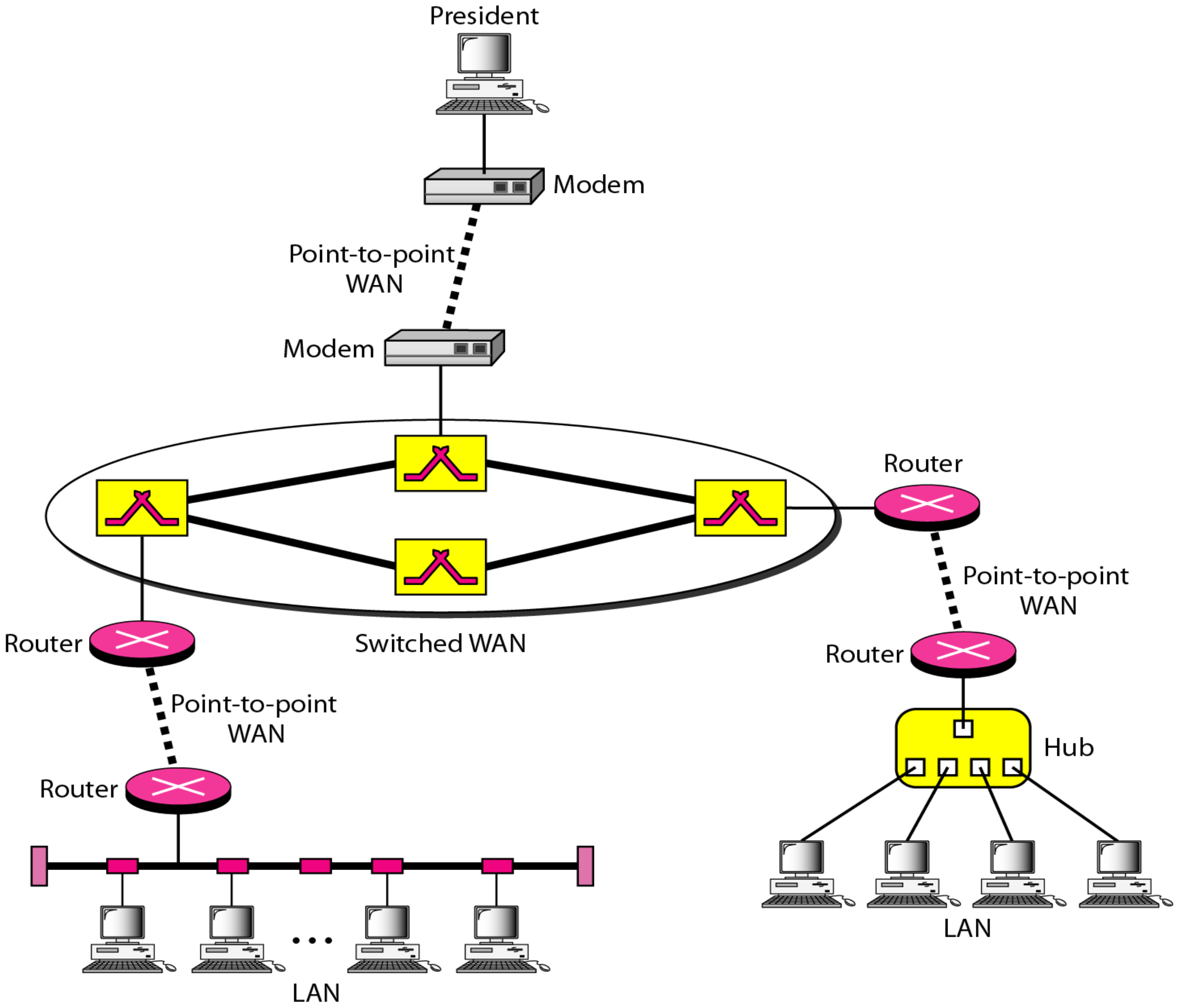
ETSF05:
Models/Paradigmes
ARQ
Routing algorithms

Network engineering

High performance

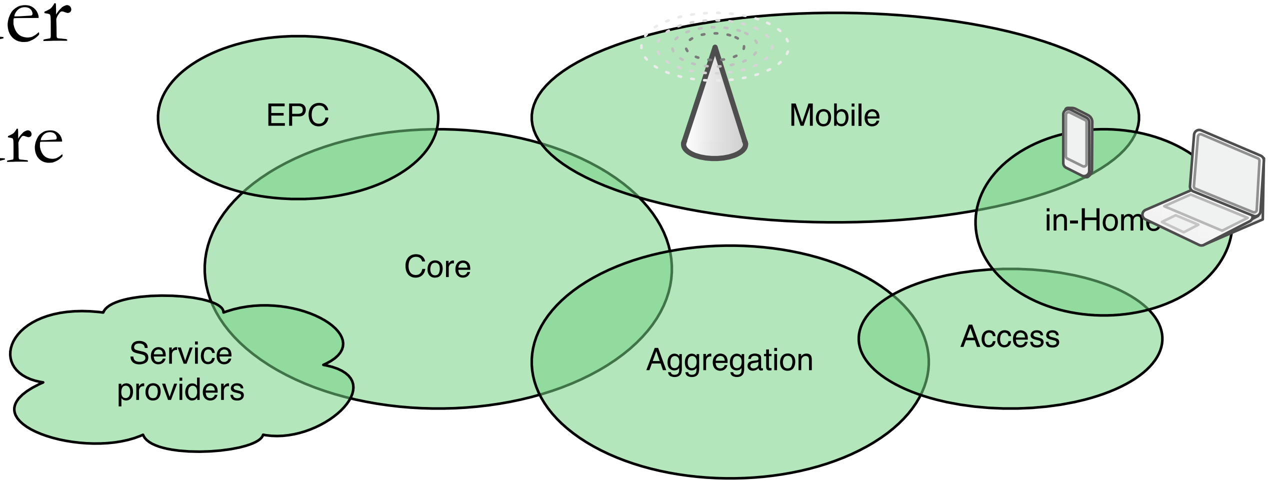
- Reliability
  - Critical infrastructure
- Speed
  - Throughput
  - Latency
- Security
  - Authentication
  - Encryption



