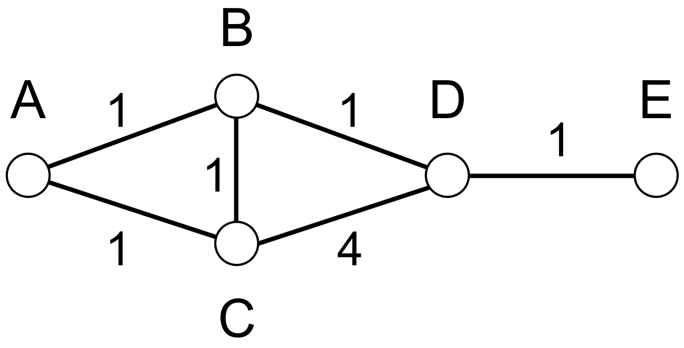
Intra-domain Routing
Distance Vector (DV) Principle

Distance Vector Principle

- every router stores information about the distance (cost) to other routers (= distance vector)
- every router transmits these distance information periodically to its neighbouring routers
- every router creates - based on these distance information - step by step its routing table, by means of an simple algorithm (Bellman-Ford algorithm)
- after convergence of the algorithm the routing table contains information about reachable destinations and the next hop routers which are on the shortest path towards these destinations

Network example:



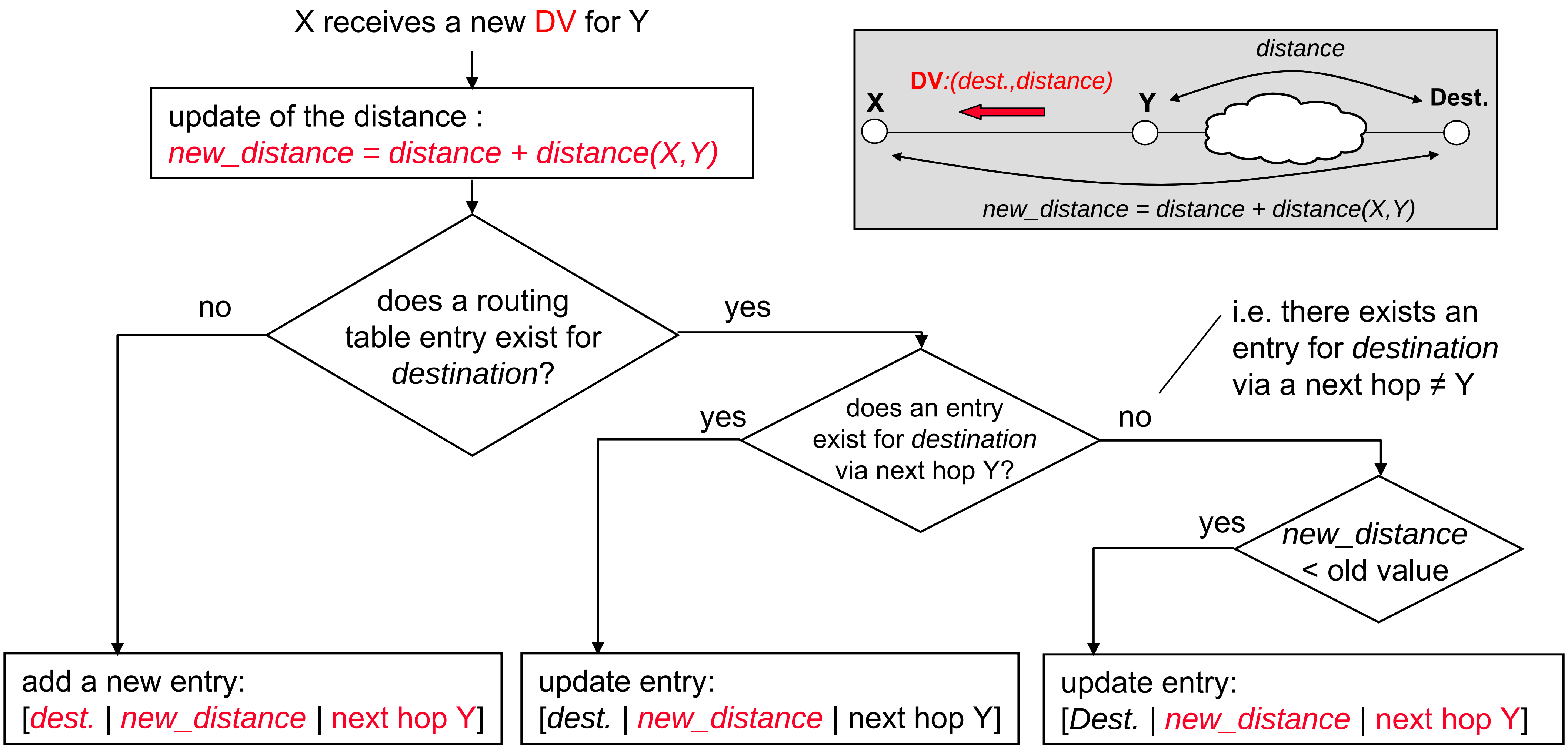
Routing table for router A:

distance vector for router A

Dest.	Dist.	next hop
B	1	B
C	1	C
D	2	B
E	3	B

Route Calculation with Bellman-Ford Algorithm

Rule for routing table update after reception of a new DV – to be executed for every component *(destination, distance)* of the DV



X receives a new DV for Y

update of the distance :
new_distance = distance + distance(X,Y)

does a routing table entry exist for *destination*?

no

yes

does an entry exist for *destination* via next hop Y?

yes

no

i.e. there exists an entry for *destination* via a next hop ≠ Y

new_distance < old value

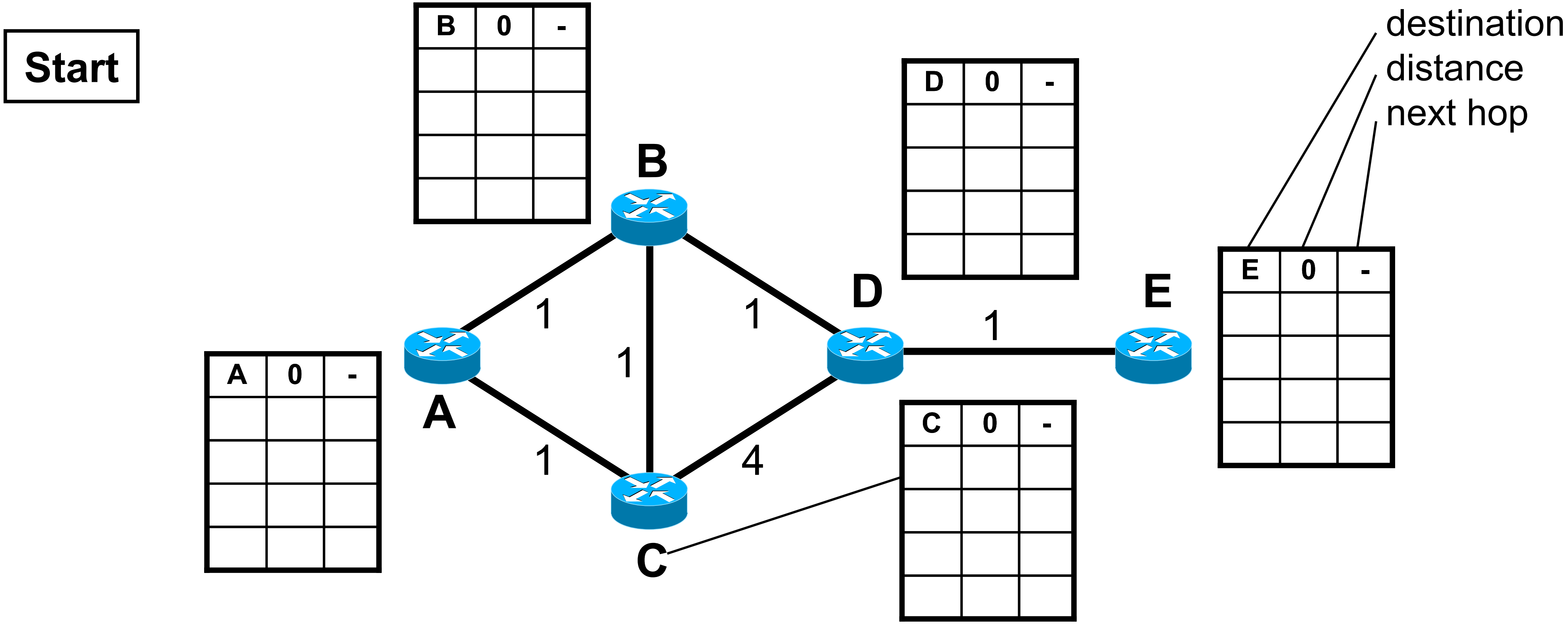
yes

add a new entry:
[*dest.* | *new_distance* | next hop Y]

update entry:
[*dest.* | *new_distance* | next hop Y]

update entry:
[*Dest.* | *new_distance* | next hop Y]

