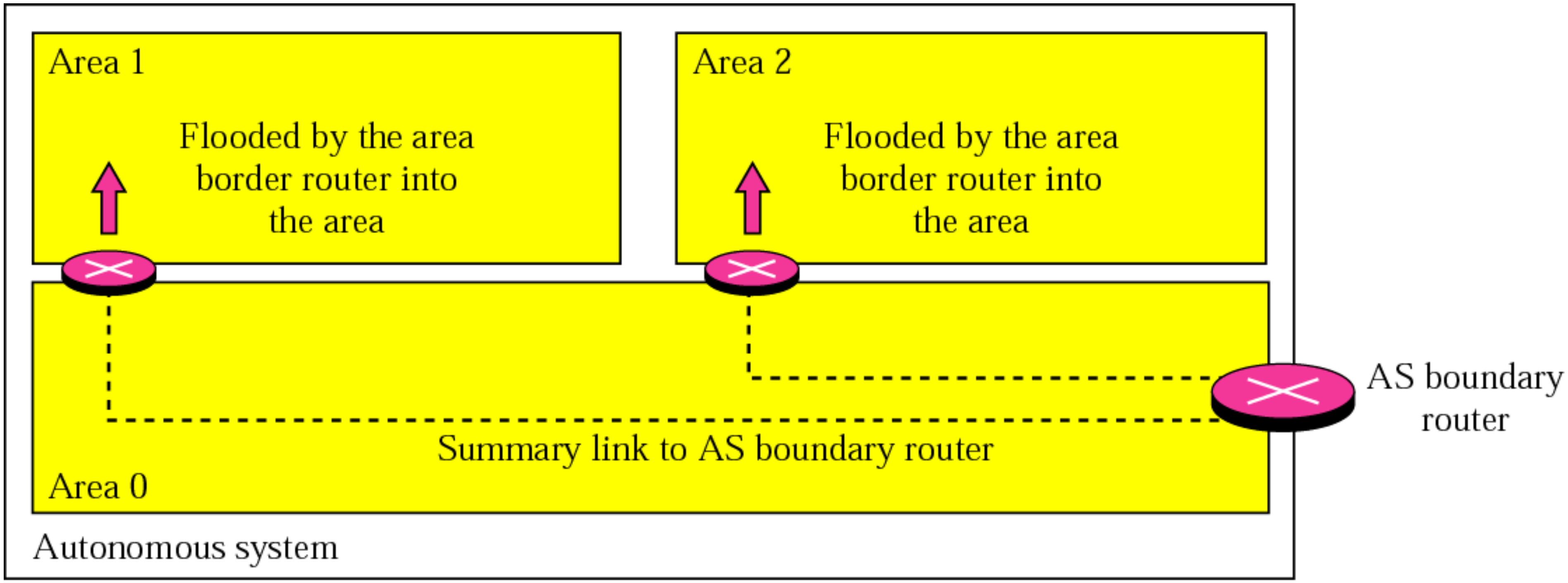



Continuing from the previous slide, interior routers can use the metric supplied by the LSA to decide which area border router it should use to forward packets to N1

