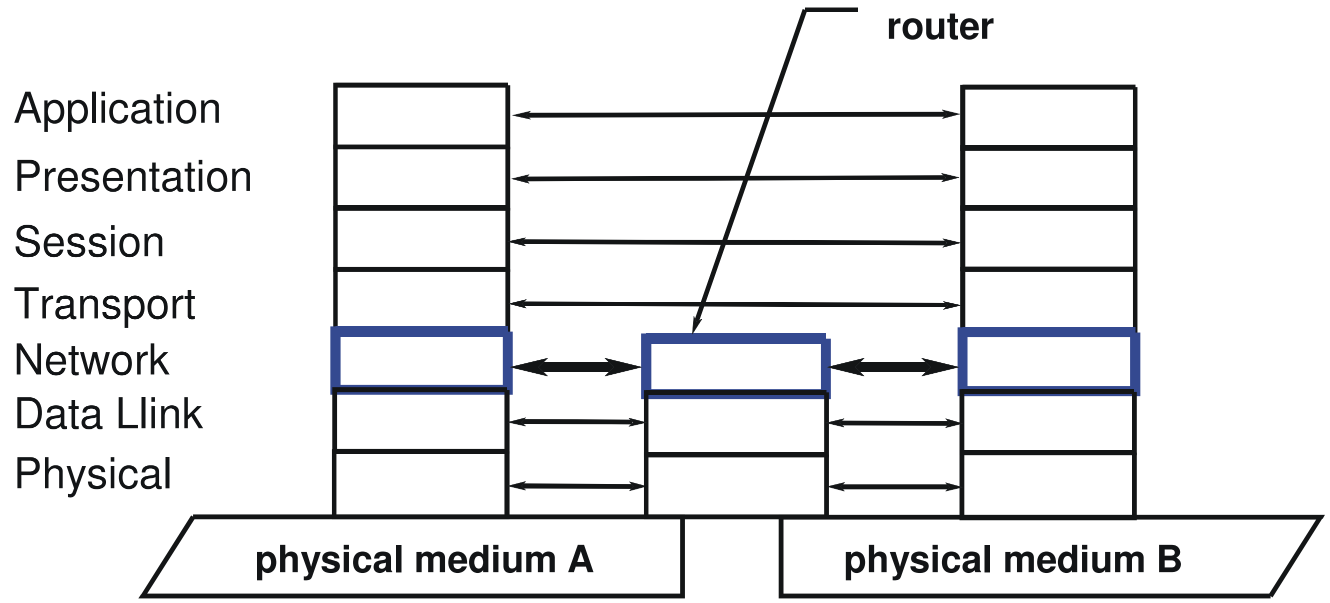
5 Wide Area Networks and Routing

5.1 Packet Switching

The Network Layer in the OSI reference model



ISO Definition for the Network Layer

The network layer provides the ability to establish, operate and terminate network connections between open systems over intermediate systems.

The network layer provides independence from routing and switching decisions.

Functions of the Network Layer

- Routing and switching of packets
- Multiplexing of end-to-end connections over a layer-2 connection
- Packet segmentation („fragmentation")
- In connection-oriented network layers additionally:
    - connection establishment and termination
    - error recognition and error correction (end-to-end)
    - guaranteeing the order of the packets
    - flow control (end-to-end)

Heterogeneous subnetworks can be interconnected by a network layer instance ("Internetworking").

5.2 Virtual Circuits vs. Datagrams

Virtual Circuit

The path through the network is determined when the connection is established, i.e., for each new virtual connection a routing decision takes place in each node only once. The entire traffic flowing over this virtual connection takes the same path through the network.

Datagram

Every packet contains the full address of the destination host. When a packet arrives at an intermediate node the destination address is used to determine the outgoing link for the next hop.

Virtual Circuit

"Perfect" channel through the network

- Error control (bit errors, lost and duplicated packets)
- Flow control
- order of messages maintained

Phases

- Connection establisment
- Data transmission
- Connection termination

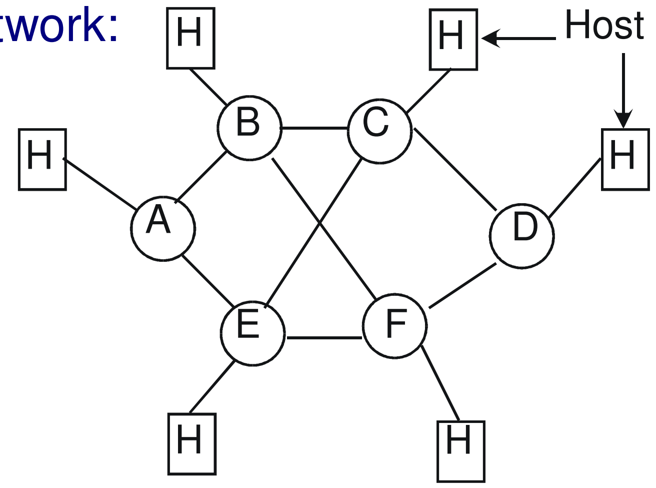
Advantages

- Low overhead for addresses in the data transmission phase
- Low computational overhead for routing in the data phase
- High quality of the arriving packet stream

Implementation of Virtual Circuits

Tables with status information on all existing virtual circuits are maintained in each node.