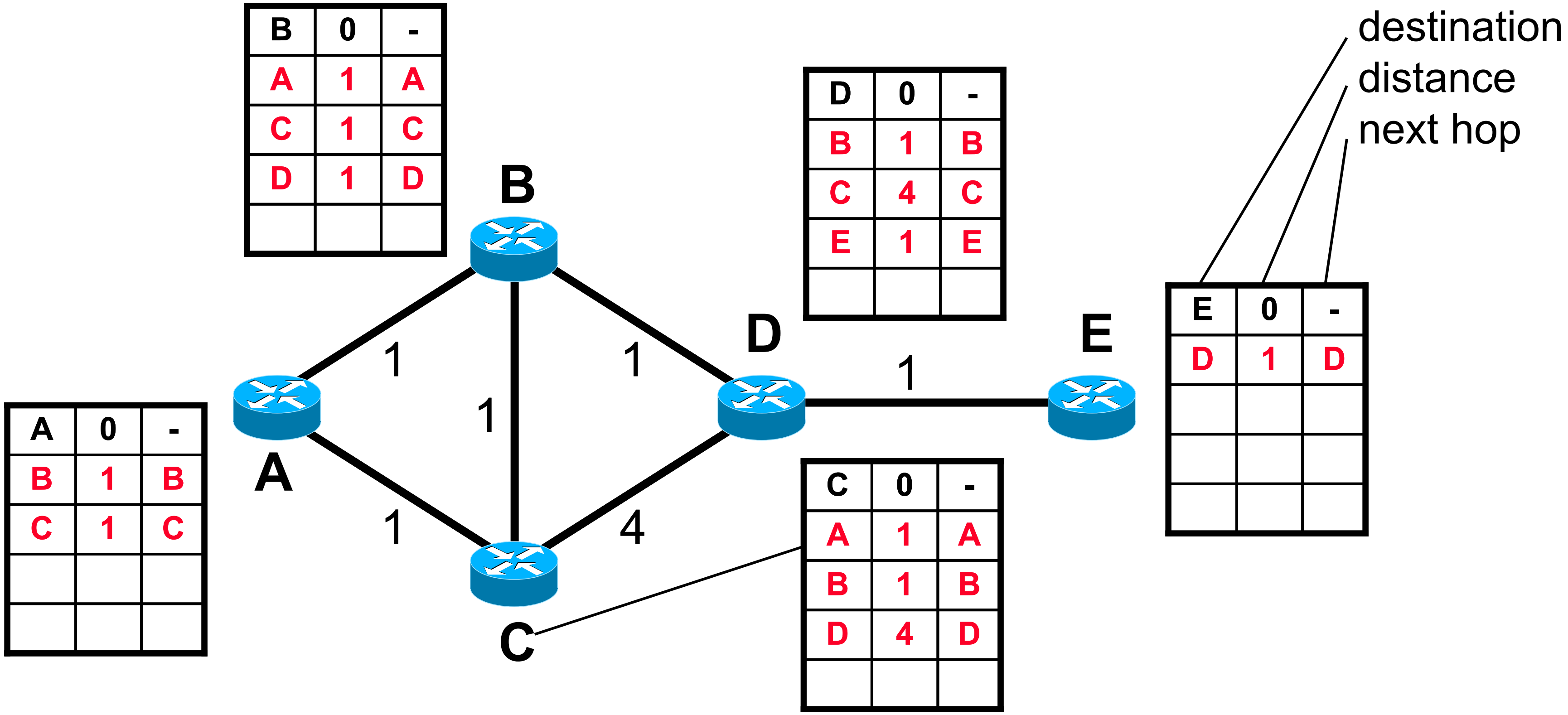
Example: Behaviour at Start (1)



Sending of DVs:
- A sends DV (A-0) to B and C
- B sends DV (B-0) to A, C and D
- C sends DV (C-0) to A, B and D
- D sends DV (D-0) to B, C and E
- E sends DV (E-0) to D

Example: Behaviour at Start (2)

Time T_1



B	0	-
A	1	A
C	1	C
D	1	D

D	0	-
B	1	B
C	4	C
E	1	E

E	0	-
D	1	D

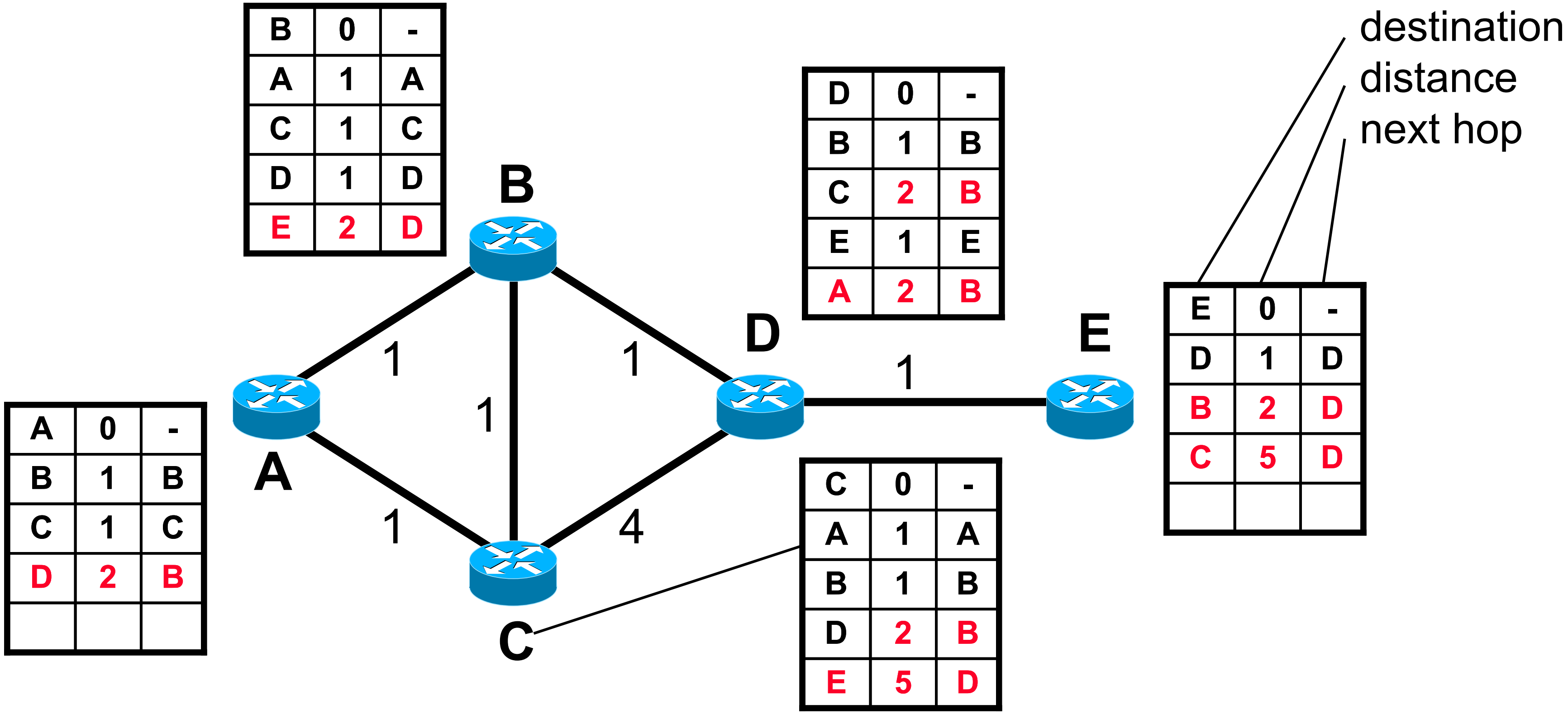
A	0	-
B	1	B
C	1	C

C	0	-
A	1	A
B	1	B
D	4	D

Sending of DVs:
- A sends DV (A-0, B-1, C-1) to B and C
- B sends DV (B-0, A-1, C-1, D-1) to A, C and D
- C sends DV (C-0, A-1, B-1, D-4) to A, B and D
- D sends DV (D-0, B-1, C-4, E-1) to B, C and E
- E sends DV (E-0, D-1) to D