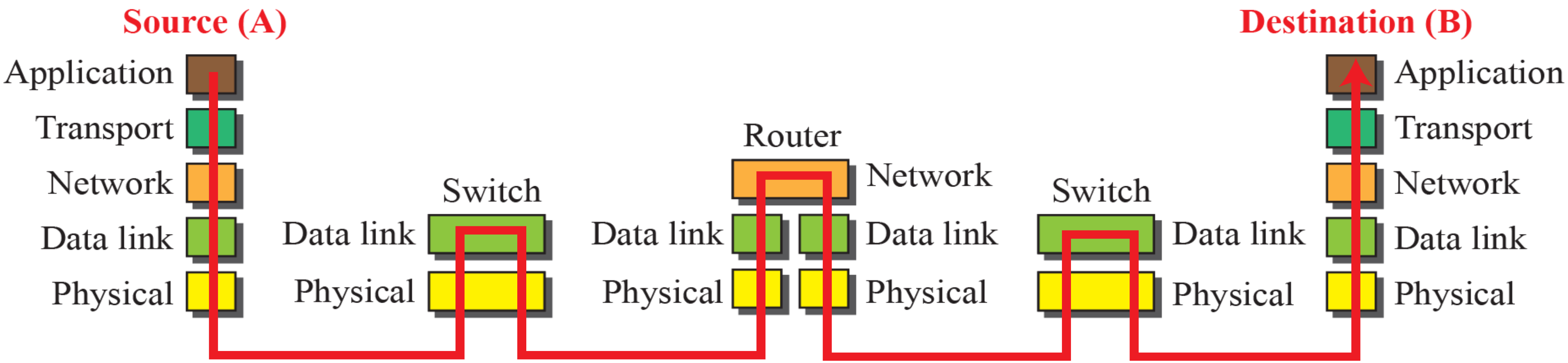
Network models

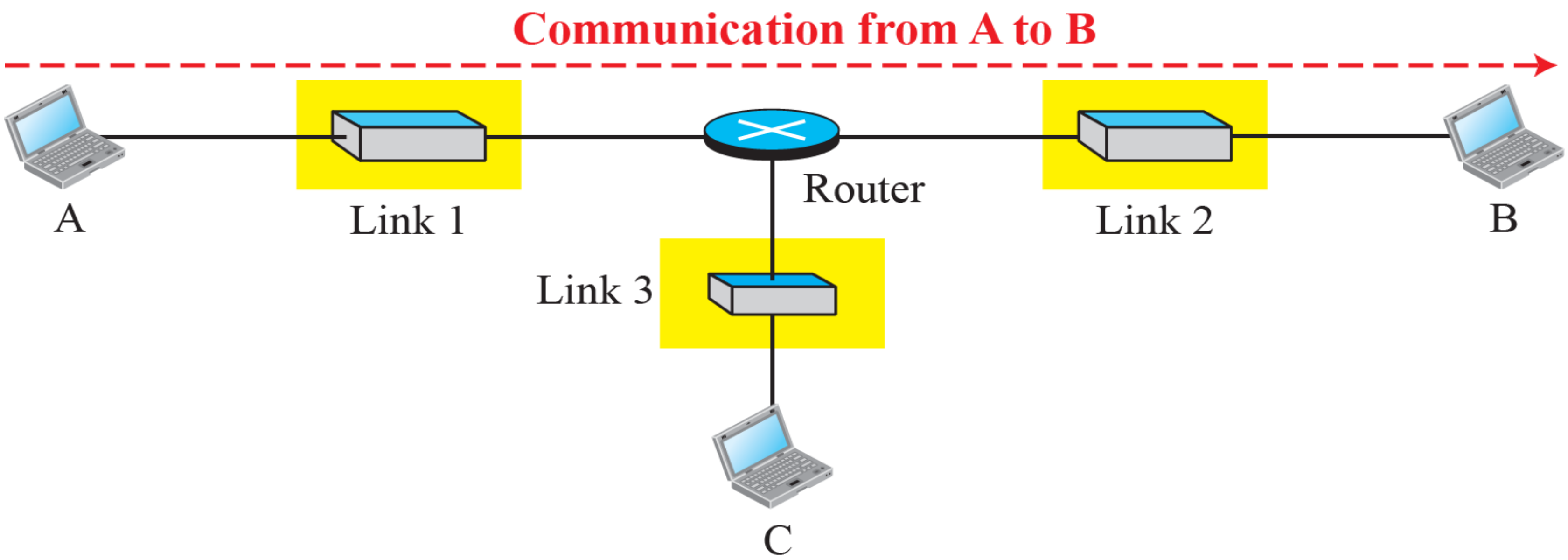
Too complicated

- Divide and conquer
  - ♦ Layered architecture
  - ♦ Hierarchy
  - ♦ Specialisation
  - ♦ Simplification

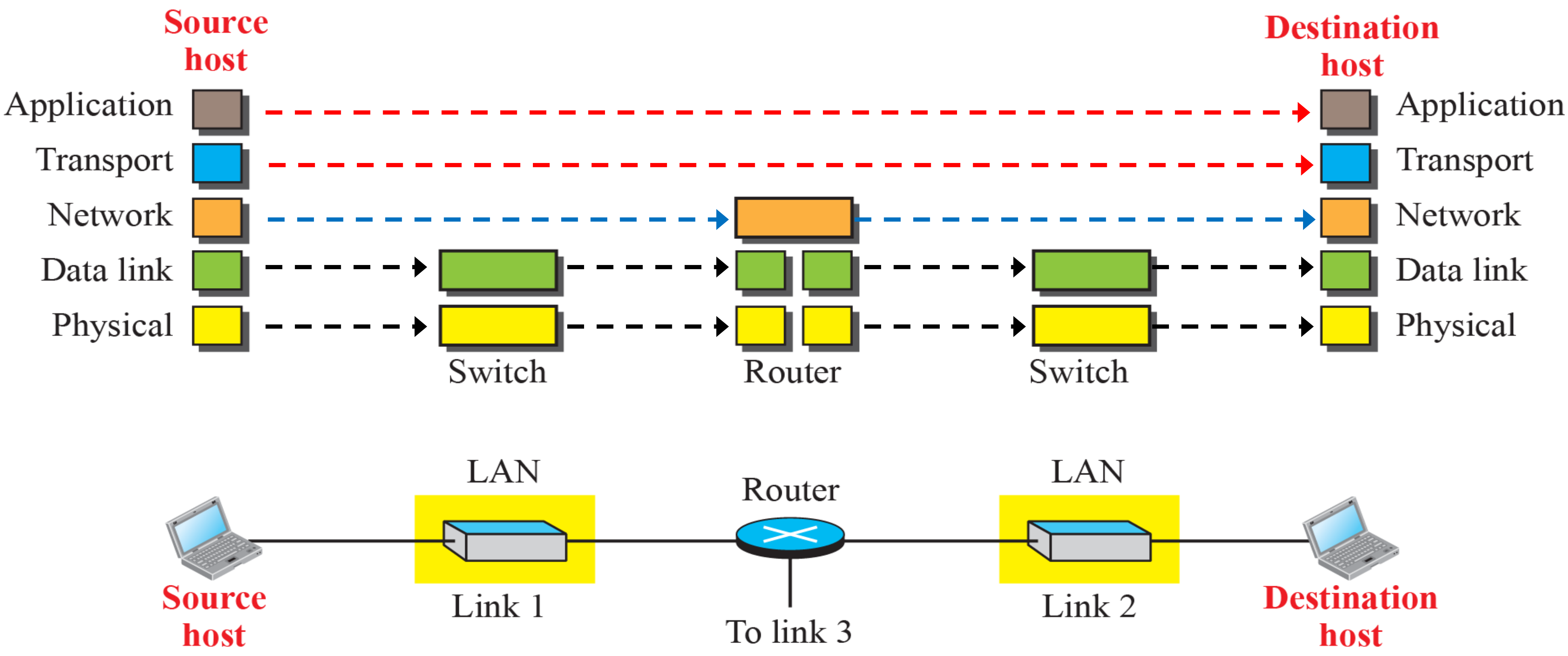


Communication stack





TCP/IP logical connections

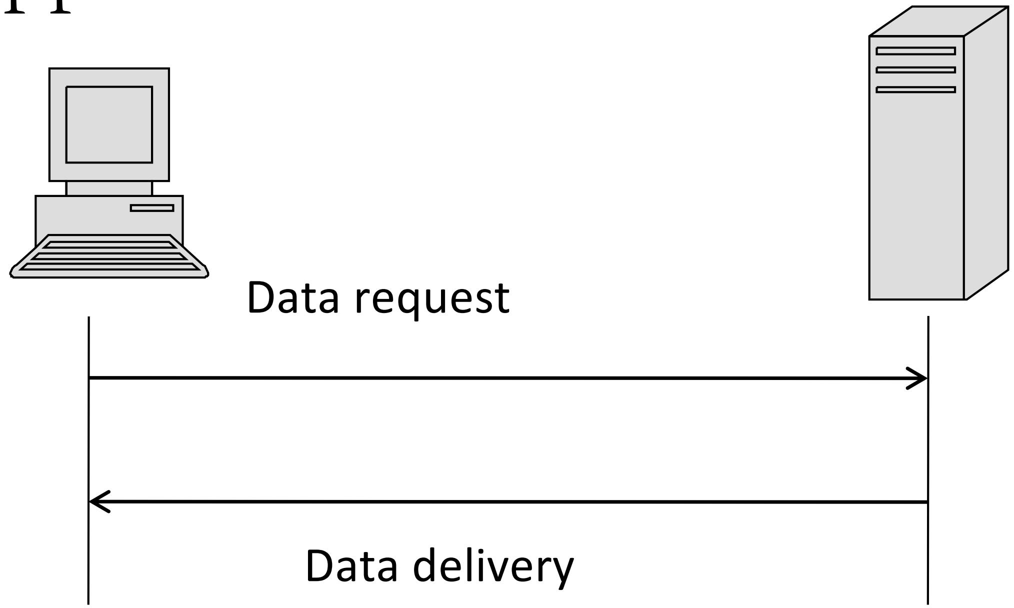


Models

- Micro computers (master – slave)
- Client-server
    - *Client-Server Paradigm*
- Peer–to-peer
    - *Peer-to-peer (P2P) Paradigm*