(a) Example of a subnetwork:

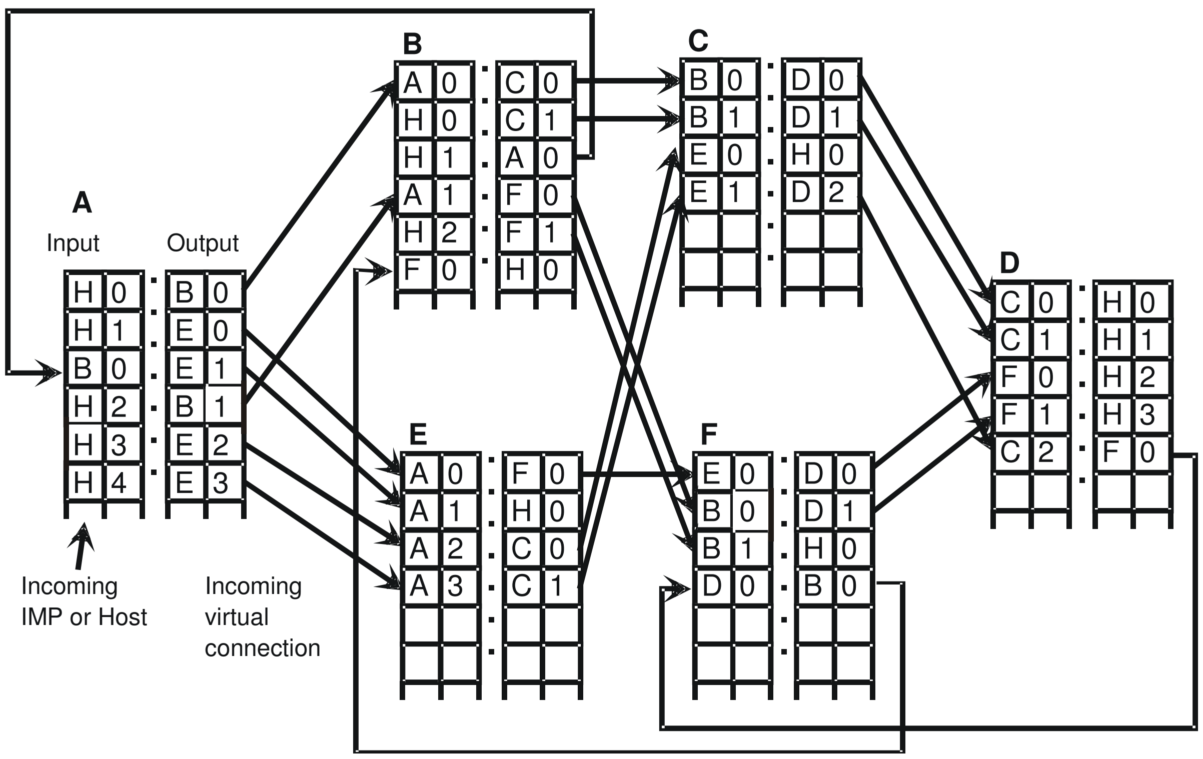


(b) Eight virtual connections through this subnetwork:

Starting from A	Starting from B
0 – ABCD	0 – BCD
1 – AEFD	1 – BAE
2 – ABFD	2 – BF
3 – AEC	
4 – AECDFB	

Status Information in the Nodes

(c) **Routing tables** in the nodes



The Datagram

Every packet (datagram) is considered an isolated unit (like a telegram):

- Full destination address contained in every packet
- Packets can arrive out of order
- No error control, no flow control in layer 3

Advantages

- Simpler than virtual circuits, therefore easier to implement
- No connection establishment and termination phases: low overhead for short-lived connections
- More reliabe since there are no status clean-up and recovery problems when a node or link fails
- Better suitable for internetworking of heterogeneous subnetworks

5.3 Routing in Point-to-Point Networks

Topology

We assume that the network topology is a graph.

In LANs that have a broadcast topology (bus, ring) there is no routing!

Routing Algorithms (1)

Task

To route the packets through the network from the source end-system to the destination end-system

The routing algorithm decides to which outgoing link of each router an incoming packet is transferred.

Desirable characteristics of a routing algorithm

- Correct
- Simple
- Robust in case of node or link loss
- Fair
- Optimal (finds the best route, causes minimal overhead)

Routing Algorithms (2)

The design criteria are in conflict. In practice a good approximation is: **Minimization of hops** from the source to the destination.

Classification of Routing Algorithms

1. Static (non-adaptive) Routing

- no consideration of the current network conditions
- For all i, j the routes between i and j are determined before start-up of the network
- no change during operation

2. Adaptive Routing

- Decisions are based on the current network topology (and sometimes load)
- Continuous measurement of the topology and the traffic
- Can be further classsified into
  - **centralized** routing algorithms
  - **isolated** routing algorithms
  - **distributed** routing algorithms

Static Routing

In **static routing** the entire topology of the network is known to a central node. It computes the optimal paths for each pair (i,j) of nodes, generates routing tables for the individual nodes and sends them out.

Static routing is only possible if the network is relatively small and changes to the topology are rare.

Multipath Routing

Alternative routes are computed for each pair of nodes (i,j). The probability of use depends on the relative weights. Multipath routing is more reliable in case of node or link failures, and balances the load better. However is is more complex than single-path routing.