Example: Behaviour at Start (3)

Time T_2



Router B:

B	0	-
A	1	A
C	1	C
D	1	D
E	2	D

Router D:

D	0	-
B	1	B
C	2	B
E	1	E
A	2	B

Router E:

E	0	-
D	1	D
B	2	D
C	5	D

Router A:

A	0	-
B	1	B
C	1	C
D	2	B

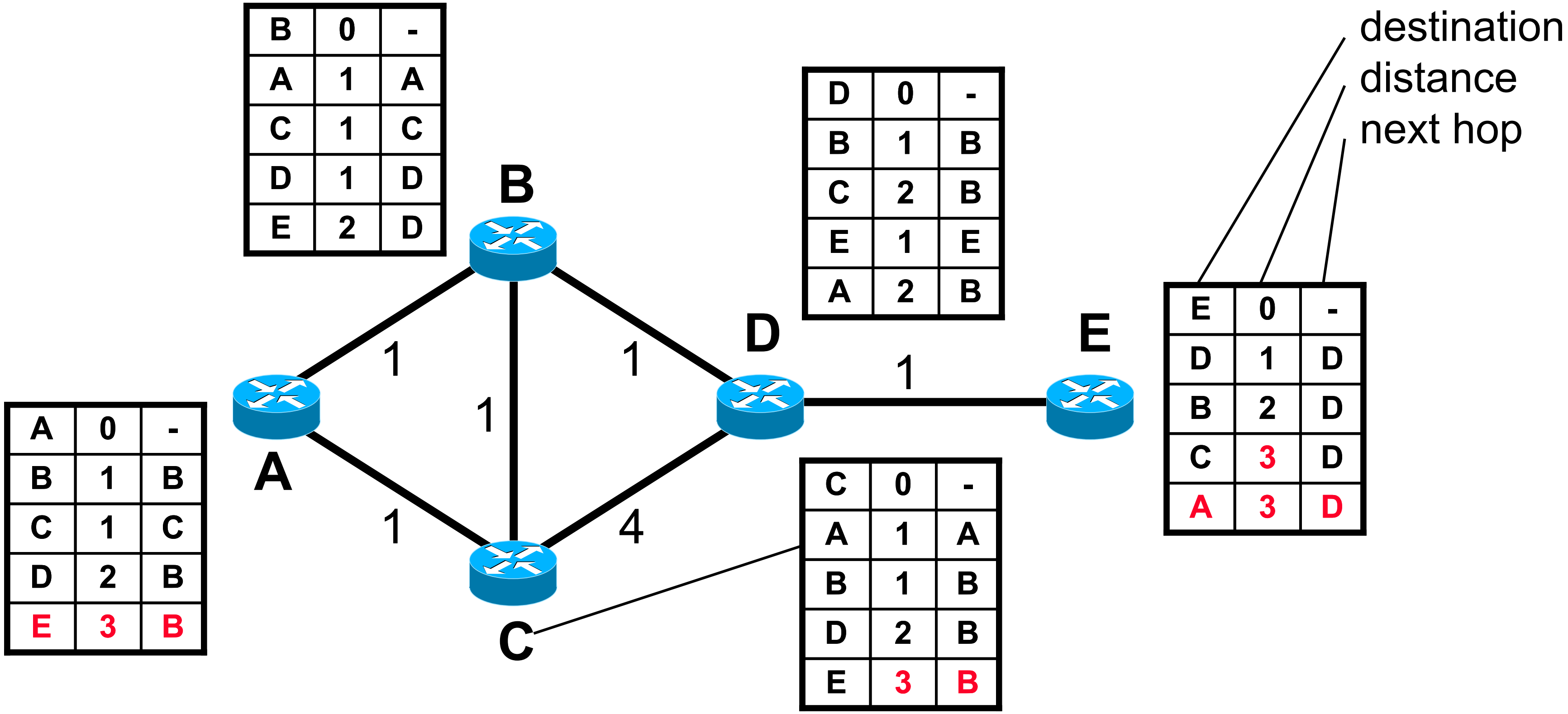
Router C:

C	0	-
A	1	A
B	1	B
D	2	B
E	5	D

Sending of DVs:
- A sends DV (A-0, B-1, C-1, D-2) to B and C
- B sends DV (B-0, A-1, C-1, D-1, E-2) to A, C and D
- C sends DV (C-0, A-1, B-1, D-2, E-5) to A, B and D
- D sends DV (D-0, B-1, C-2, E-1, A-2) to B, C and E
- E sends DV (E-0, D-1, B-2, C-5) to D

Example: Behaviour at Start (4)

Time T_3



B	0	-
A	1	A
C	1	C
D	1	D
E	2	D

D	0	-
B	1	B
C	2	B
E	1	E
A	2	B

E	0	-
D	1	D
B	2	D
C	3	D
A	3	D

A	0	-
B	1	B
C	1	C
D	2	B
E	3	B

C	0	-
A	1	A
B	1	B
D	2	B
E	3	B

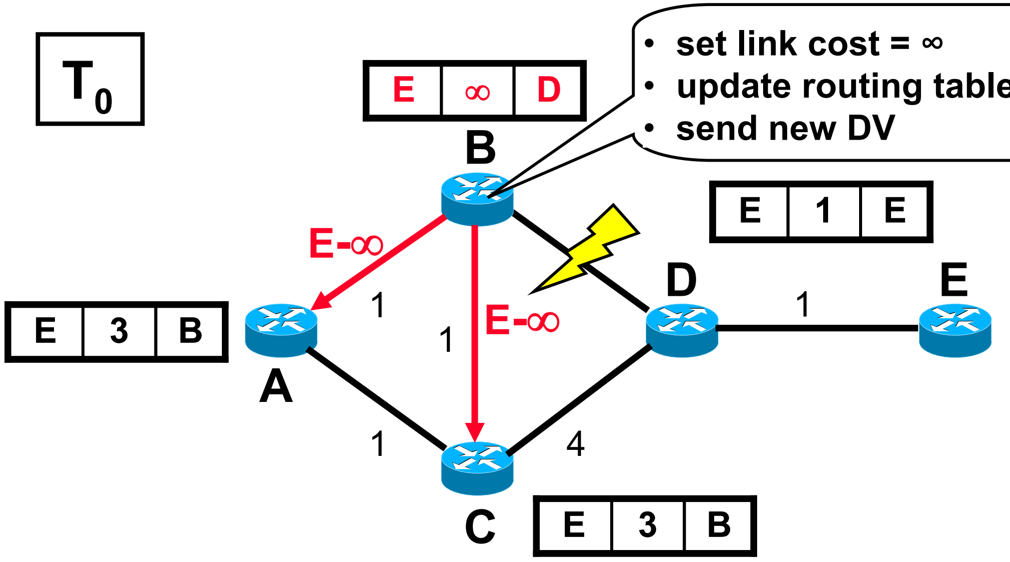
Sending of DVs:
- A sends DV (A-0, B-1, C-1, D-2, E-3) to B and C
- B sends DV (B-0, A-1, C-1, D-1, E-2) to A, C and D
- C sends DV (C-0, A-1, B-1, D-2, E-3) to A, B and D
- D sends DV (D-0, B-1, C-2, E-1, A-2) to B, C and E
- E sends DV (E-0, D-1, B-2, C-3, A-3) to D

Example: Behaviour after Change of Network Topology* (1)

