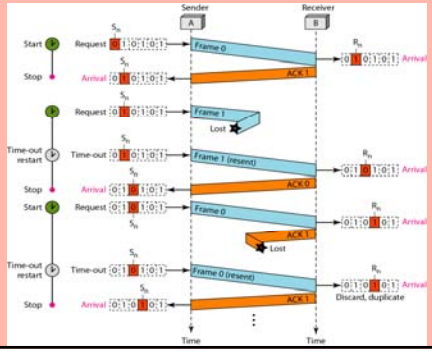
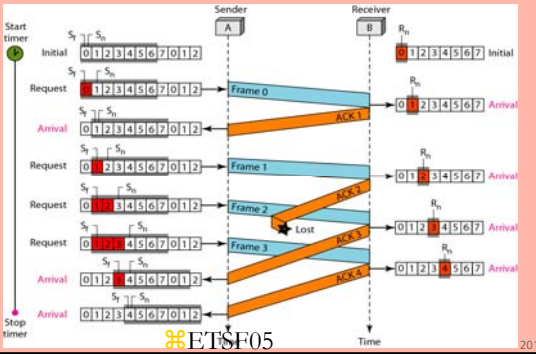
ETSF05:
Network models
Paradigmer
Länkprotokoll: Flödeskontroll vs felhantering
Routingalgoritmer

Jens A Andersson

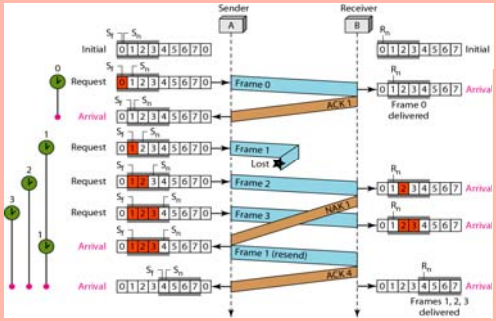
Stop-and-wait ARQ flow diagram



Go-back-N ARQ flow diagram



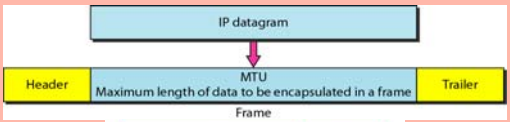
Selective repeat ARQ flow diagram



BONUS MATERIAL

2012-10-01

Maximum datagram size



Protocol	*MTU*
Hyperchannel	65,535
Token Ring (16 Mbps)	17,914
Token Ring (4 Mbps)	4,464
FDDI	4,352
Ethernet	1,500
X.25	576
PPP	296