Multipath Routing (1)

Every node contains a routing table with one row for every destination:

D	O_1	W_1	O_2	W_2		O_n	W_n

D	Destination
O_i	i^{th} best outgoing link
W_i	Weight for O_i
	W_i is the probability that O_i is used.

$$\left(\sum_{i=1}^{n} W_i = 1 \right)$$

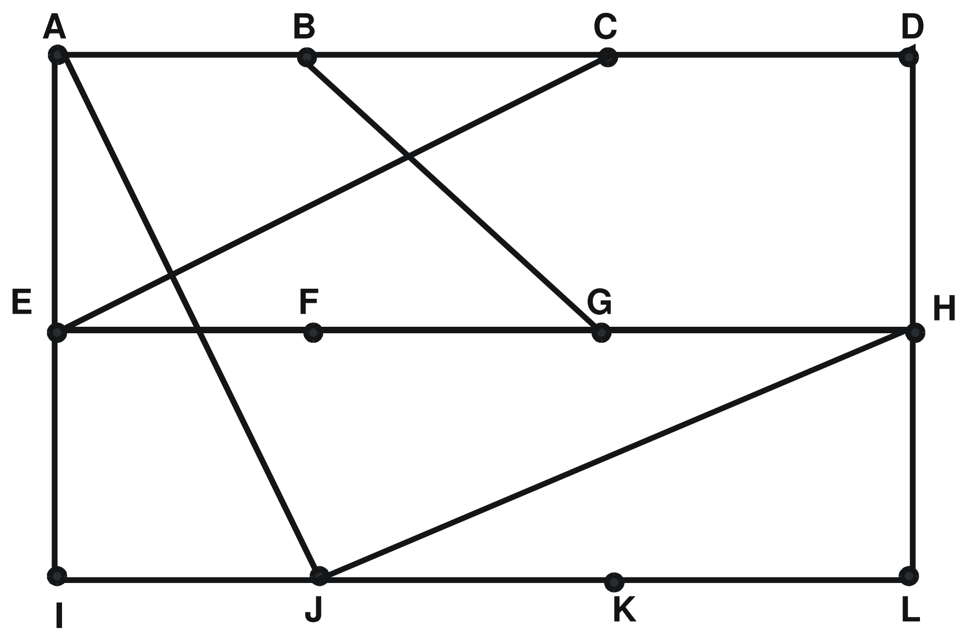
Multipath Routing (2)

Selection of an alternative:

```
generate a random number z ($0 <= z <= 1$)
choose $O_1$ if $0 <= z < W_1$
choose $O_2$ if $W_1 <= z < W_1 + W_2$
...
choose $O_n$ if $(W_1 + W_2 + .. + W_{n-1}) <= z <= 1$
```

Static Routing: Example Topology

Topology of the example network



We consider the paths beginning at node J.

Static Routing: Example Table

Static routing table with alternative paths for node j

Des.	1st choice		2nd choice		3rd choice	
A	A	0.63	I	0.21	H	0.16
B	A	0.46	H	0.31	I	0.23
C	A	0.34	I	0.33	H	0.33
D	H	0.50	A	0.25	I	0.25
E	A	0.40	I	0.40	H	0.20
F	A	0.34	H	0.33	I	0.33
G	H	0.46	A	0.31	K	0.23
H	H	0.63	K	0.21	A	0.16
I	I	0.65	A	0.22	H	0.13
-						
K	K	0.67	H	0.22	A	0.11
L	K	0.42	H	0.42	A	0.16

Determination of Routing Tables

In static routing routing tables are pre-computed by the network operator. Before network start-up they are loaded into the nodes and not changed anymore.

Characteristics

- simple
- good results for a constant topology and constant network traffic

But:

- inappropriate for strongly varying traffic and changes in topology
- inappropriate for large networks (does not scale well)

Still occasionally used In practice, for example in SNA networks.

The network operator always knows the entire topology. He/she can use Dijkstra‘s "Shortest Path" algorithm once for each node for the construction of the routing tables.

Centralized Adaptive Routing (1)

Principle

- There is a Routing Control Center (RCC) in the network.
- Each node periodically sends status information to the RCC, for example
  - list of immediate neighbors
  - current queue lengths
  - utilization of its links.
- The RCC collects the information and computes the optimal path for each pair of nodes, computes the individual routing tables and distri-butes them to the nodes.

Centralized Adaptive Routing (2)