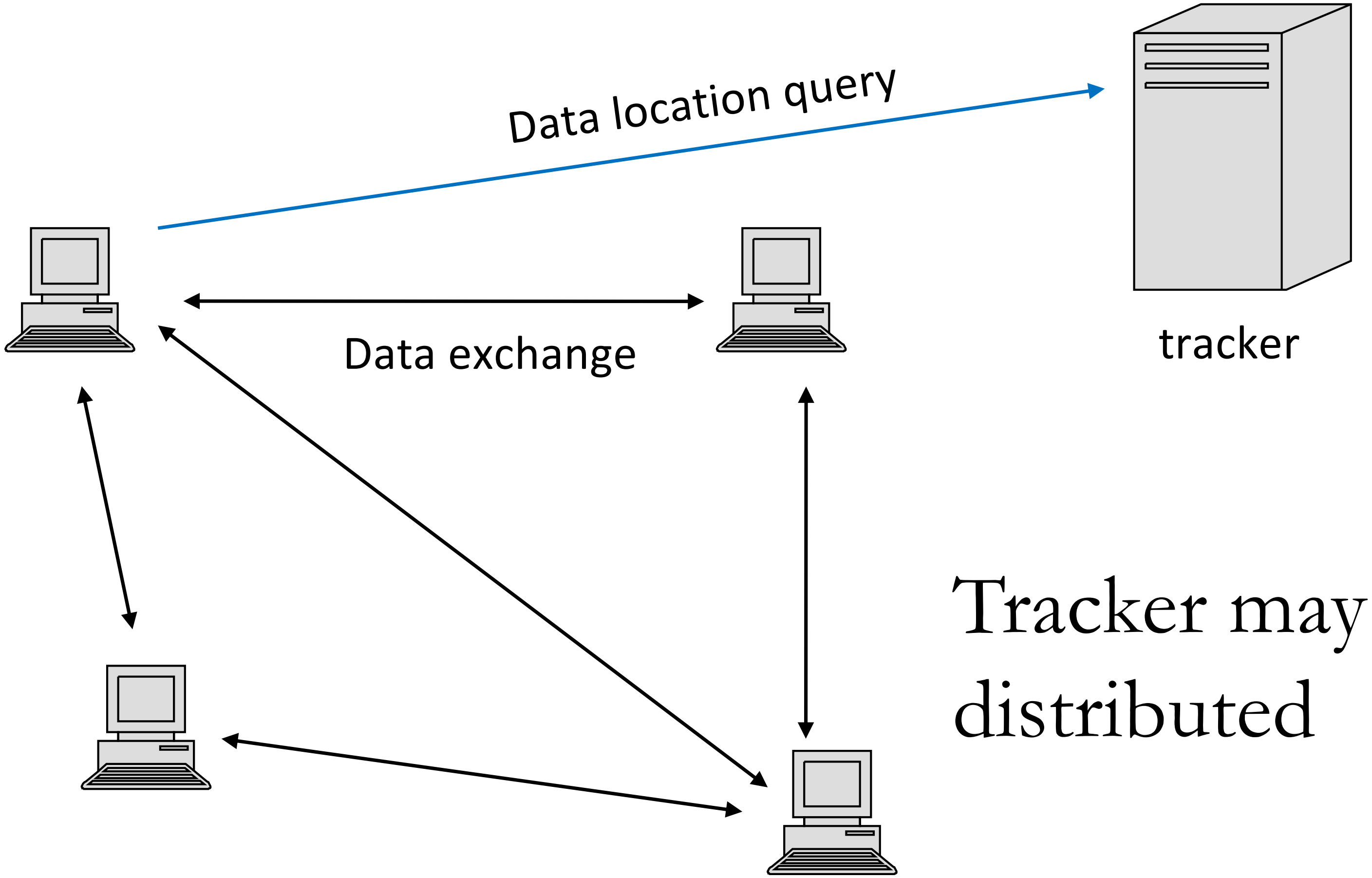
Client-Server paradigm

- Application



- Module supporting
- application

Peer-to-peer



Tracker may be distributed

Control protocols (link level)

Error control

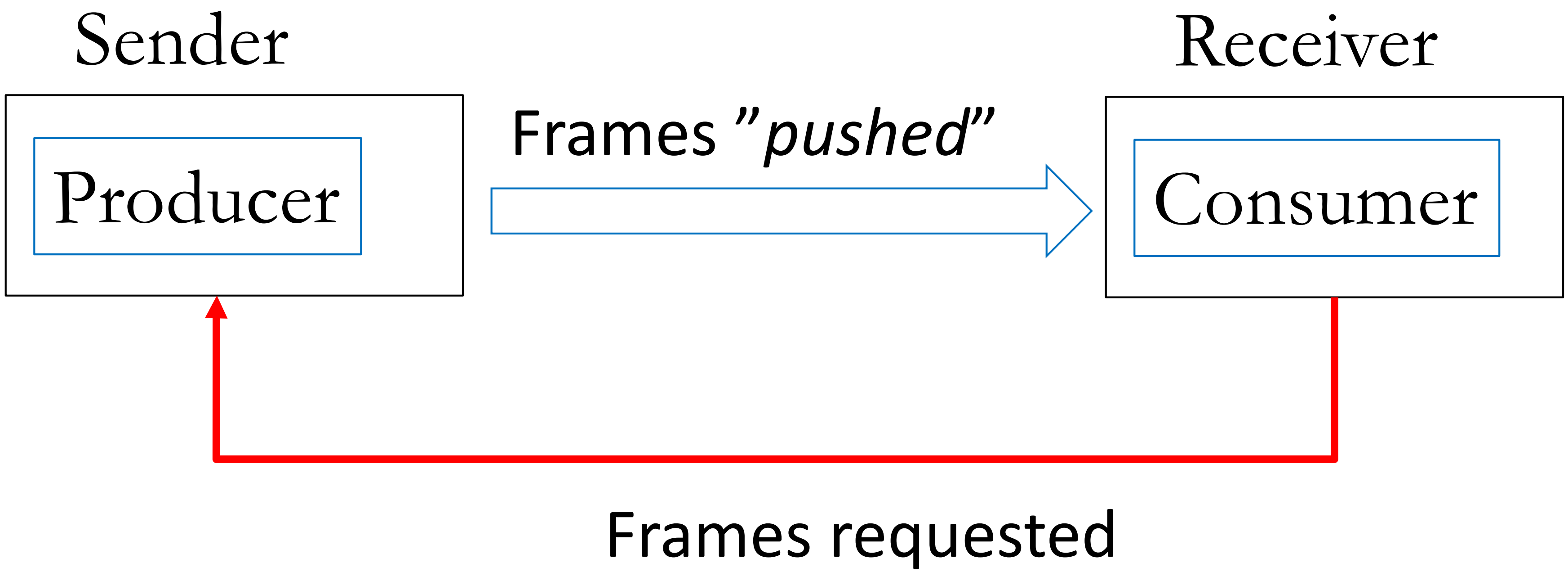
- Discovery
- Retransmission (ARQ)
- Correction

Flow control (ARQ)

- Stop-and-wait
- Go-back-N
- Selective Repeat

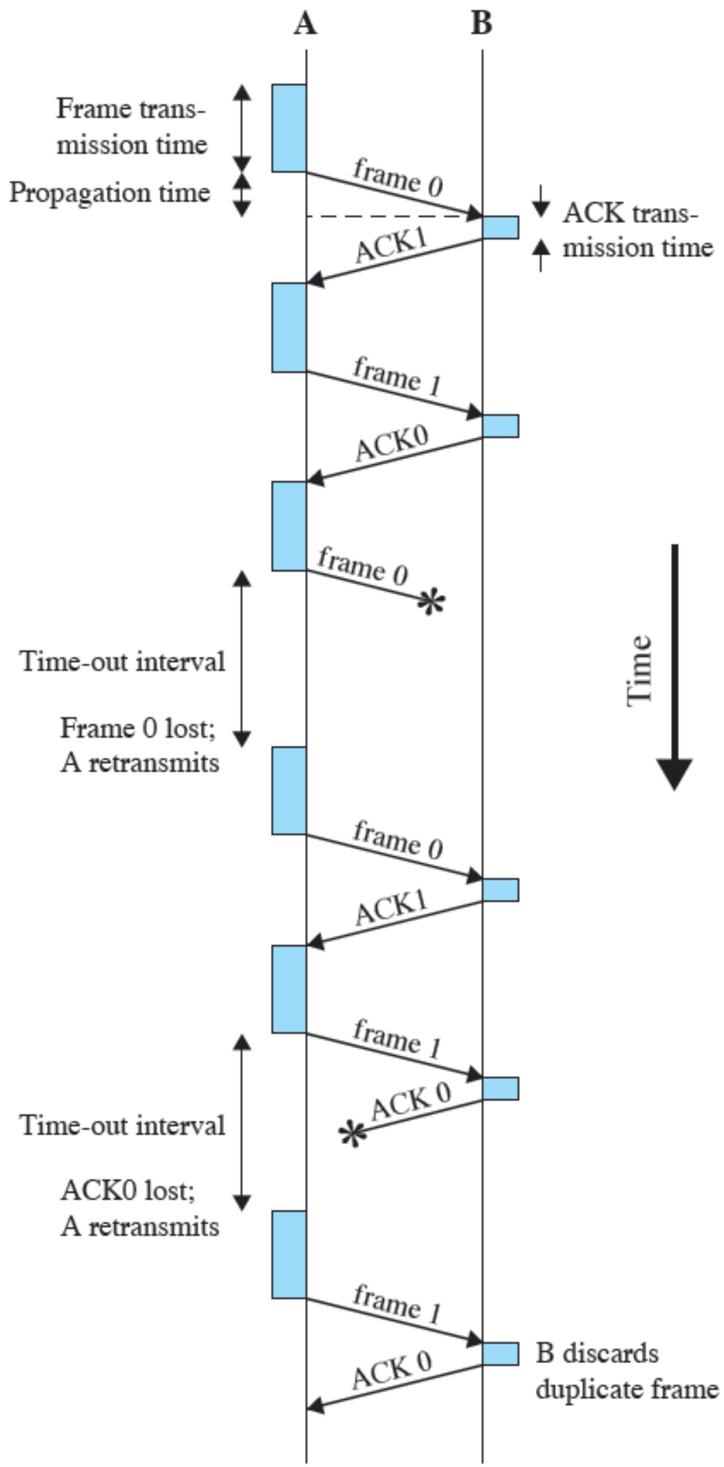
May occur at higher layers too (cf. TCP)