2012-10-01

Fragmentation

Needed when IP datagram size > MTU

IPv4

- Performed by the router meeting the problem

IPv6

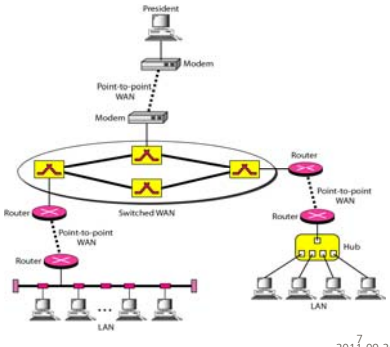
- Performed by the source router only

Defragmentation by destination host

2012-10-01

Network engineering

High performance
- Reliability
- Throughput
- Speed
- Security



Network models

Too complicated
- Divide and conquer

Layered architecture
- Hierarchy
- Specialisation
- Simplification

Figure 2.5: ***Communication through an internet***

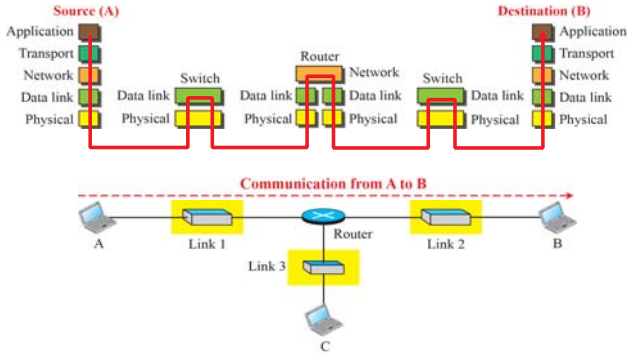
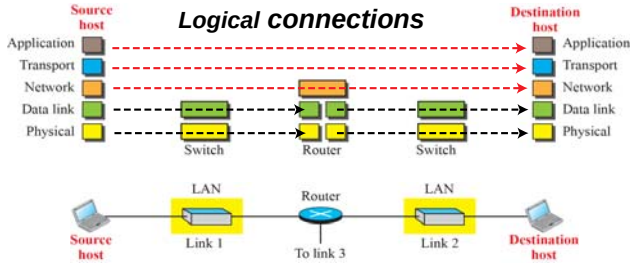
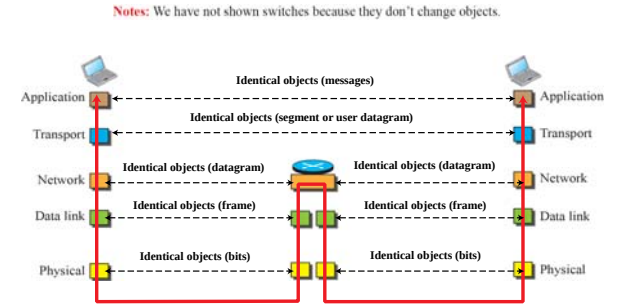