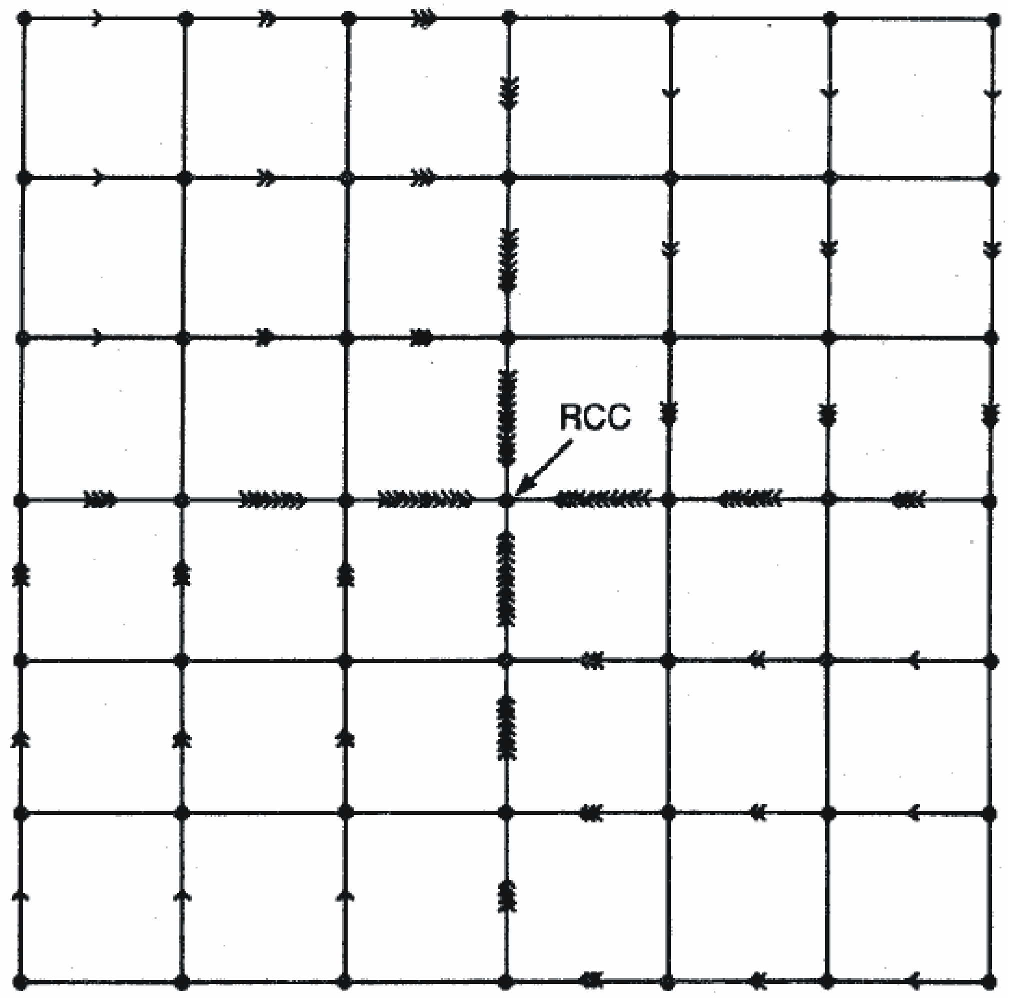
Characteristics

- The RCC has complete information => decisions are optimal
- The individual nodes don‘t have to do the routing computation.

But:

- Route computation has to take place frequently (say, once per minute).
- There will be a traffic concentration in the proximity of the RCC, thus a performance bottleneck.
- The technology is not robust: the RCC is a single point of failure.
- The algorithm fails when the network gets partitioned.
- The individual nodes receive new routing tables at different times => inconsistencies and thus "routing loops" will occur.

The Routing Control Center



Isolated Adaptive Routing

Principle

- No exchange of routing information between nodes
- Decisions are based on local information only

Examples of algorithms

- Backward Learning
- Flooding

Algorithm "Backward Learning"

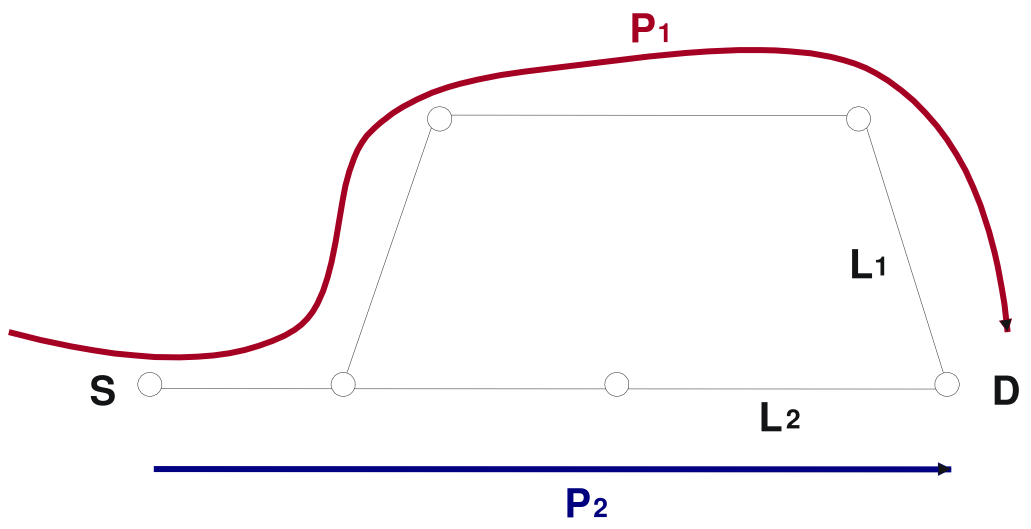
- A node "learns" from the arriving packets:
  packet (..., S, H, ...) with S = source node, H = hop counter,
  packet is received on link L => S is reachable over L in H hops
- Routing table in the node: Each entry is a triple

  (destination node, outgoing link, H_{min})
- Updating of the routing table:

  Node receives packet (..., S, H...) on link L

```
if not (S in table)
    then  add(S,L,H)
    else  if H < $H_{min}$
          then update(S,L,H)
```

Backward Learning: Example



```
P1( ..., S,4,...) -> add(S,L₁,4)
P2( ..., S,3,...) -> update(S,L₂,3)
```

Backward Learning: Path Degradation

Problem

Algorithm does not adapt to path degradations.

Solution

Periodic deletion of routing tables. A new learning period begins.

But: how often?