Flow Control

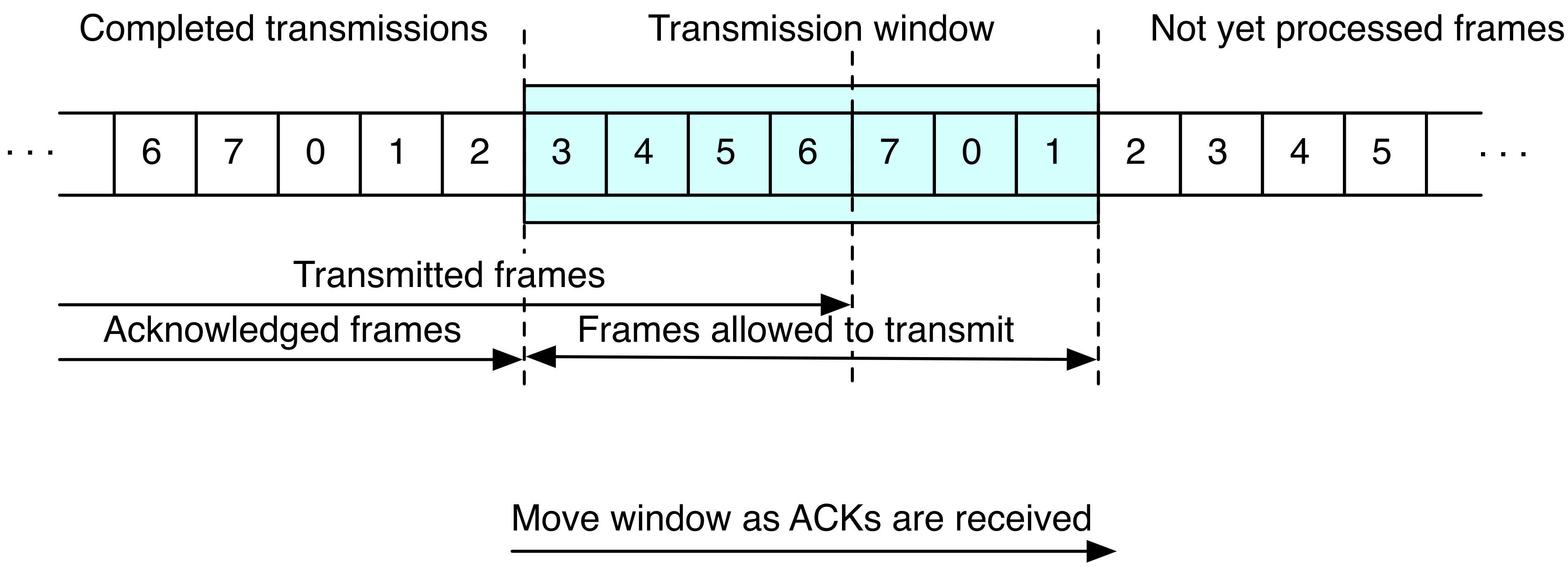


Stop-and-wait ARQ flow diagram

- Ineffective
- Long wait states, especially over long links



Sliding window (sender)



Go-Back-N ARQ

Most commonly used error control

ACK (this or next)

Based on sliding-window

Use window size to control number of outstanding frames

While no errors occur, the destination will acknowledge incoming frames as usual

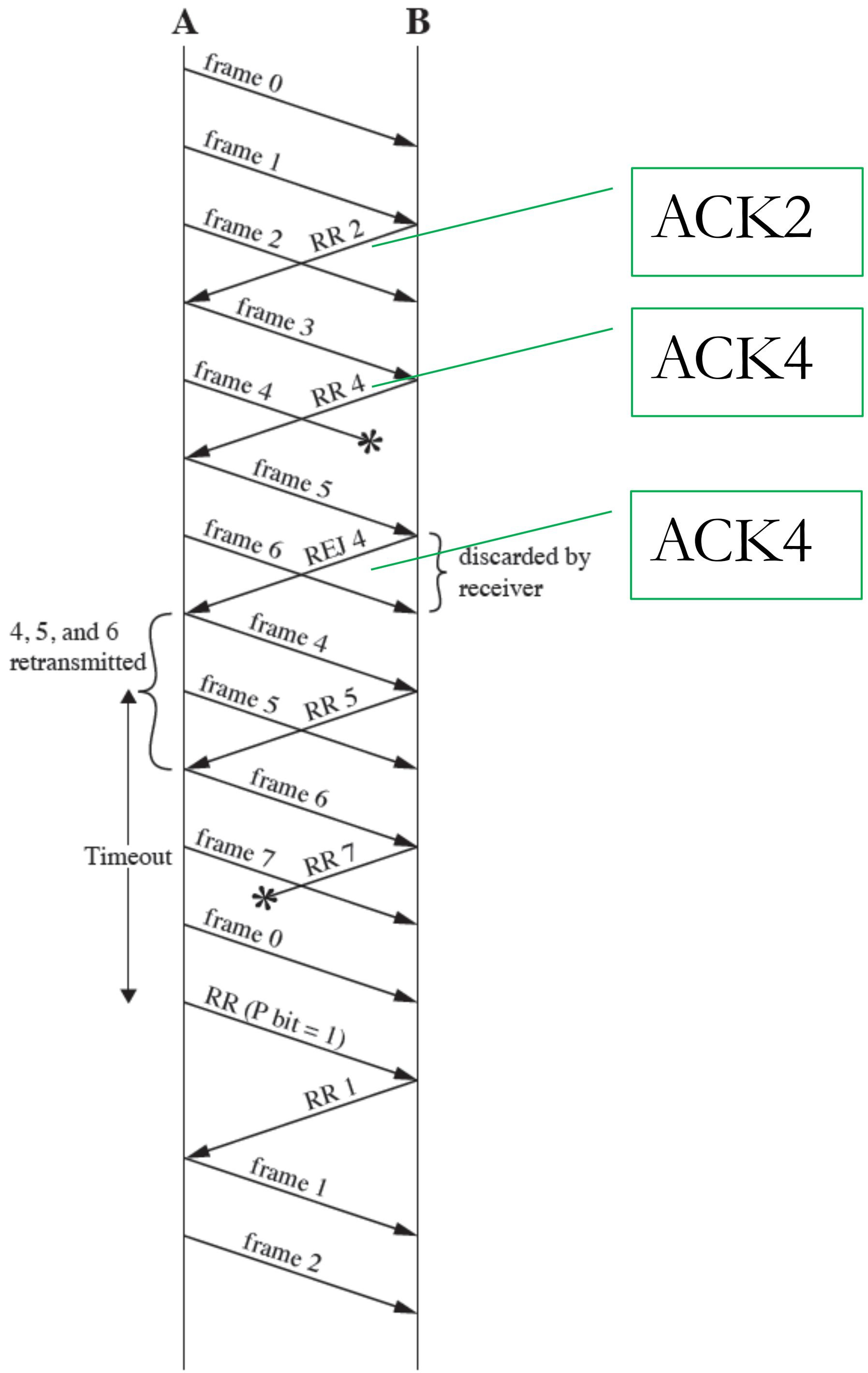
- **RR=receive ready, or piggybacked acknowledgment**

If the destination station detects an error in a frame, it may send a negative acknowledgment

ACK (previous or this)

- **REJ=reject**
- Destination will discard that frame and all future frames until the frame in error is received correctly
- Transmitter must go back and retransmit that frame and all subsequent frames

Go-Back-N



(a) Go-back-N ARQ

Selective-Reject (ARQ)

Also called selective retransmission or selective repeat

Only rejected/missing frames are retransmitted

Subsequent frames are accepted by the receiver and buffered

Minimizes retransmission

Receiver must maintain large enough buffer

More complex logic in transmitter

- Less widely used

Useful for links with long propagation delay, e.g satellite, also at transport layer (TCP)

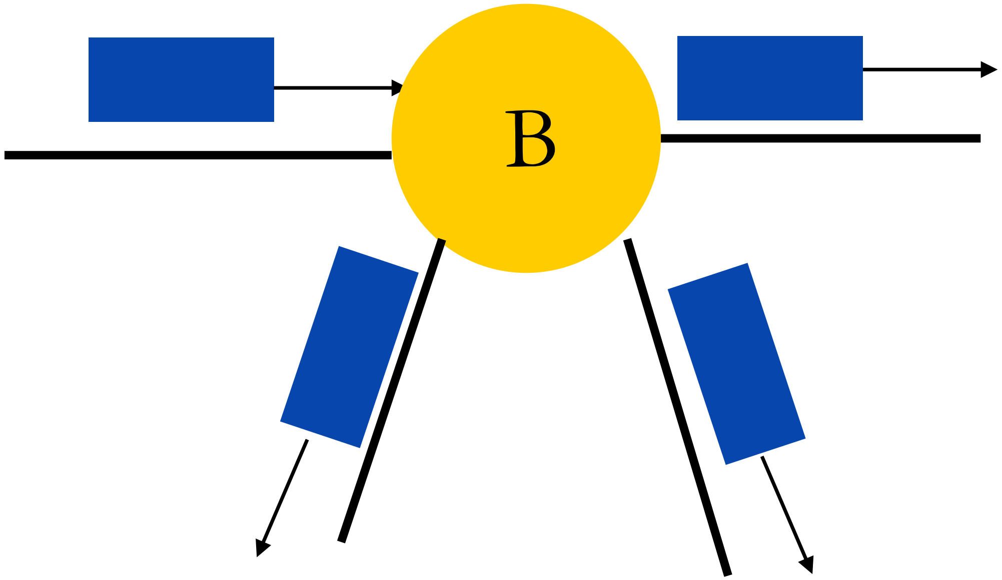
Congestion control

Higher layers

- Control injection of traffic to follow free capacity over several links
- Works together with ARQ mechanisms
- Later in TCP

Routing – How to find the best way?