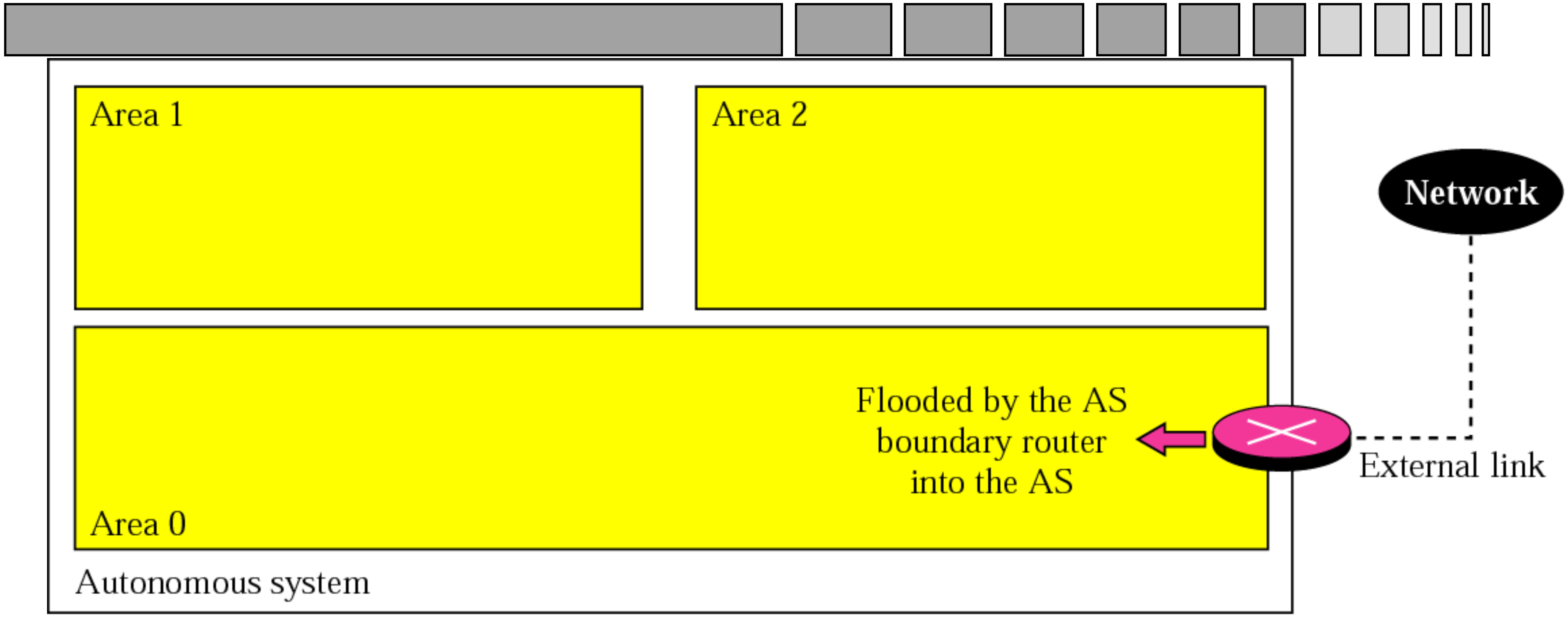
# OSPF LSA types

- {Router link LSA, Network Link LSA} → topology within an OSPF area
- With "summary link to network LSAs", all networks in other OSPF areas within the AS are now reachable
- To reach networks outside the AS, we need
  - Summary link to AS boundary LSA
  - External link LSA