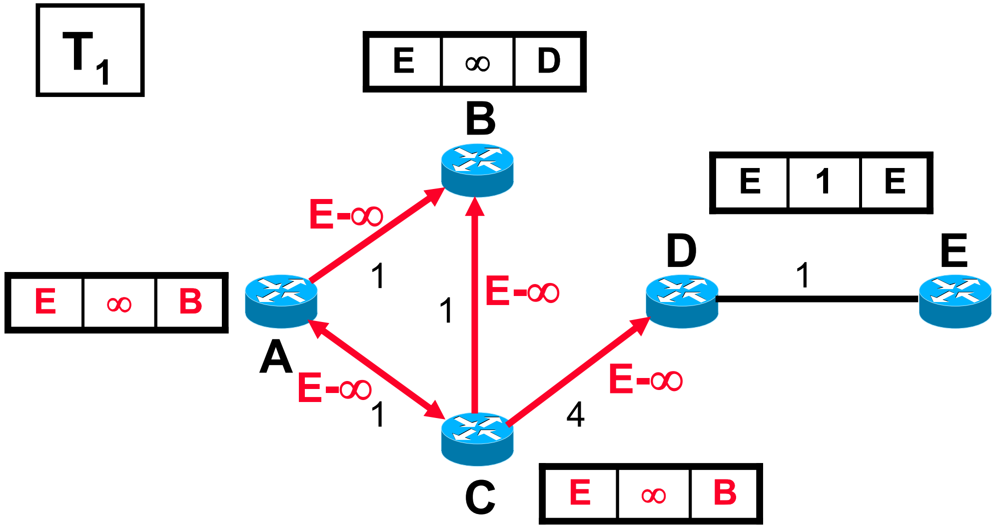
*only route towards E is considered

T_0

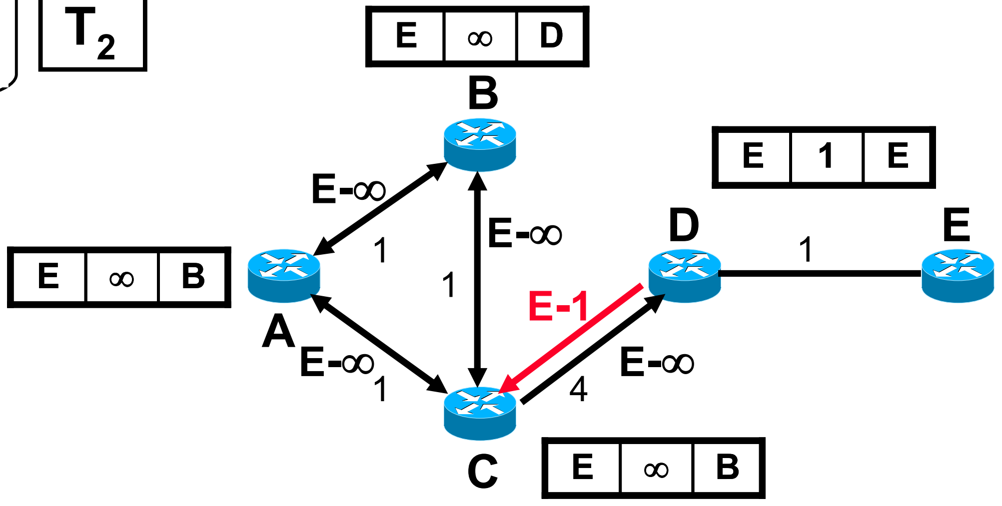
- set link cost = ∞
- update routing table
- send new DV