Flooding means that an incoming frame is echoed on all outgoing ports. Hop count is used to avoid loops.



Instead, how to find the best path?

Routing algorithms (no mobility)

- The art of finding a least cost tree of a graph
    - From sender to receiver
    - From every node to all other nodes
- Three variants
    - Distance Vector
    - Link State
    - Path Vector
        - Policy-based routing

Routing algorithms (mobility)

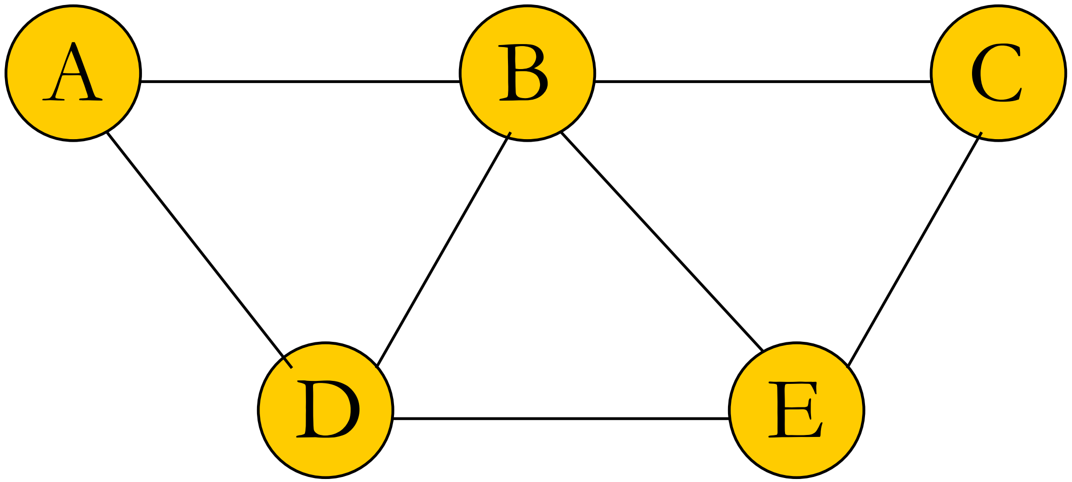
Key consideration, rate of change

- Above approaches may not converge
- Flooding
- Record route
- Source routing
- Geographic routing etc.

Covered in ETSN 10, Network Architecture and Performance

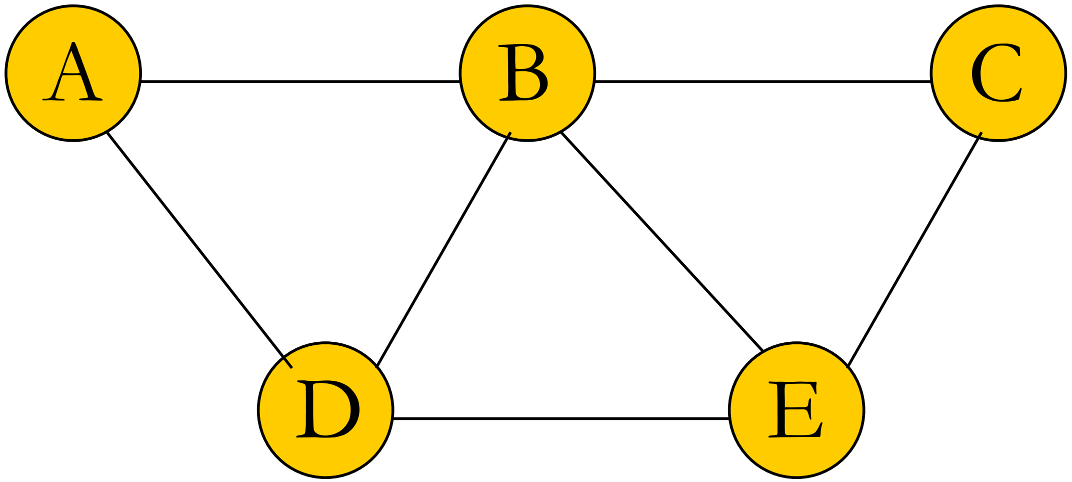
Network graph

A network can be represented as a graph of nodes and links



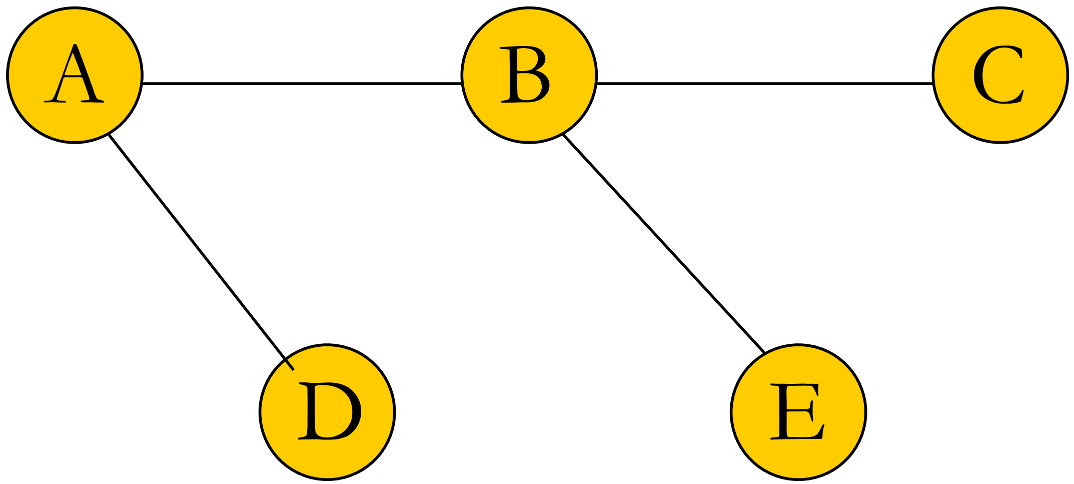
Least-hop path

Find the shortest path between all nodes



Least-hop path

E.g. From A to all other nodes



Link cost

Every link in the graph has an associated cost

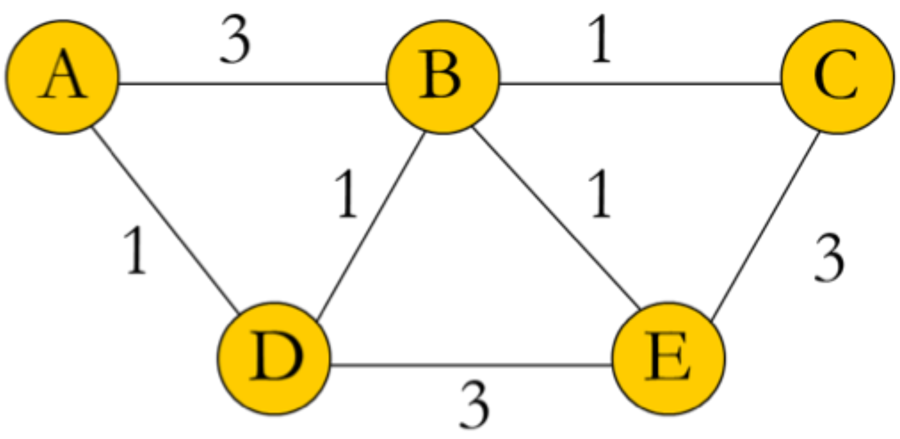
The cost can have many dimentions, e.g.:

- Raw link speed
- Latency
- Link load
- Distance
- Medium
- Etc.

Graph

A graph consists of nodes(N) and edges (E) with weights $w(e)$.