# Summary link to AS boundary router



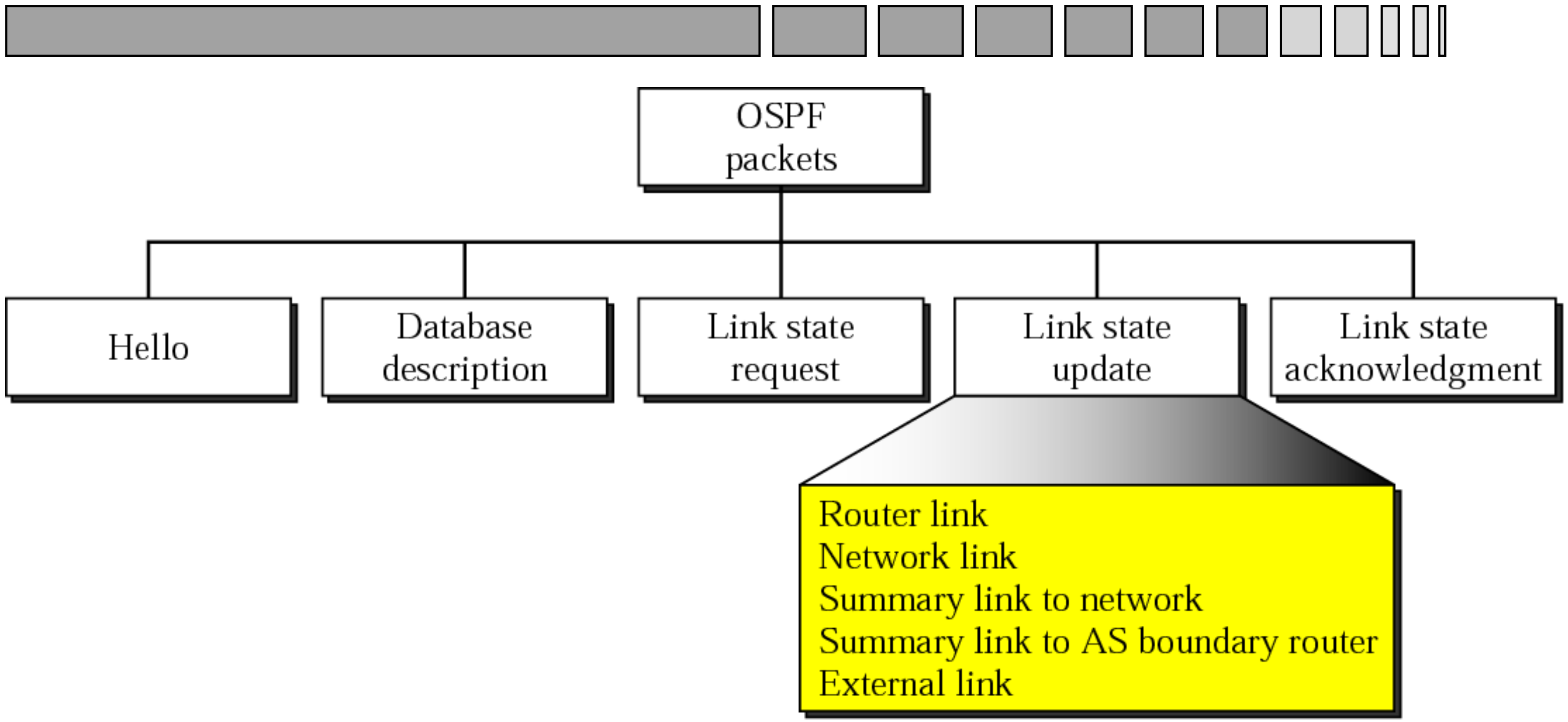
# External link



# Router start up

- A router that has just started needs to bootstrap its link state database by contacting its neighbours
- OSPF defines the following messages
  - Database description message
    - Provides basic information of database contents
  - Link state request packets
  - Link state update packets

# OSPF packet type



# *Issues in routing protocol design*

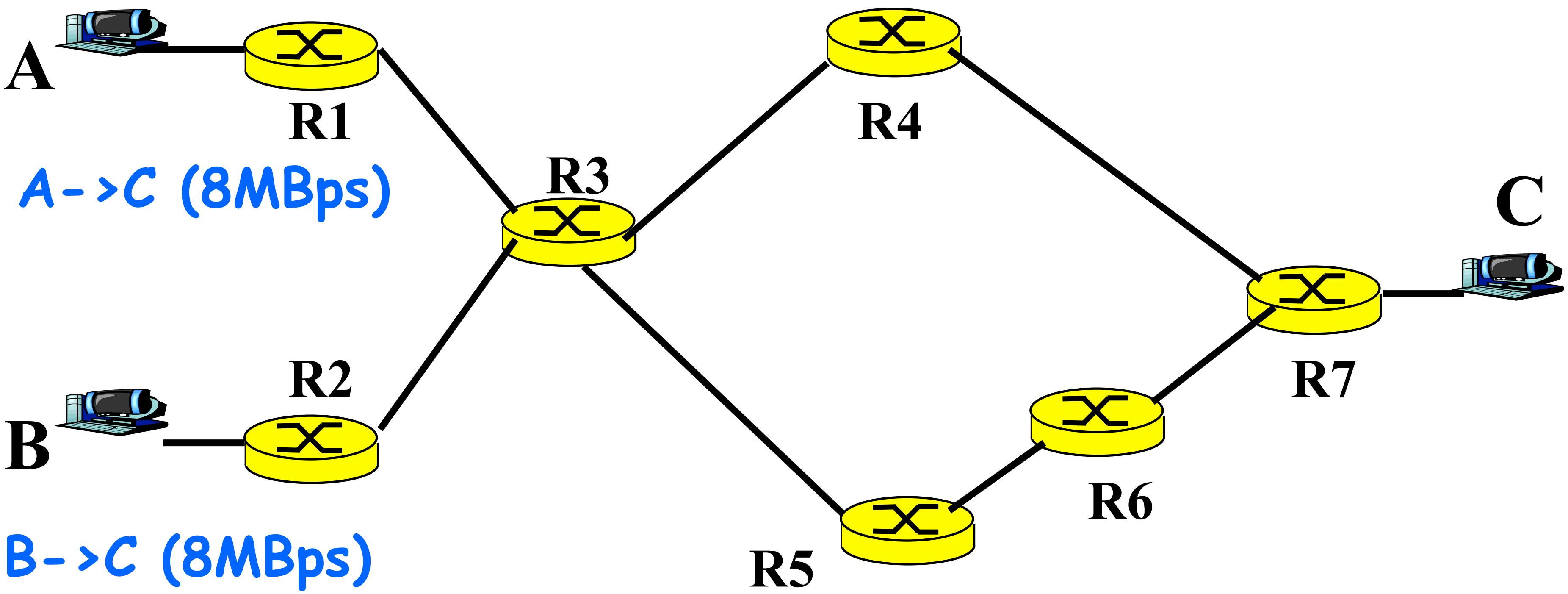
- Issues
  - Communication and processing overheads
  - Optimality
  - Scalability
  - Stability
    - » Convergence time
    - » Loop freedom
  - Security etc.
- Can you identify how OSPF has taken care of some of these issues?