Figure 2.6: *Logical connections between layers in TCP/IP*



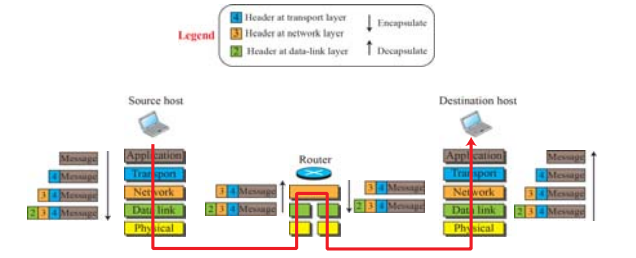
2.10

Figure 2.7: *Identical objects in the TCP/IP protocol suite*



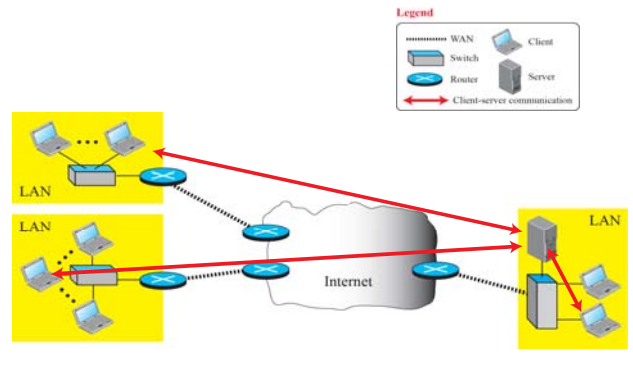
2.11

Figure 2.8: *Encapsulation / Decapsulation*



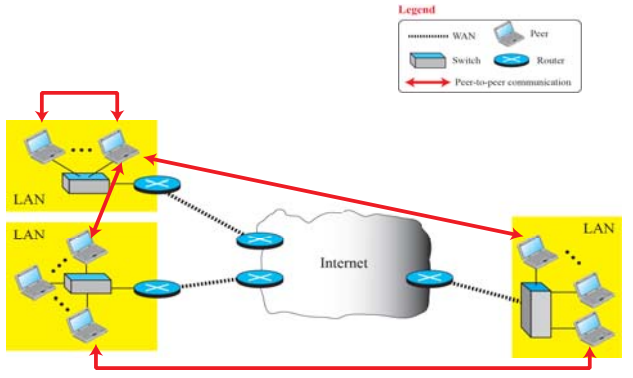
2.12

Figure 25.2: Example of a client-server paradigm



25.13

Figure 25.3: Example of a peer-to-peer paradigm



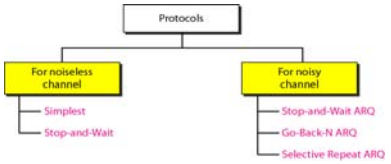
25.14

Data link control protocols

Framing

Flow control
- Send data
- Wait for ACK

Error control
- Detect error
- Retransmit



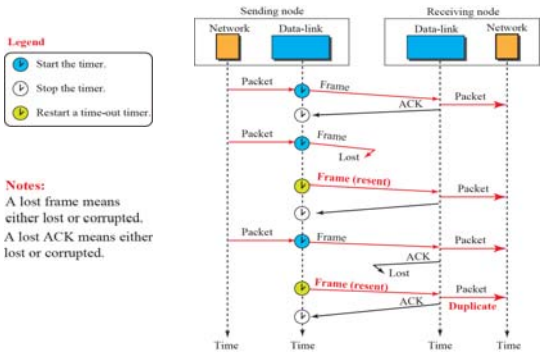
Inte alltid att båda (flödeskontroll, felhantering) finns i länklagret!

15
2011-09-26

Figure 111.5: ***Flow control at the data link layer***

11.16

Figure 111.12: ***Flow diagram for Example 111.3***



11.17

Go-Back-N?

- Viktigt inför nästa del och TCP (ETSF10)
- Föreläst i repetitionsföreläsningen
- Kommer på övningen

18

Routing

- Konsten att bygga least-cost trees
  - Från sändare till mottagare
  - Från varje nod till varje annan nod
- Tre principer
  - Distance Vector
  - Link State
  - Path Vector
    - Policy-based routing

19

Distance vector routing

All best known paths **shared to neighbours**

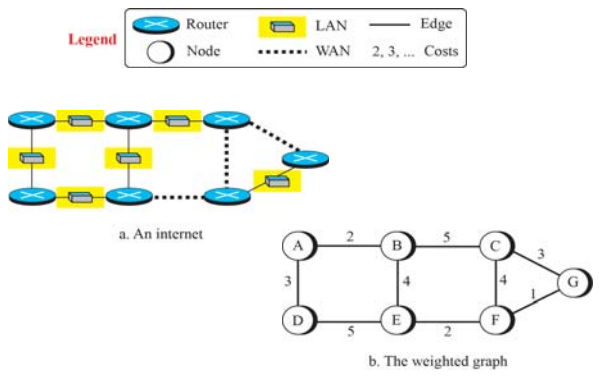
- Periodically
- Upon any change

Routing tables **updated** for

- New entries
- Cost changes

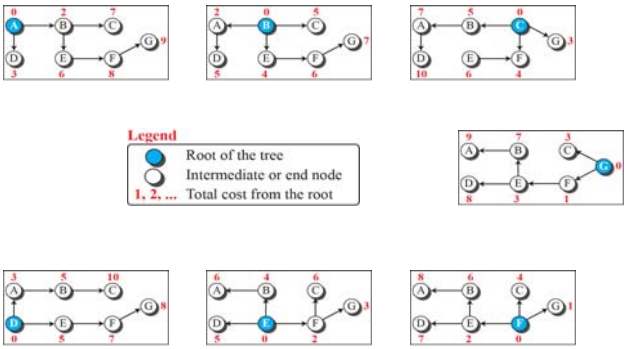
”Global knowledge shared locally”

Figure 20.1: An internet and its graphical representation



20.21

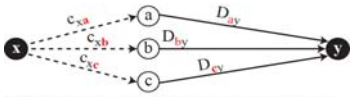
Figure 20.2: ***Least-cost trees for nodes in the internet of Figure 4.56***



Note! This is the final result when the network has converged

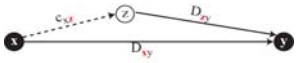
20.22

Figure 20.3: ***Graphical idea behind Bellman-Ford 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