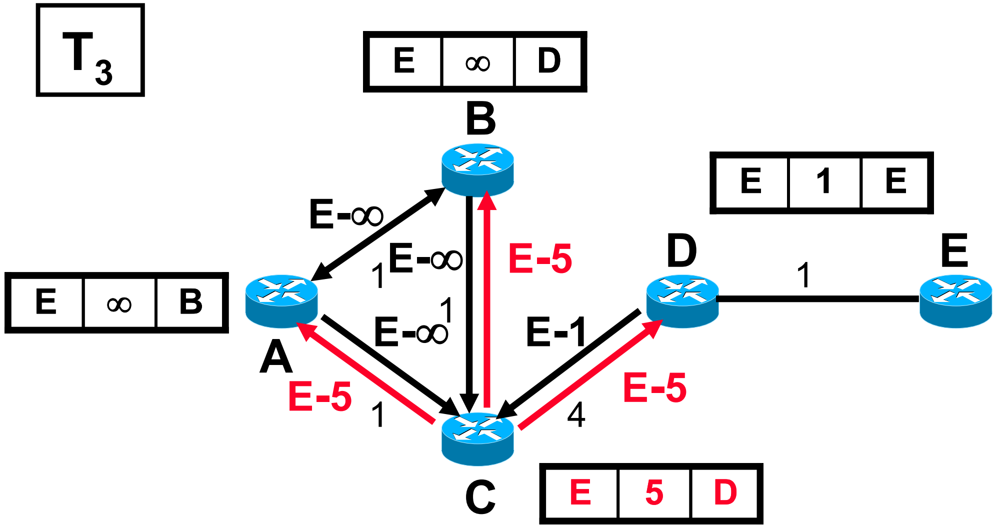
T_1



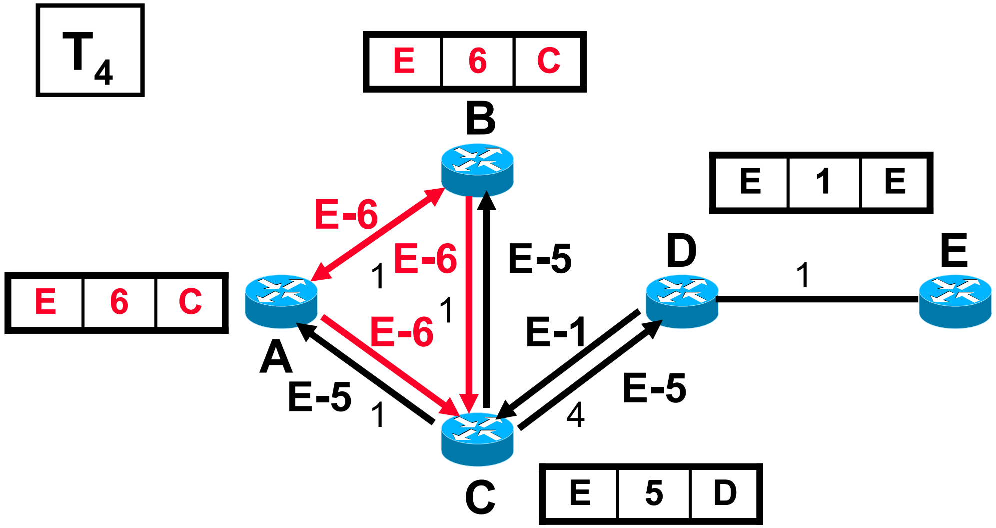
T_2



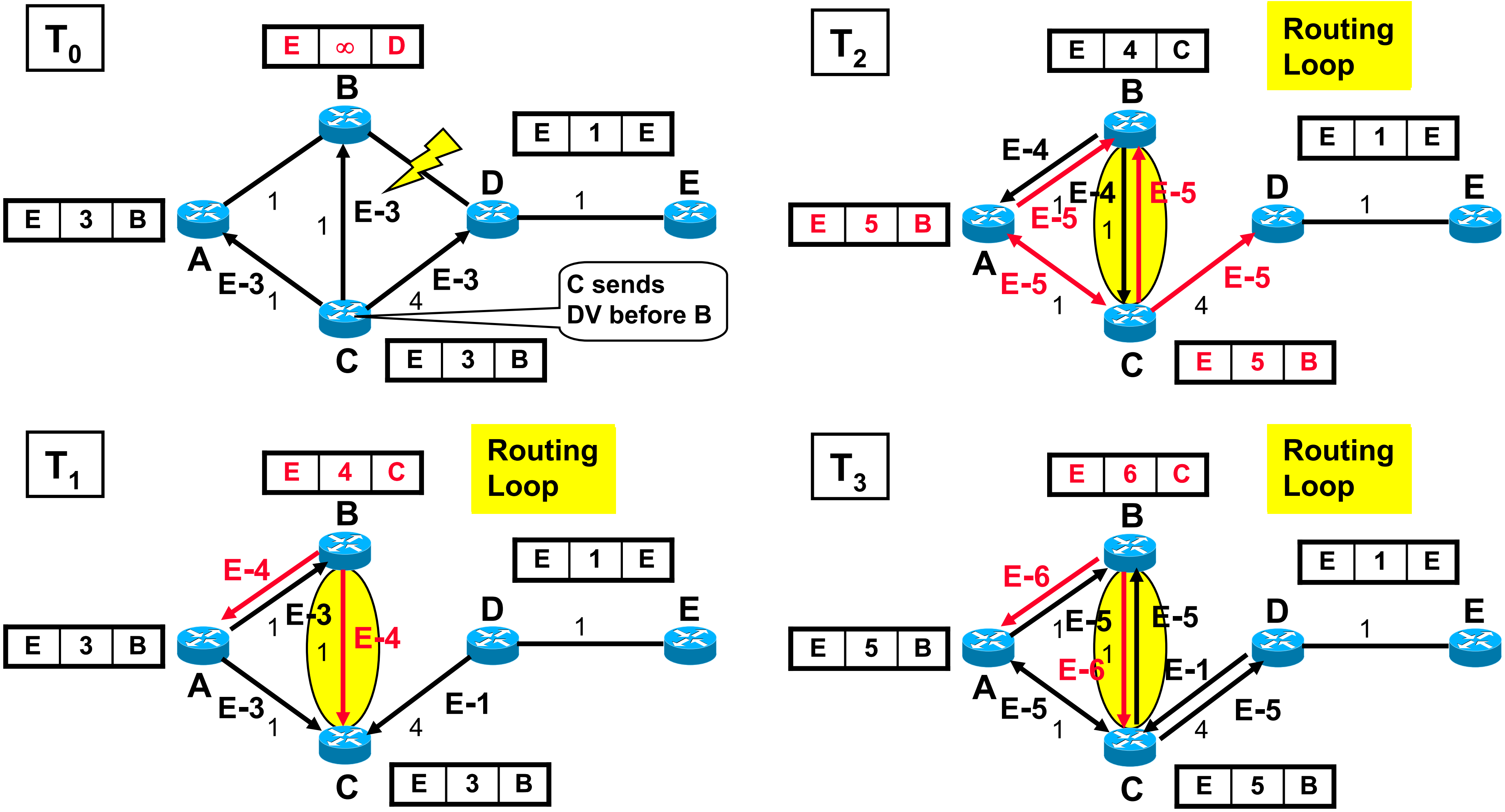
T_3



Example: Behaviour after Change of Network Topology (2)

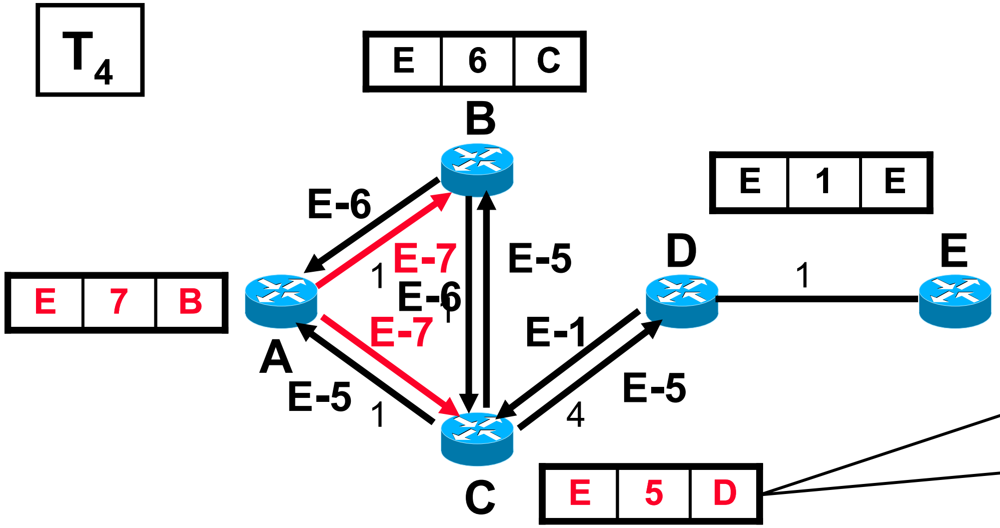


Typical Problems of DV Routing: Bouncing (1)



Typical Problems of DV Routing: Bouncing (2)

T_4

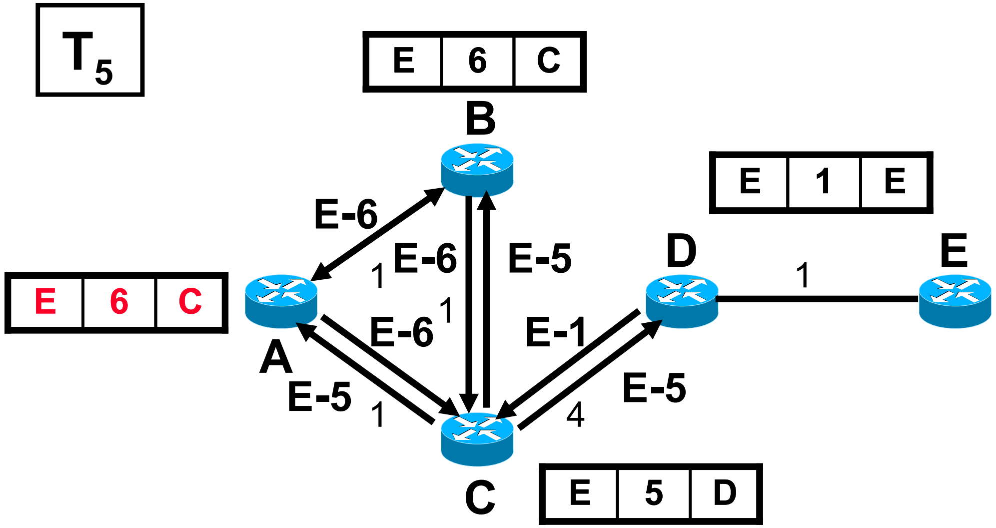


Assumption

Order of message reception at C:

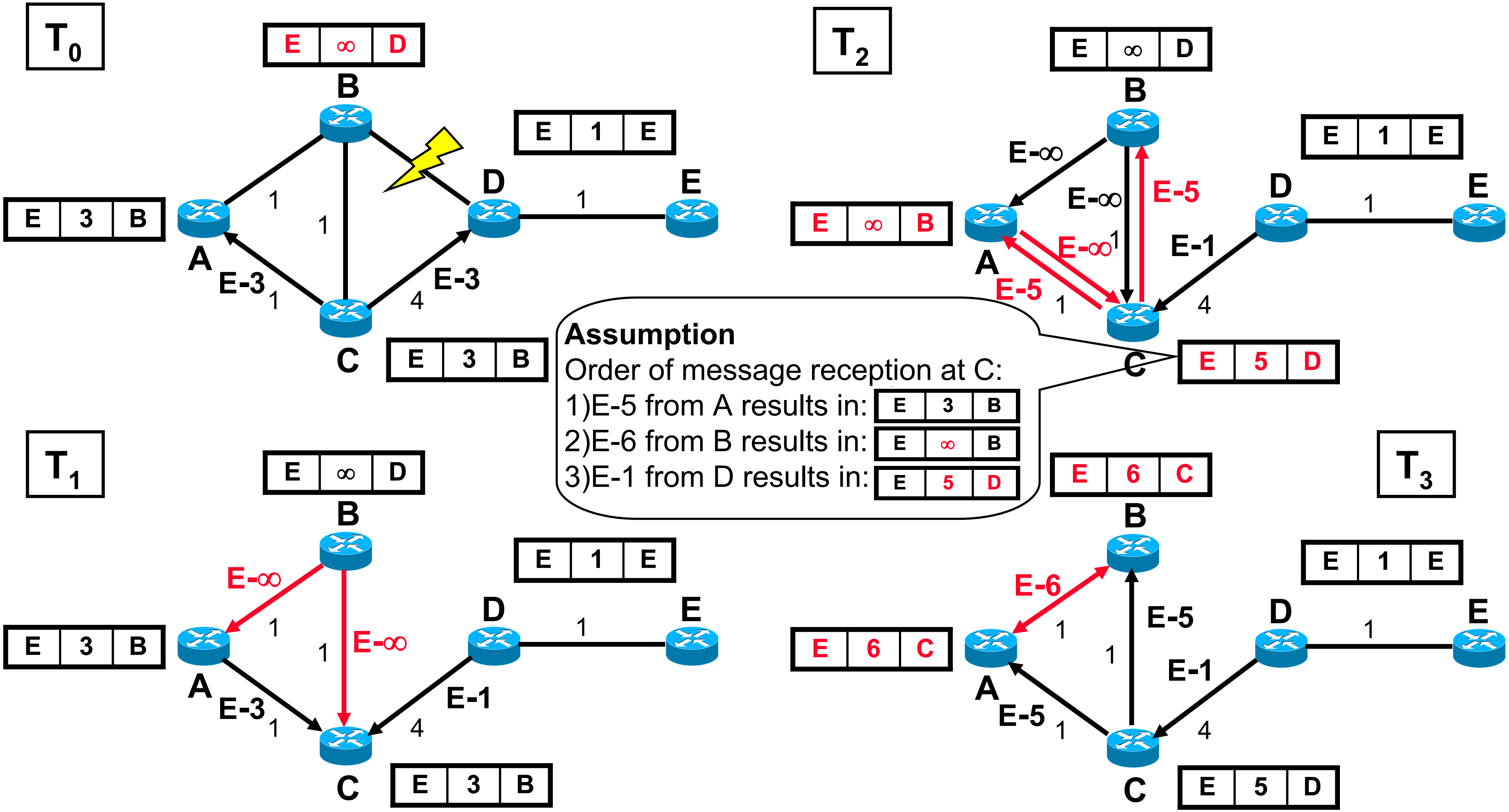
1) E-5 from A results in:	E	5	B
2) E-6 from B results in:	E	7	B
3) E-1 from D results in:	E	5	D