# Intra-domain traffic engineering

- Can we control the paths that the traffic flows take by controlling the OSPF weights?
- Given that
  - If there is only one shortest path to a destination, all traffic to that destination will use that path
  - Multiple equal-cost shortest paths, the flow is split equally on them
    - Why do we need this requirement?

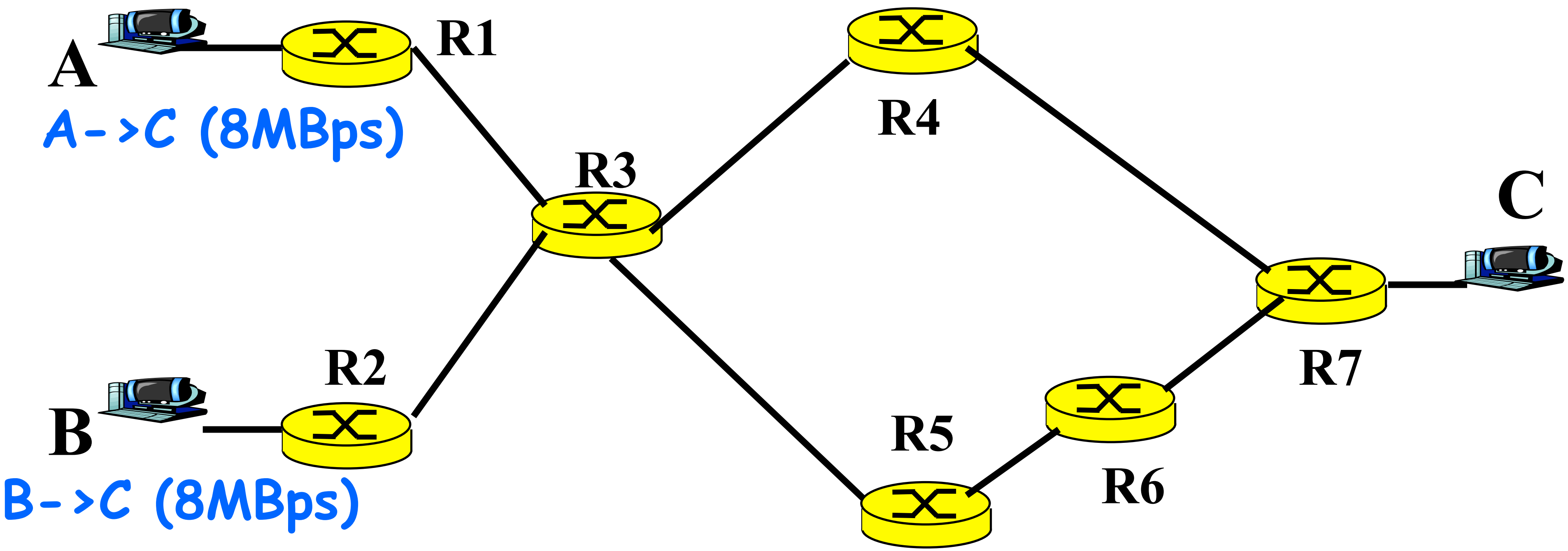
# Intra-domain traffic engineering: Example (1)

- Each point-to-point link has capacity 10Mbps
- Two flows A-C and B-C, 8 Mbps each
- OSPF link weight = 1 for each link
- Question: What paths will the flows take?