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

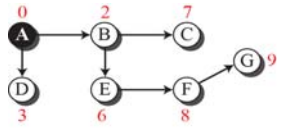
20.23

Updating algorithm (Bellman-Ford)

```
if (advertised destination not in table)
   {
   add new entry // rule #1
   }
else if (adv. next hop = next hop in table)
   {
   update cost // rule #2
   }
else if (adv. cost < cost in table)
   {
   replace old entry // rule #3
   }
```

2012-10-01

Figure 20.4: ***The distance vector corresponding to a tree***



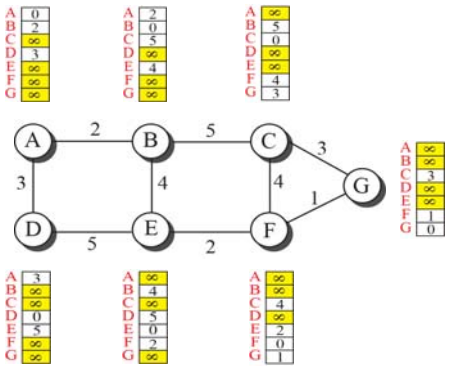
a. Tree for node A

	A
A	0
B	2
C	7
D	3
E	6
F	8
G	9

b. Distance vector for node A

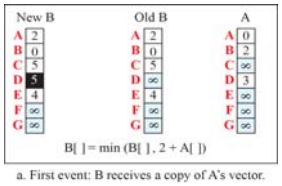
20.25

Figure 20.5: ***The first distance vector for an internet***



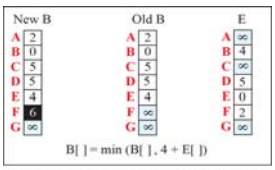
20.26

Figure 20.6: ***Updating distance vectors***



a. First event: B receives a copy of A's vector.

Note:
X[ ]: the whole vector



b. Second event: B receives a copy of E's vector.

- Periodiska uppdateringar!?
- Problem med länkar och noder som försvinner.

20.27

Mer om distance vector

I ETSF10:

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

Link state routing

Local topology info **flooded** globally

- Periodically (very seldom in practise)
- Upon any change

Create database with link states in each node

Routing tables updated for

- New entries
- Cost changes

”Local knowledge shared globally”

Initial link state knowledge

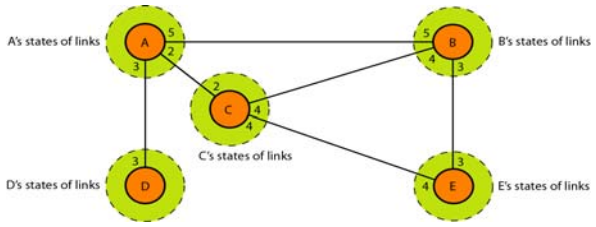
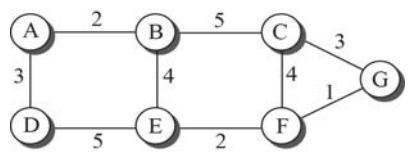


Figure 20.8: ***Example of a link-state database***



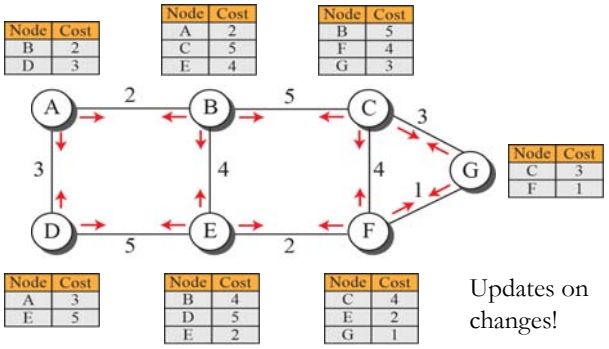
a. The weighted graph

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

20.31

⌘*Figure 20.9:* ***LSPs created and sent out by each node to build LSDB***



Updates on changes!