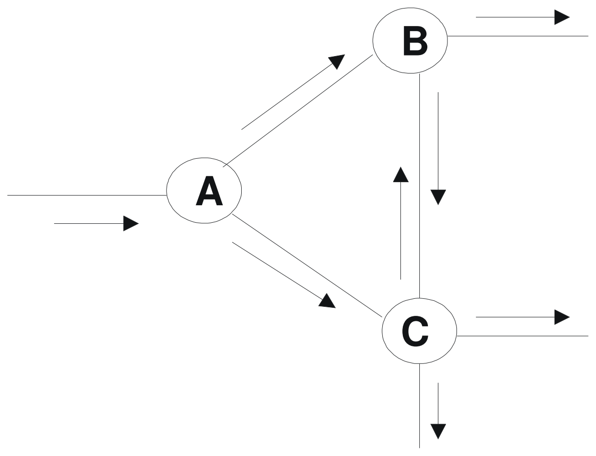
- too often: network is in the learning phase most of the time
- too seldom: slow reaction to degradations

Also:

We cannot send to a node from which we have never seen a packet.

Algorithm "Flooding"

An incoming packet is forwarded on all outgoing links except the one it came from.



Flooding: Silencing the Packet Flow

Problem: explosion of the number of packet copies in the network

Fading out the process

- put a hop counter into the packet header
- initialize it with the diameter of the network (= longest path in the network (worst case))
- decrement the hop counter on each hop
- copies receive the hop counter of the original
- counter = 0: packet is dropped by the router

Characteristics of Flooding

- very robust, very simple, but
- large number of copies, heavy network load

  => employed only for special applications or in a first phase of other routing algorithms to create initial status information

Distributed Routing (1)

Principle

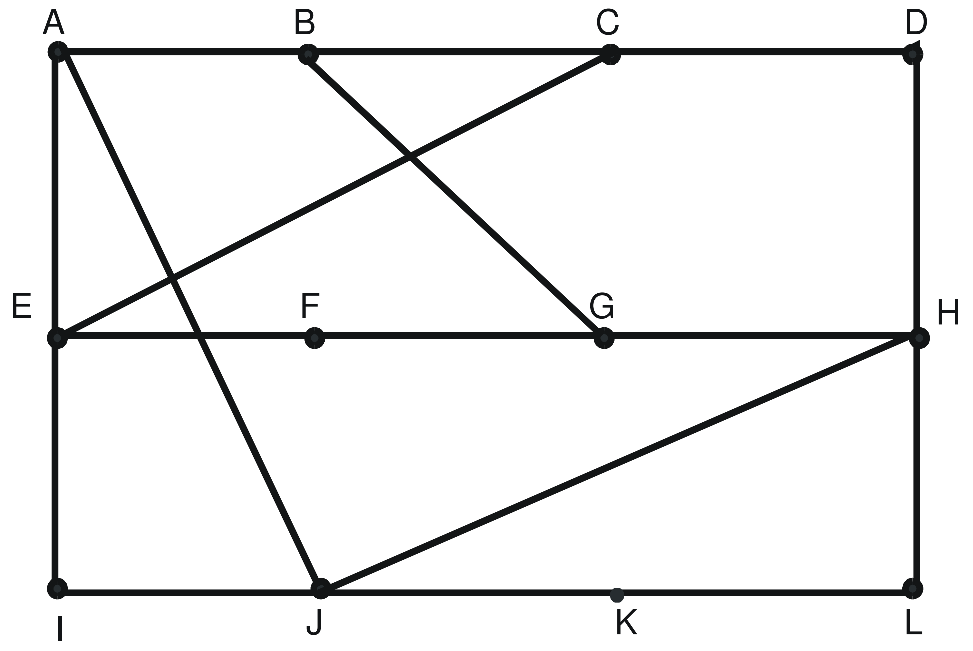
The nodes explicitly exchange routing information with their neighbors:

- Each node knows his distance to each neighbor:
  - number of hops (= 1)
  - delay (round-trip time)
  - queue length, etc.
- Each node periodically sends a list with his estimated distances to all known destination nodes to his neighbors.
- node X receives a list E from neighbor Y
- distance (X, Y) = e
- distance (Y, Z) = E(Z)
- => distance (X, Z) over Y is E(Z) + e

The table with these distances is called **distance vector**. The algorithm is thus called **distance vector routing**.

Distributed Routing (2)

Example



We consider the distances known to node J.

Distributed Routing (3)

	A	I	H	K	new DV	
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	-
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay=8	JI delay=10	JH delay=12	JK delay=6		

Right column: newly determined distances at node J after receiving the distance vectors from the neighbors

Hierarchical Routing

The size of the routing tables is proportional to the size of the network:

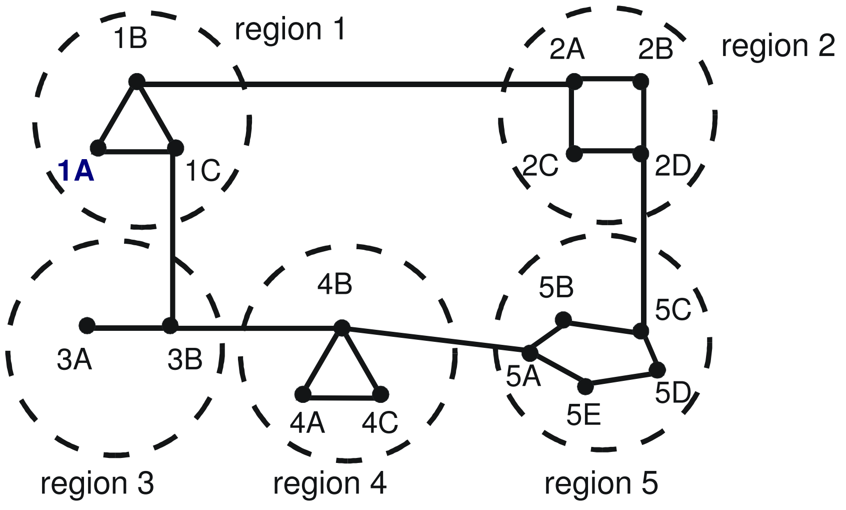
- large memory requirement in the nodes
- considerable CPU time for searching the tables
- much bandwidth for the exchange of routing information

Hierarchical routing helps to solve these problems:

- Nodes are grouped into regions
- Each node knows
  - all details of his region
  - his routes to all other regions
- In the Internet a “region” is a subnetwork of the IP address space.