# Intra-domain traffic engineering: Example (2)

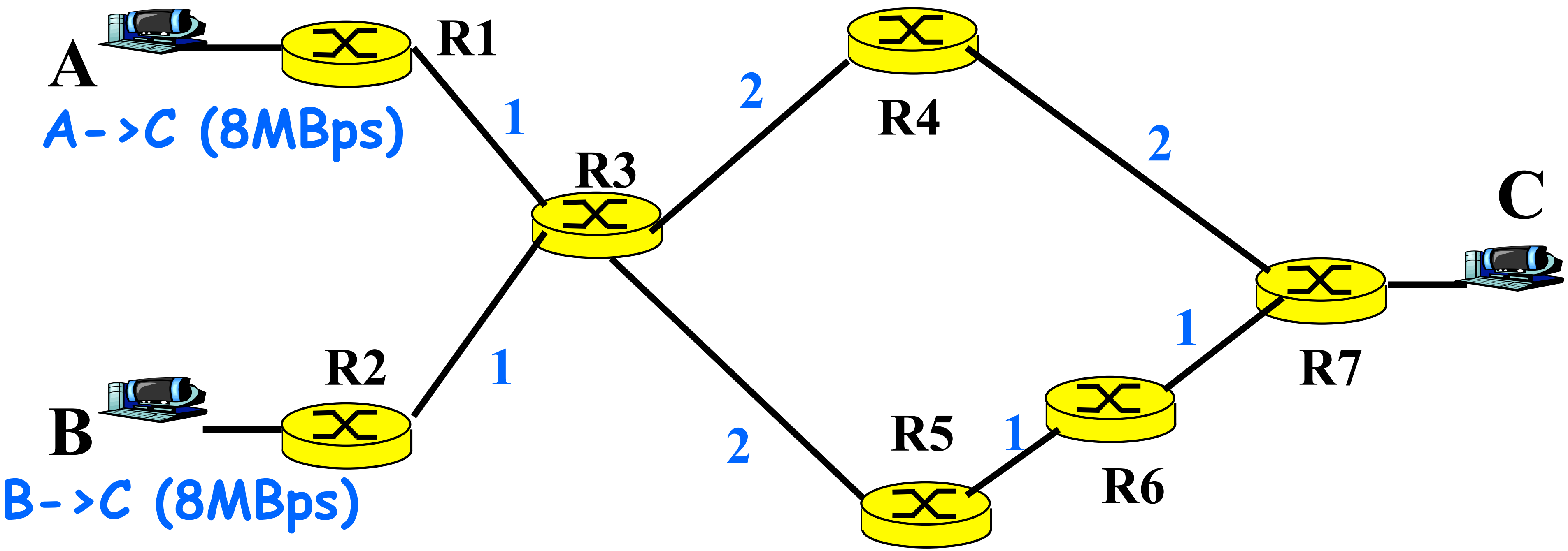
- With OSPF link weight = 1 for each link
  - Both flow uses R1-R3-R4-R7
- What is the problem with this choice of paths?
- Question: Can we change the OSPF link weights so that the flows use different paths
  a) If splitting traffic on equal-cost shortest path is NOT allowed?
  b) If splitting traffic on equal-cost shortest path is allowed?



# Intra-domain traffic engineering: Example (3)

a) If splitting traffic on equal-cost shortest path is NOT allowed?
   - Both flows will always use the same path

b) If splitting traffic on equal-cost shortest path is allowed?
   - For example, using the link weights given below



# How to assign link costs?

- Problem:
  - Given:
    - » Network topology (routers, link, link bandwidth)
    - » Traffic demands: (source, destination, bandwidth)
  - Find a set of OSPF link weights such that congestion is minimised (can use other objectives)
- The problem is NP-hard
- Heuristic solution

# Some intra-domain routing research problems

- Intra-domain traffic engineering
- How to measure source-destination traffic demand?
- How to cope with large fluctuations in traffic demand?

# References

- IBM Redbook - Section 4.6
- Forouzan, 3rd Ed., Chapter 14