Example (undirected graph)

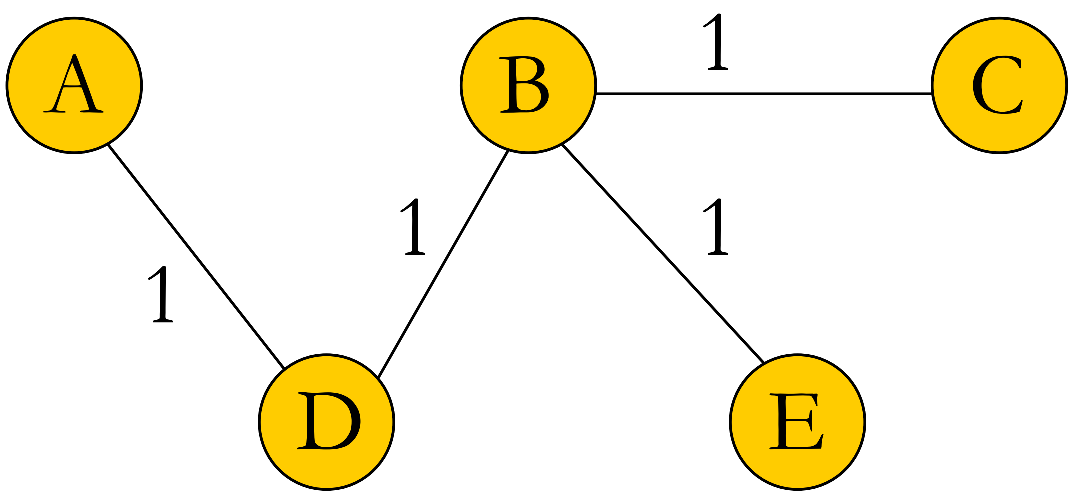


$N = \{A,B,C,D,E\}$

E	*w(e)*
AB	3
AD	1
BC	1
BD	1
BE	1
CE	3
DE	3

Least-cost path

- In network graphs, weights are positive and additative.
- A Least-cost path is a path with minimum total cost



Routing table

- Routing core idea: each router makes independent decision of where to forward a packet
- Keeps a table of rules for next hop
- The problem is:
  - How to fill in the table?
  - Updating table as something changes?

Distance vector routing

All known shortest paths are shared with neighbours

- Periodically
- When changes occur

Routing tables are updated with

- New entries
- Changed costs

”Global knowledge spread locally”

Least cost alg 1
Bellman-Ford

Find shortest path from source node s to the rest.

Let $d(n)$ be cost from s to n

Init:

$d(s) = 0$

$d(n) = \infty,\ n \neq s$

for i = 1 $to|N| - 1$

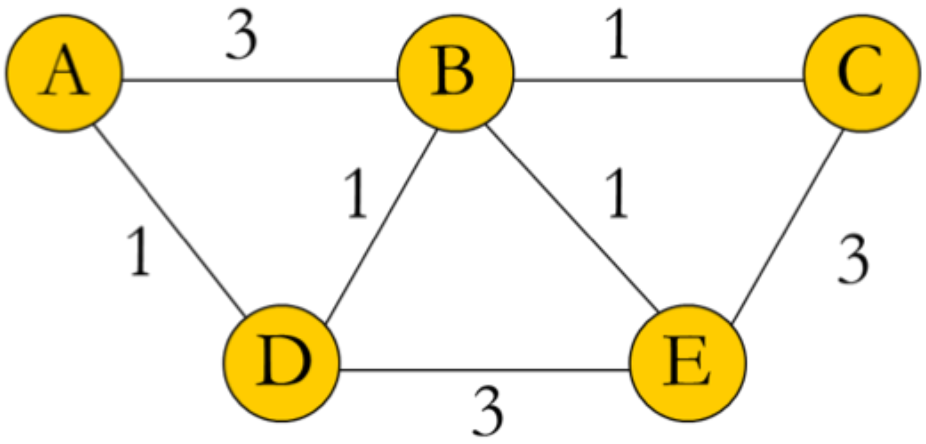
for each $n \in N$

$d(n) = \min_{u \in N}\{d(u) + w(u, n)\}$ // Find shortest path from node

// u to node n in a single step

Keep track of the shortest path

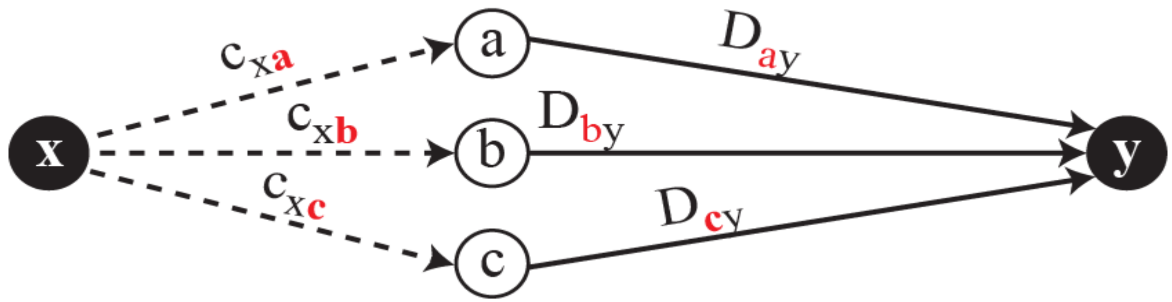
Example: Bellman-Ford



Nod	A	B	C	D	E
init	0	∞	∞	∞	∞
i=1	0	3	∞	1	∞
i=2	0	2	4	1	4
i=3	0	2	3	1	3
i=4	0	2	3	1	3

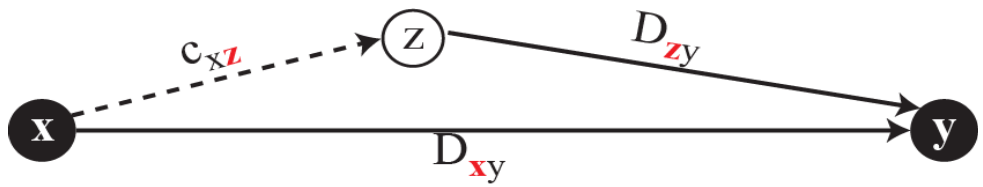
- i gives all possible nodes i hops away

Bellman-Ford's algorithm graph view



a. General case with three intermediate nodes

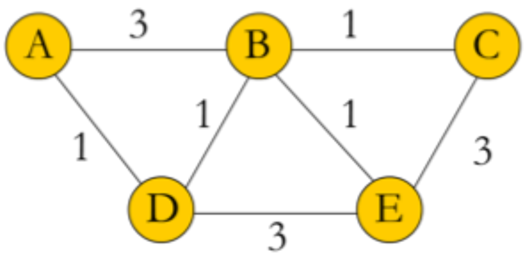
$$D_{xy} = \min\{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}) \ldots\}$$

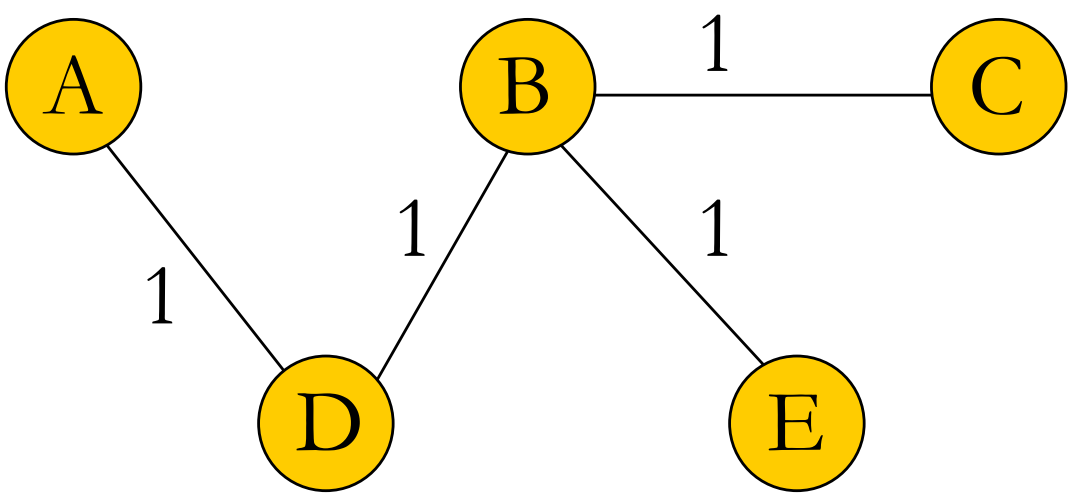


b. Updating a path with a new route

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

Network graph as a tree

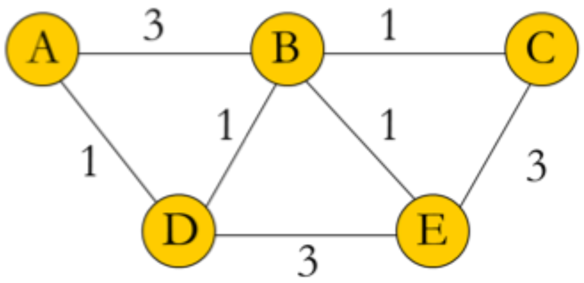




Distance vector for A after network converged

Node	Dist
A	0
B	2
C	3
D	1
E	3

Updates



$$A[\,] = \min(A[\,], cost(A\text{-}B) + B[\,])$$

A, original

Nod	Dist
A	0
B	3
D	1

Update from B

Nod	Dist
A	3
B	0
C	1
D	1
E	1

A, updated

Nod	Dist
A	0
B	2
C	4
D	1
E	4

Distance Vector, things to ponder

- Periodic updates!?
- Problem when edges and nodes disappear.
- How discovering neighbours?
- How discover a neighbour disappearing?

More on distance vector

Later:

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

Link state routing, principle

Local topology flooded to entire network

- At every change
- Periodically (in reality seldom)

Every node constructs own database

Routing table updated with

- New entries
- Changes in cost

”Local knowledge is spread globally”

Least cost alg 2
Dijkstra

Find shortest path from source node s to the rest.