Disadvantage: globally optimal decisions are no longer possible.

Example for Hierarchical Routing



full table for node 1A

DES.	LINE	HOP
1A	-	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

hierarchical table for node 1A

DES.	LINE	HOP
1A	-	-
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

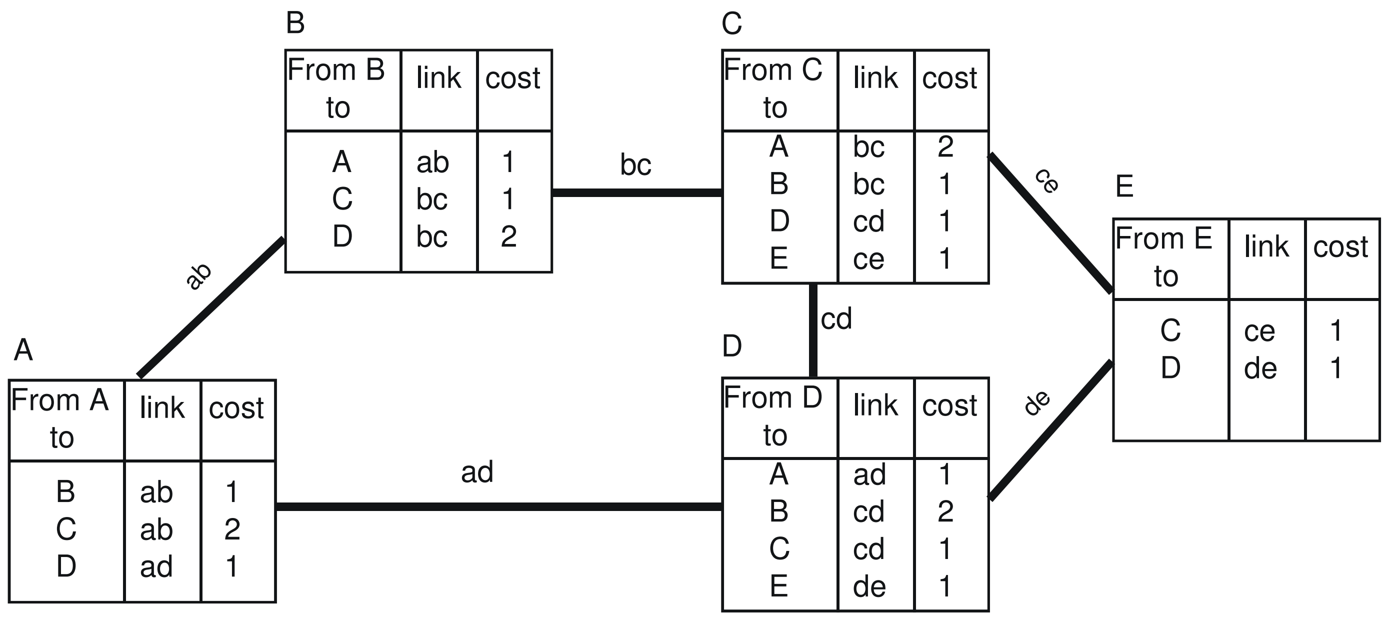
Routing in the Internet

Distance Vector Routing

In the early years of the internet the most commonly used procedure was an adaptive distributed procedure on the basis of distance vectors (distance vector routing). The employed protocol is called **RIP** (Routing Information Protocol).

With RIP all internet routers periodically exchange distance vector messages and update their routing tables accordingly.

Example for Distance Vector Routing (1)