T_5



Prevention of Bouncing: Split Horizon*

*split horizon = no sending of distance info. to neighbours from which the route was learned



Assumption

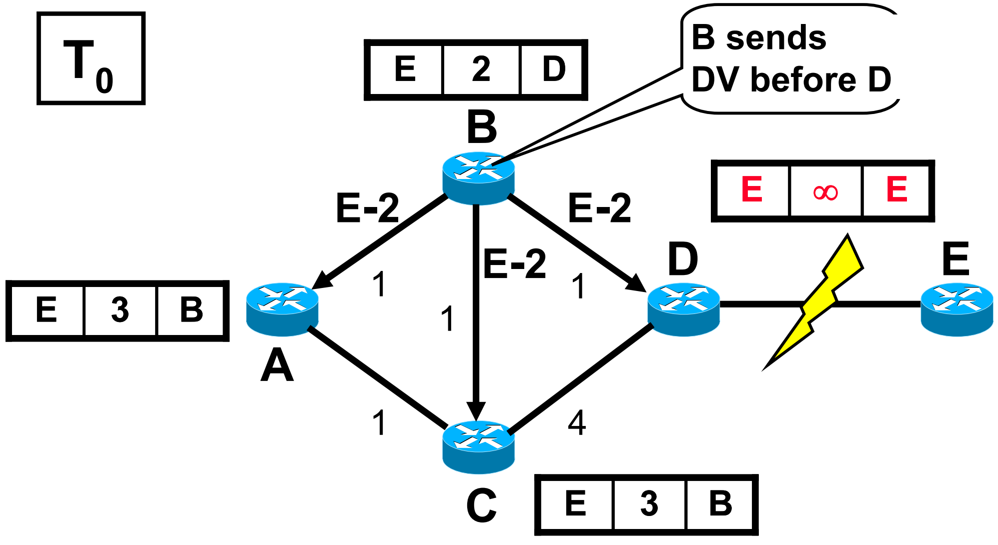
Order of message reception at C:

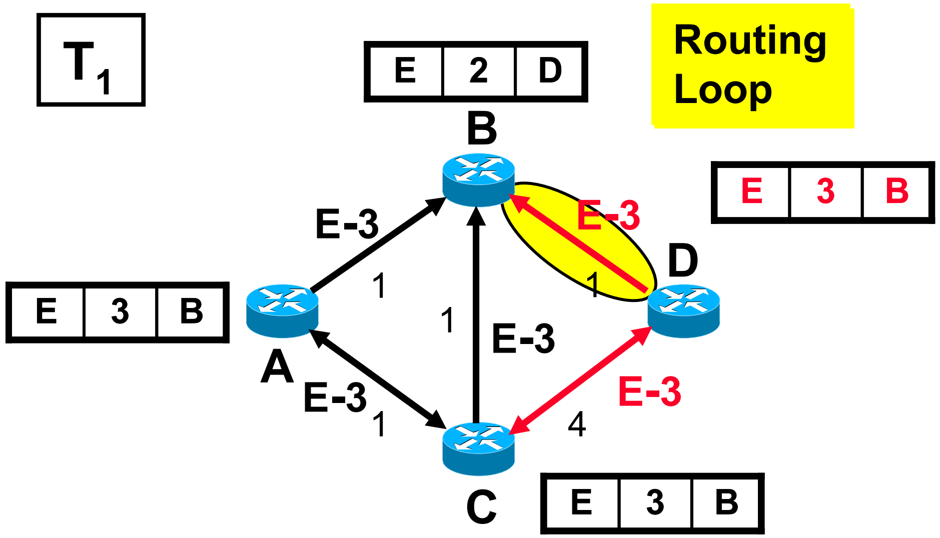
1)E-5 from A results in: E 3 B

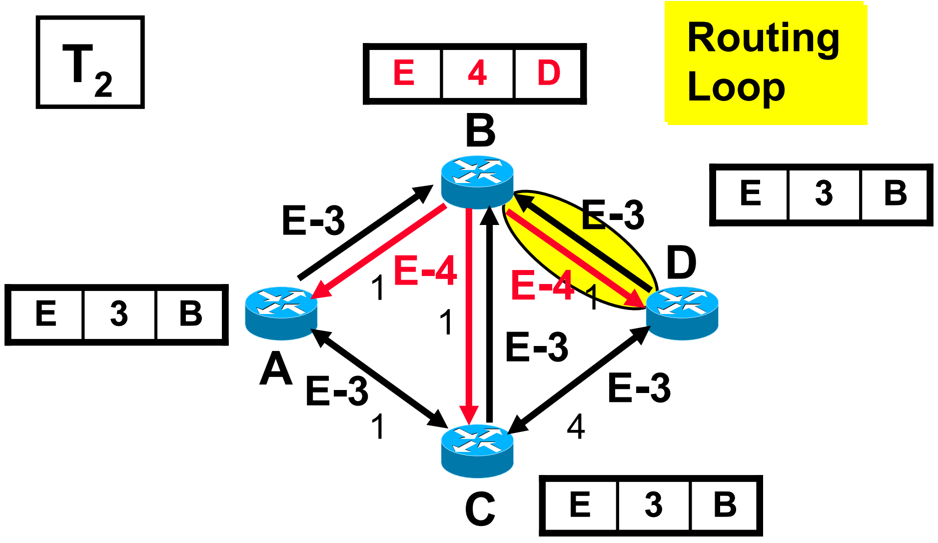
2)E-6 from B results in: E ∞ B

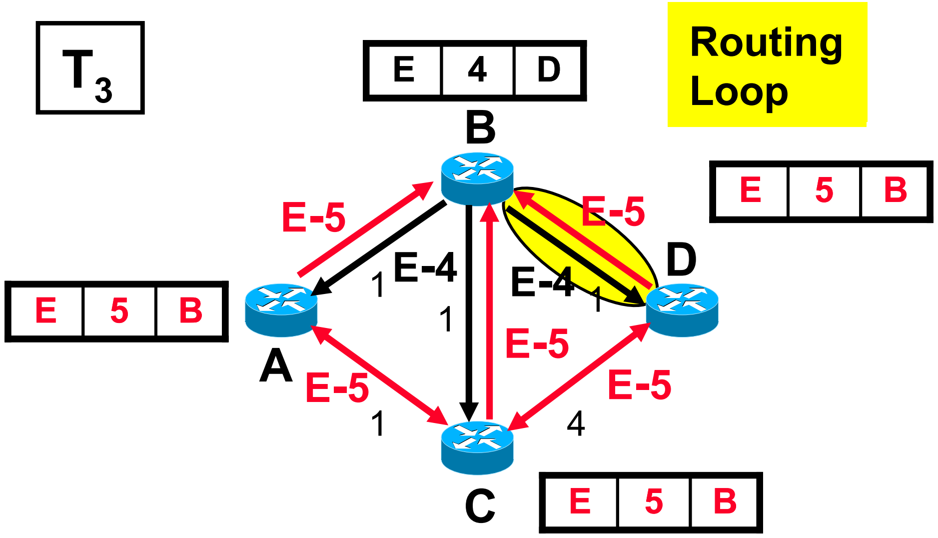
3)E-1 from D results in: E 5 D

Typical Problems of DV Routing: Count to Infinity



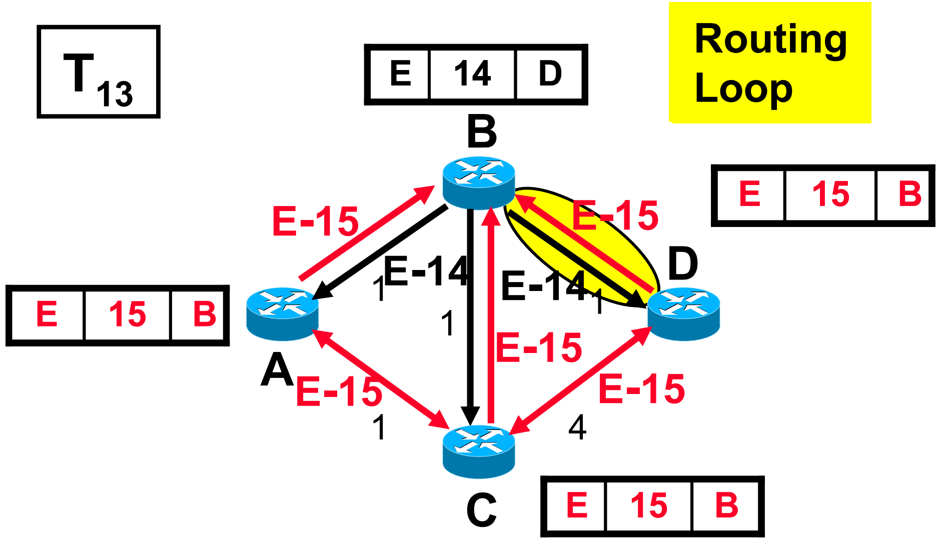


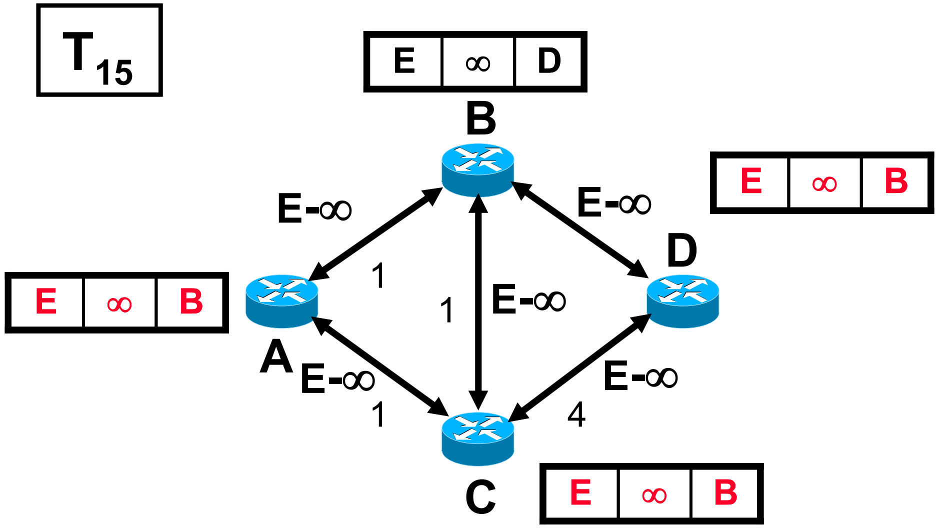


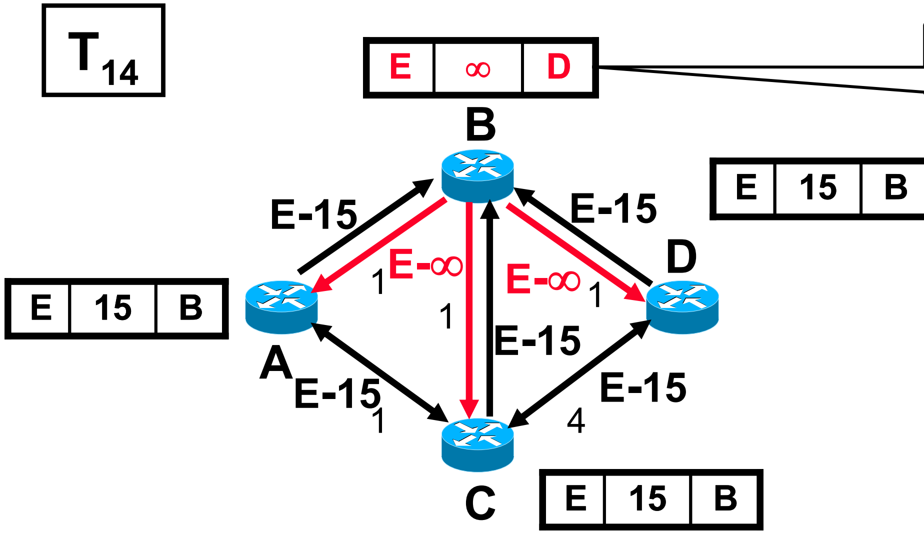


Prevention of Count to Infinity: Maximum Distance*

*here a max. distance of 15 is assumed







set distances larger than 15 to ∞

∞ = unreachable

Distance Vector Routing - Summary (1)

Properties of DV routing:

- the routers only know the distances to all other routers, but don’t have knowledge about the whole network topology
- the routers send their distance information to all neighbours
- the route calculation is performed step by step via the Bellman-Ford Algorithm (distributed route calculation)

Advantages of DV routing:

- simple implementation / simple algorithm

Disadvantages of DV routing:

- waste of bandwidth (always the complete DV is sent)
- bad performance (without modification):
  - slow convergence in large networks
  - occurrence of routing loops: bouncing / count to infinity problem
- upper limit of network size due to the introduction of a maximum distance (to avoid the count to infinity problem)

Distance Vector Routing - Summary (2)

Performance improvements for DV routing:

- **split horizon:** no sending of distance information to neighbours from which they were learned ⇒ avoid routing loops
- **split horizon + poison reverse:** sending of modified distance information (distance = ∞) to neighbours from which they were learned ⇒ accelerates the avoidance of routing loops
- **triggered updates:** sending of new DVs directly after a change in the routing table (instead of waiting for the next update cycle) ⇒ faster convergence
- **path hold down:** time interval which starts immediately after a router learned that a destination is unavailable; during this time interval all new distance information about this destination is ignored ⇒ thus it is ensured that the information about the unreachability of a destination is propagated through the whole network and no erroneous DV (of routers which are not informed yet) disturb this process
- **definition of a maximum distance:** ⇒ avoidance of count to infinity

DV-Protocol Example: RIP (Routing Information Protocol)

Properties of RIPv1 (RFC1058, RFC1388):

- cost metric = distance = hop count
- maximum hop count = 15 (16=infinity)
- route calculation with Bellman-Ford algorithm (distributed calculation)
- RIP DVs are sent periodically every 30 s and also after routing table changes or after an explicit request; lifetime of DVs: 180s
- RIP messages use UDP, Port 520
- no support of subnet masks with variable length (VLSM)
- no authentication of messages
- 2 possible operation modes: 1) active/passive mode: sending and reception of DVs; 2) silent mode: only reception of DVs possible
- 2 possible message types: 1) request: request to a neighbouring router to send its DV (or part of it); 2) response (advertisement): sending of DVs to neighbouring routers (periodically or after request)

Improvements in RIPv2 (RFC1723):

- support of VLSM
- authentication of messages
- use of IP multicast addresses for an efficient sending of DVs

Comparison of different DV Routing Protocols

Properties	RIPv1	RIPv2	IGRP
Count to Infinity	X	X	X
Split Horizon	X	X	X
Hold down Timer	X	X	X
Triggered Updates	X	X	X
VLSM Support		X	X
Routing Algorithm	Bellman-Ford	Bellman-Ford	Bellman-Ford
Metric	Hops	Hops	Compos. Metric
Hop Limit	15	15	100
Scalability	low	low	med.