Let $d(n)$ be cost from s to $n, E = set\ of\ nodes$

Init:

$d(s) = 0$

$d(n) = \infty,\ n \neq s$

$\mathrm{P} = s$ // Permanent

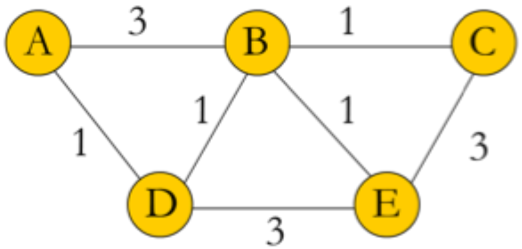
while $\mathrm{P} \subset E$

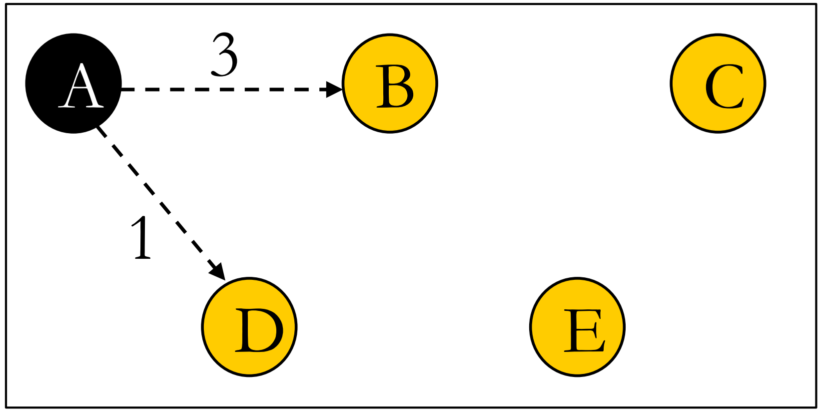
d(u) = neigbours (last in P)

$u = arg \min_{u \notin P} d(u)$ // Find lowest weight to last permanent's neighbors, add d(u) to tentative

$\mathrm{P} = P \cup u$ // Add u to permanent

SPF example with graph





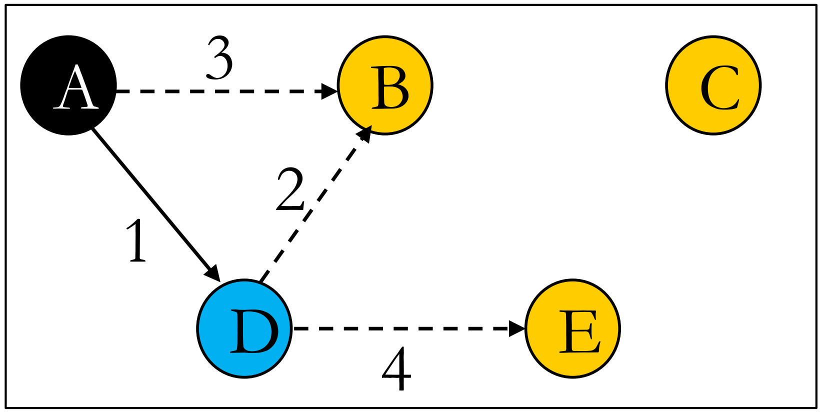
Rotnod

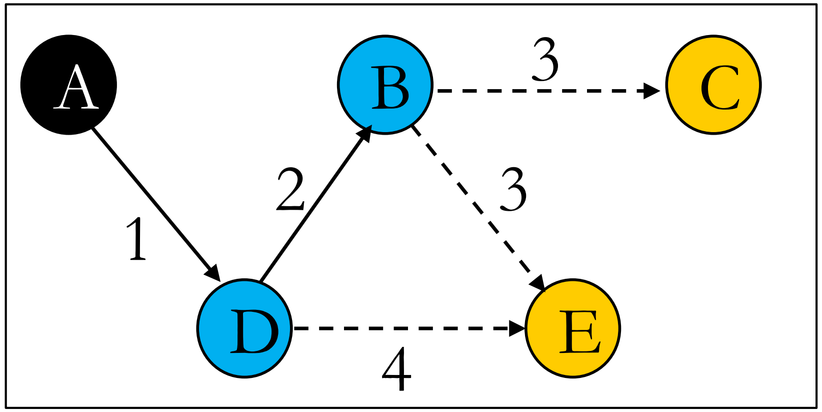
Permanent nod

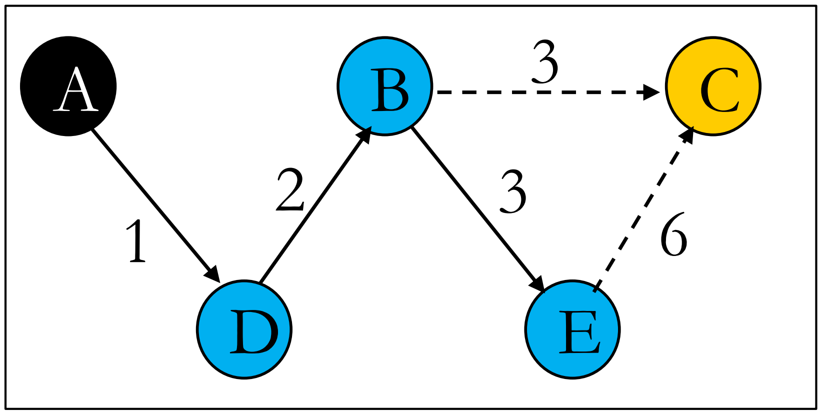
Preliminär nod

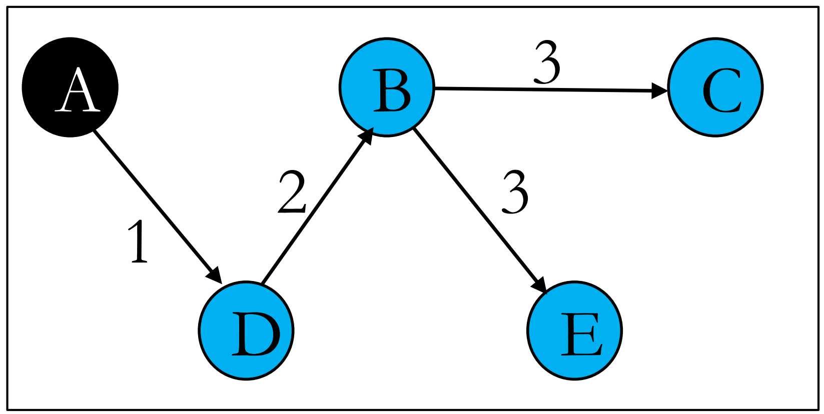
Potentiell väg

Permanent väg





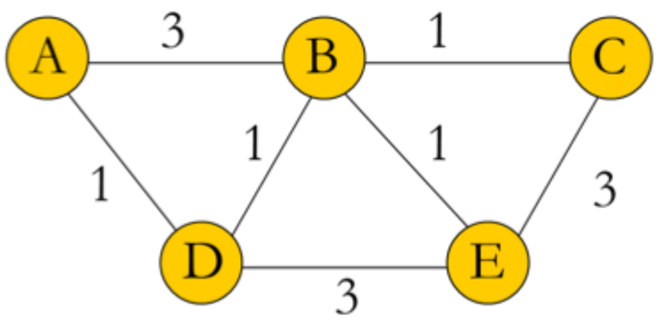




Dijkstra table

Visited	L(A)	L(B)	L(C)	L(D)	L(E)
ϕ	**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}					

Exemple *link state* database



	A	B	C	D	E
A	0	3	∞	1	∞
B	3	0	1	1	1
C	∞	1	0	∞	3
D	1	1	∞	0	3
E	∞	1	3	3	0

∞ means node unknown

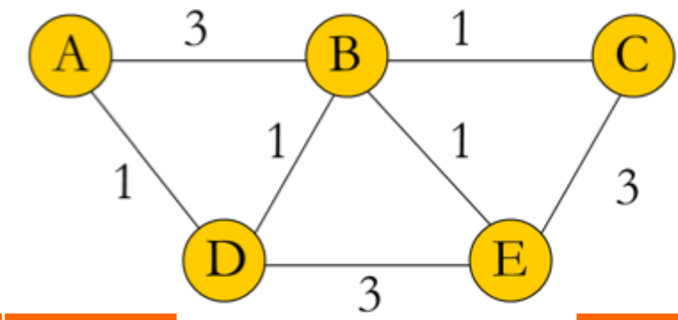
0 distance to self

Link State Advertisements

Nod	Kostn
A	3
C	1
D	1
E	1

Nod	Kostn
B	3
D	1

Nod	Kostn
B	1
E	3



Nod	Kostn
A	1
B	1
E	3

Nod	Kostn
B	1
C	3
D	3

Updated at change!

Link State, things to ponder

- Periodic updates!?
- Problem with nodes and edges that disappear.
- Discover that neighbour disappeared?
- Discover new neighbour?