B

From B to	link	cost
A	ab	1
C	bc	1
D	bc	2

C

From C to	link	cost
A	bc	2
B	bc	1
D	cd	1
E	ce	1

A

From A to	link	cost
B	ab	1
C	ab	2
D	ad	1

D

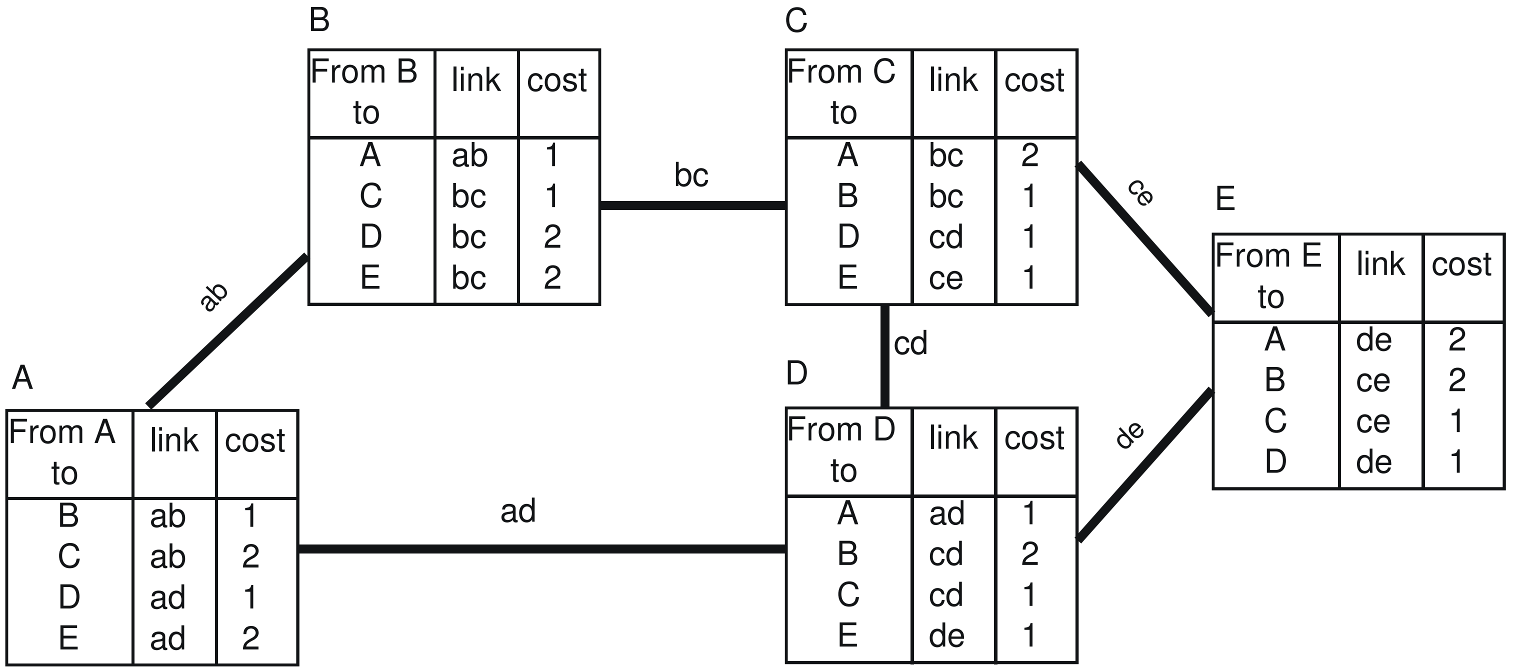
From D to	link	cost
A	ad	1
B	cd	2
C	cd	1
E	de	1

E

From E to	link	cost
C	ce	1
D	de	1

(a) node E has just been added to the network

Example for Distance Vector Routing (2)



A

From A to	link	cost
B	ab	1
C	ab	2
D	ad	1
E	ad	2

B

From B to	link	cost
A	ab	1
C	bc	1
D	bc	2
E	bc	2

C

From C to	link	cost
A	bc	2
B	bc	1
D	cd	1
E	ce	1

D

From D to	link	cost
A	ad	1
B	cd	2
C	cd	1
E	de	1

E

From E to	link	cost
A	de	2
B	ce	2
C	ce	1
D	de	1

(b) after an exchange of RIP messages

OSPF Routing

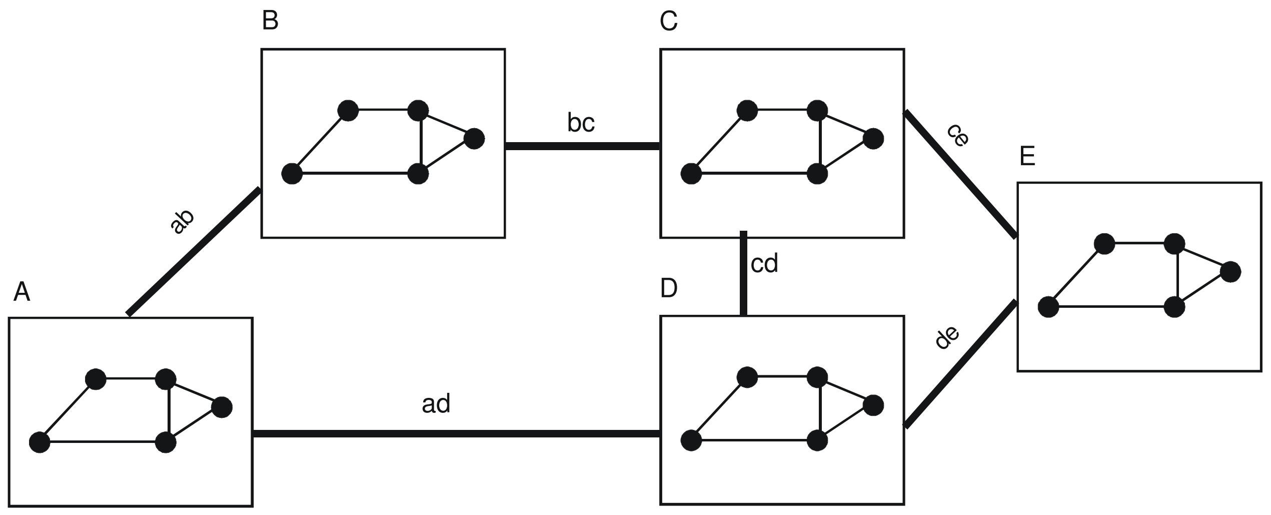
Another important and widely used routing algorithm on the Internet is **OSPF** (**Open Shortest Path First**). The basic idea is that all nodes know the entire network topology at all times and can compute all optimal paths locally.

If the topology changes, the nodes exchange topology update messages. Each node maintains a local “database“ of the entire topology, called the **link state database**.