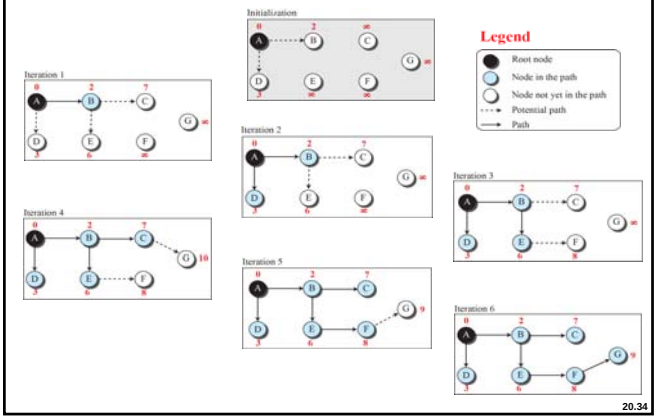
20.32

Tree generation algorithm (Dijkstra)

```
put yourself to tentative list
while tentative list not empty
   {
   pick node which can be reached
            with least cumulative cost
   add it to your tree*
   put its neighbours to tentative list**
            with cumulative costs to reach them
   }
                              *(a.k.a. permanent list)
            **(if not already there with lower cost)
```

33

Figure 20.10: ***Least-cost tree***



20.34

Mer om link state

I ETSF10:

- Routingprotokoll OSPF
- Konceptet Areor eller hur man minskar flooding

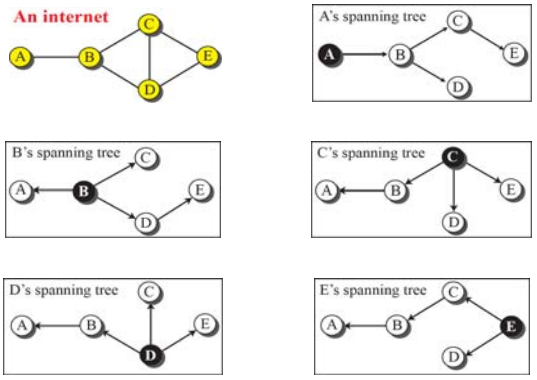
35

Path Vector Routing

- Policy Based Routing
- Select ***best path*** of multiple paths
- Similar to Distance Vector
- Add path vector for each destination
- Add only paths that passes through accepted nodes
- Do not add paths where own node is element
- Path Vector length is only/major discriminator

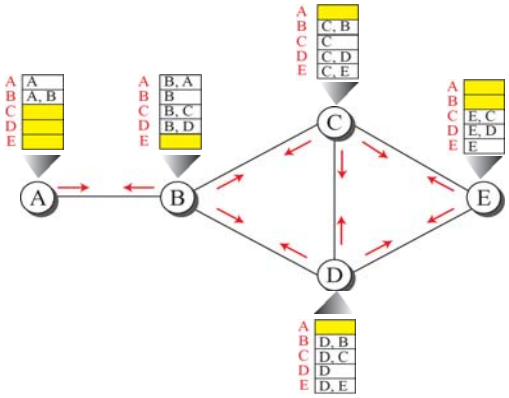
36

Figure 20.11: ***Spanning trees in path-vector routing***



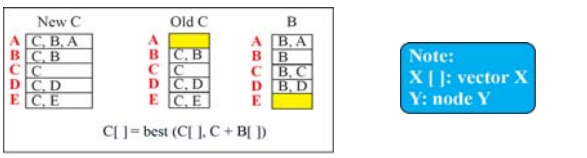
20.37

Figure 20.12: ***Path vectors made at booting time***



20.38

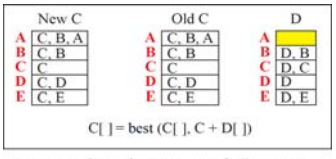
Figure 20.13: ***Updating path vectors***



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

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

"best" är implementering av policyn



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

20.39

Mer om path vector

I ETSF10:

- Autonuma system, AS
- Routing mellan domäner
- Policy routing
- Routingprotokoll BGP

40