More on link state

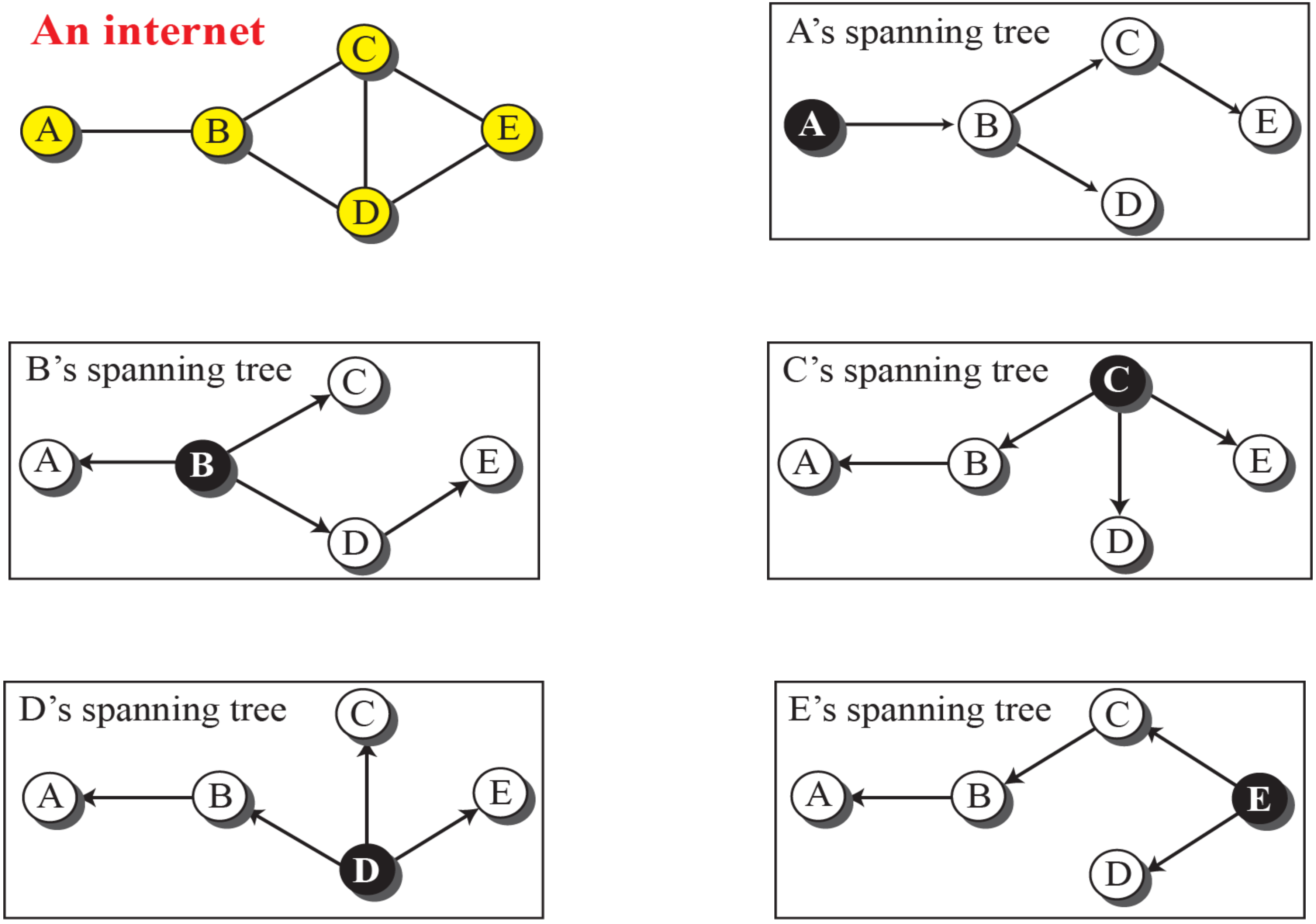
Later:

- Routing protocol OSPF
- The Area concept, how to limit flooding

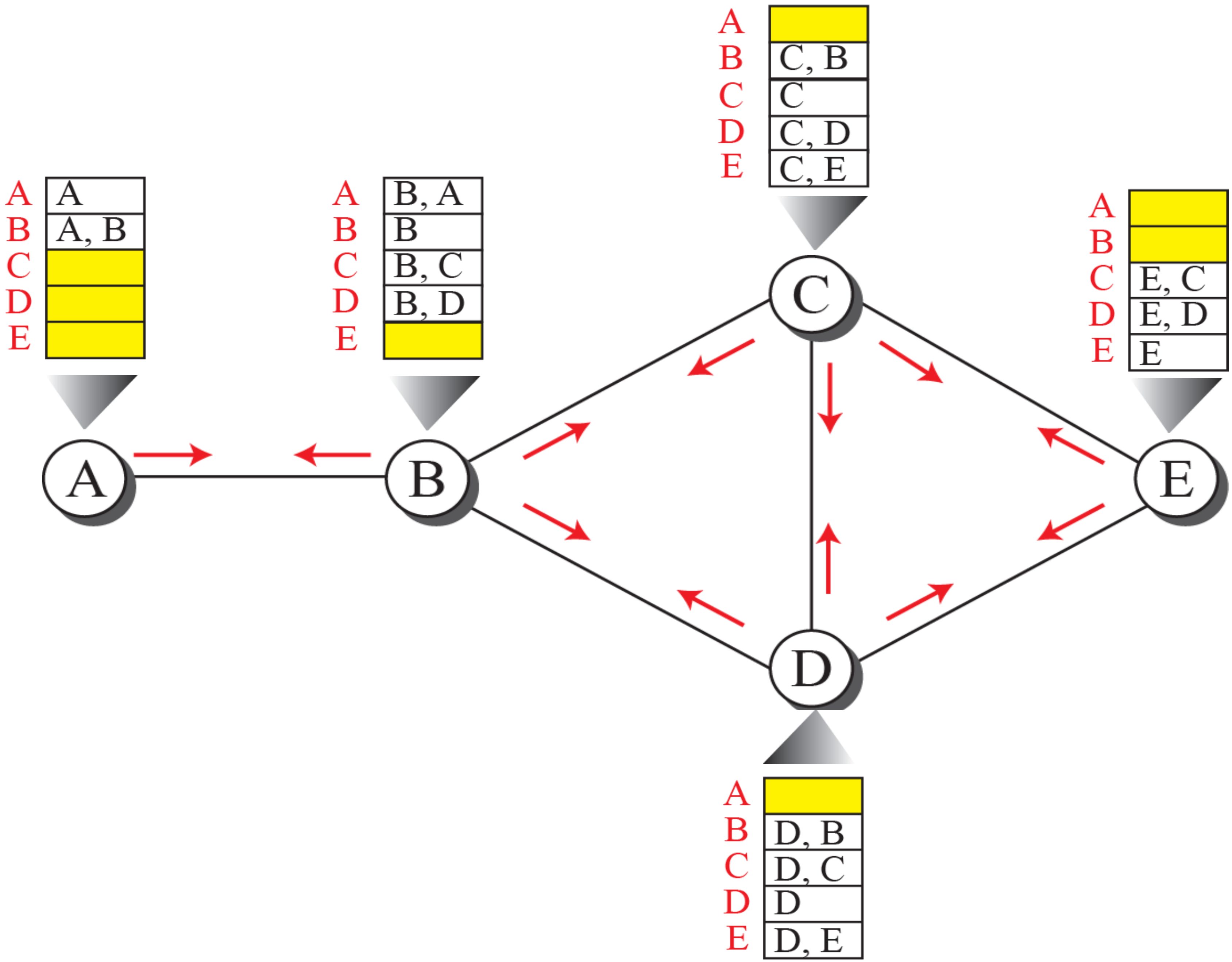
Path Vector Routing

- Add *path vector* for every destination
- Resembles *Distance Vector*
  - *Path Vector* contains info on entire path to destination
- Find ***best path*** among many possible
- *Policy Based Routing*
  - Only use paths through acceptable nodes
  - Do not use paths where self is a node (loop!)
  - Path vector length often most important parameter

Spanning trees in path-vector routing



Path vectors made at boot time



Updating path vectors

	New C	Old C	B
A	C, B, A		B, A
B	C, B	C, B	B
C	C	C	B, C
D	C, D	C, D	B, D
E	C, E	C, E	

C[ ] = best (C[ ], C + B[ ])

Event 1: C receives a copy of B's vector

Note:
X [ ]: vector X
Y: node Y

	New C	Old C	D
A	C, B, A	C, B, A	
B	C, B	C, B	D, B
C	C	C	D, C
D	C, D	C, D	D
E	C, E	C, E	D, E

C[ ] = best (C[ ], C + D[ ])

Event 2: C receives a copy of D's vector

More about path vector

Later:

- Autonomous systems, AS
- Routing between domains
- Policy routing
- Routing protocol BGP

Comparison of LS and DV algorithms

Message complexity

LS: with n nodes, E links, O(nE) msgs for full knowledge, changes sent to all nodes

DV: exchange between neighbours only

Speed of Convergence

LS:

- $O(n^2)$ algorithm requires O(nE) msgs
- may have oscillations

DV:

- Convergence time may be slower
- may be routing loops
- count-to-infinity problem