The optimal paths to all destinations can be computed locally with **Dijkstra‘s Shortest Path algorithm** (Shortest Path First = SPF). In the Internet slang the algorithm is thus called “Open Shortest Path First“.

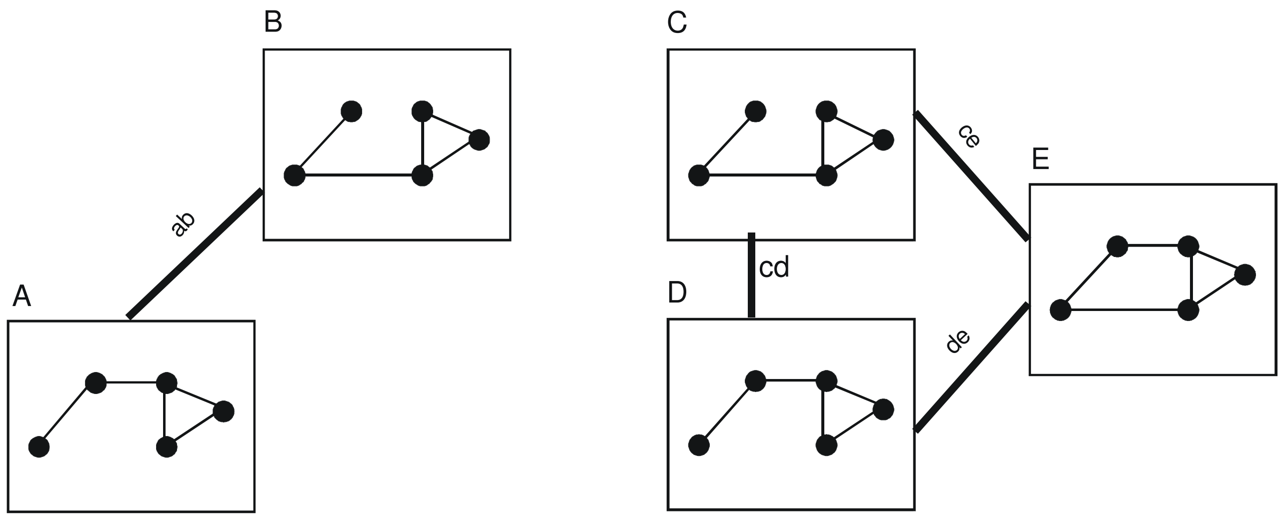
Algorithms of this class are called ”link state routing algorithms”.

Example for OSPF Routing (1)



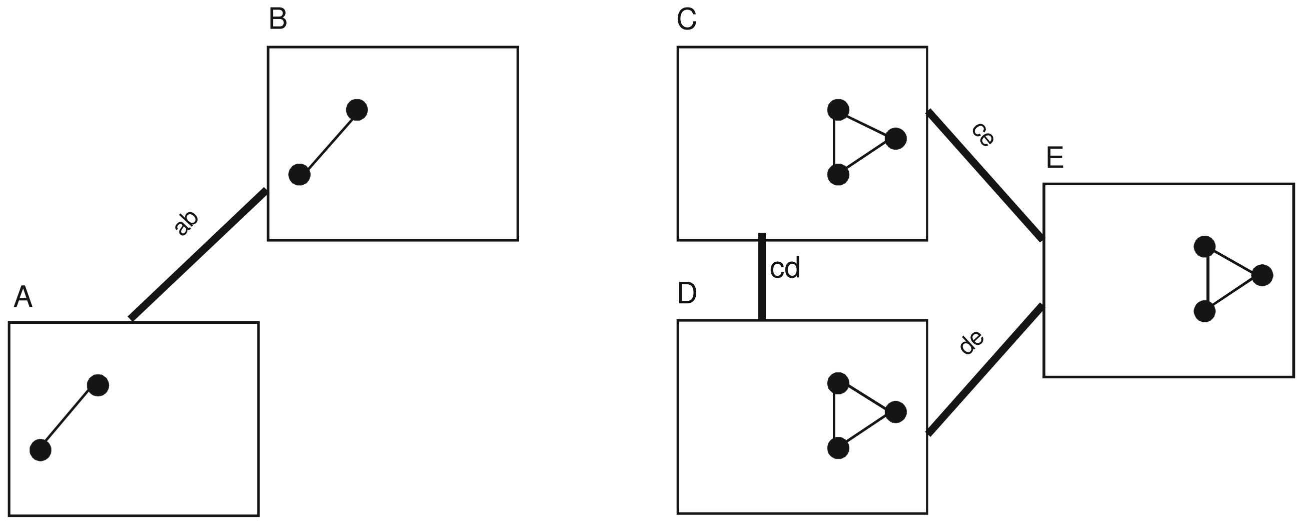
(a) network in stable condition

Example for OSPF Routing (2)



(b) links bc and ad failed

Example for OSPF Routing (3)



(c) after an exchange of OSPF messages

Conclusions

- Routing is done at layer 3, the Network Layer.
- At runtime a router forwards the incoming packets on the basis of a **routing table**.
- Hierachical routing helps to solve the scalablility problem for large networks. In the Internet a region corresponds to an IP subnetwork.
- A **routing algorithm** computes the entries of the routing tables.
- Static routing can only be used for small networks with little change in topology and traffic.
- Adaptive routing with a centralized routing control center has the dis-advantage of a central performance bottlenech and a central point of failure.
- The Internet uses adaptive distributed routing algorithms. The most popular ones are **RIP** (older, based on **distance vectors**) and **OSPF** (currently used, based on a local **link state database**).