Intra-domain Routing
Link State (LS) Principle

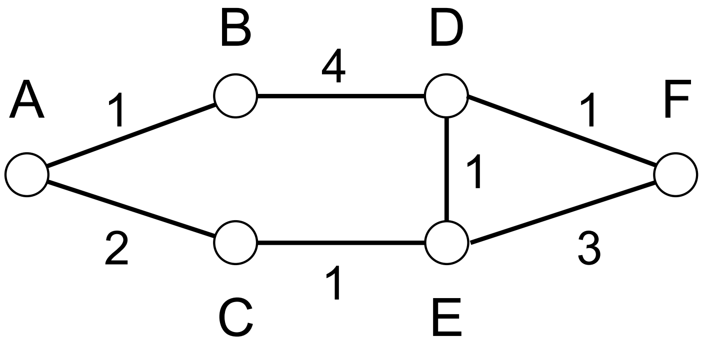
Link State Principle

- every router knows the complete network topology and link costs - this information is stored in the **Link State Data Base (LSDB)**
- based on this information every router calculates the best paths (i.e. the shortest paths in terms of link costs) to all destinations using the **shortest path algorithm** (e.g. the **Dijkstra algorithm**)
- the next hops (as seen from this router) of these shortest paths are then stored in the routing table
- changes in the network topology are distributed via **Link State Update (LSU)** messages* (flooding) in the whole network
- after reception of a link state update message a router updates its link state database, starts recalculating the shortest paths and updates the routing table if needed

***Note:** for reliable flooding of LSU messages a special mechanism is used to avoid an erroneous update of the link state databases

Link State Routing Example

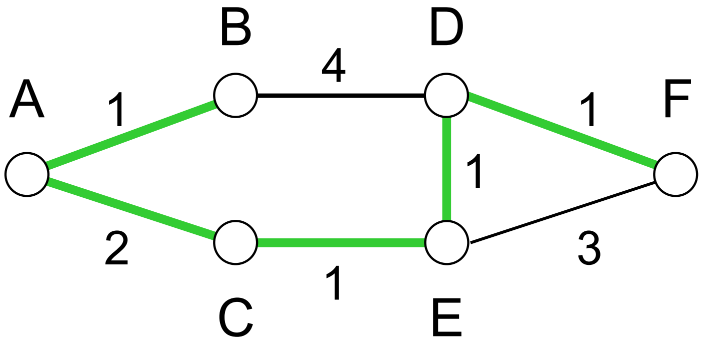
Network example:



link state data base at router A:

Link	Cost
AB	1
AC	2
BD	4
CE	1
DE	1
DF	1
EF	3

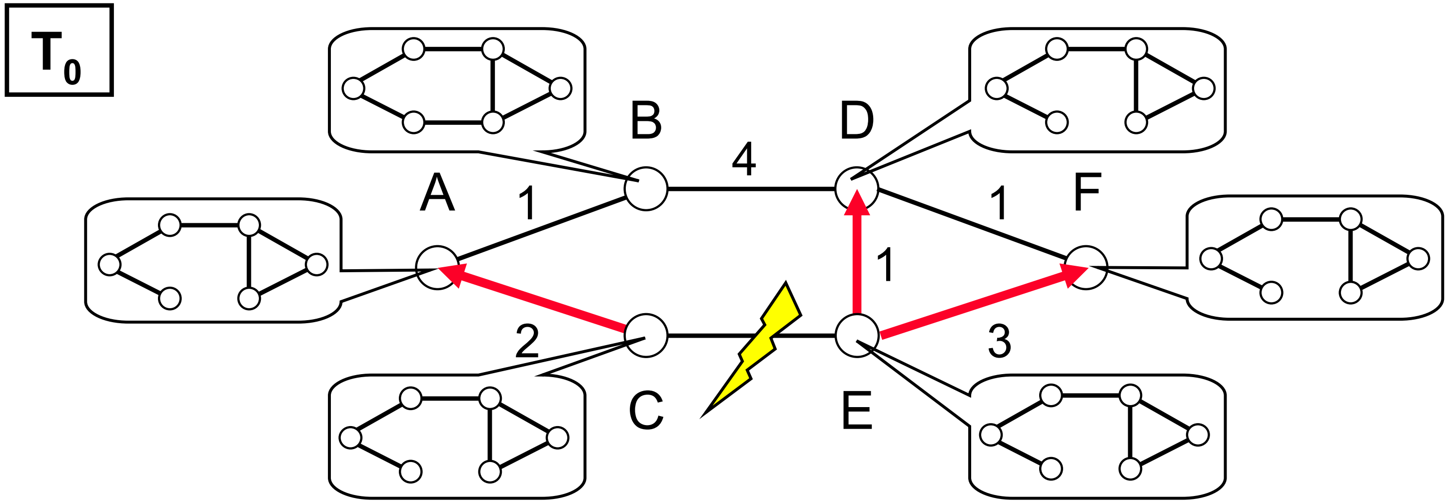
tree of the shortest paths for router A:

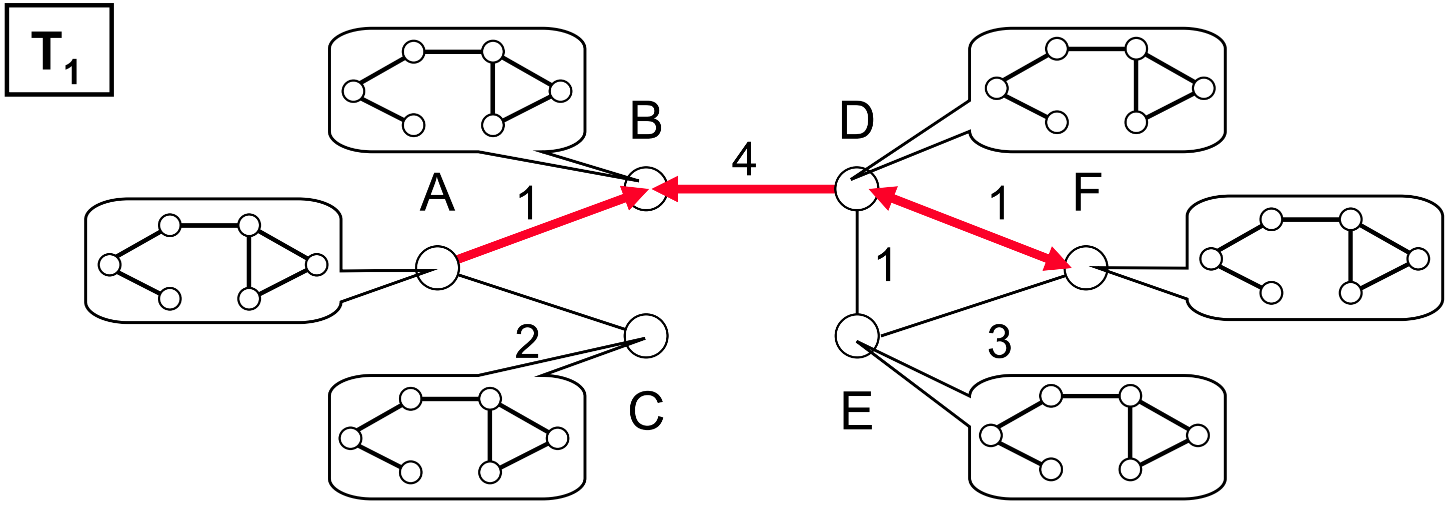


routing table at router A:

dest.	path cost	next hop
B	1	B
C	2	C
D	4	C
E	3	C
F	5	C

Example: Behaviour after Change of Network Topology





flooding of link state updates: →

Link C-E: Cost =∞

visualization of the link state data base

Shortest Path Calaculation with Dijkstra Algorithm

Dijkstra Algorithm: find the shortest paths starting at a source node to all destinations (**tree of shortest paths**)

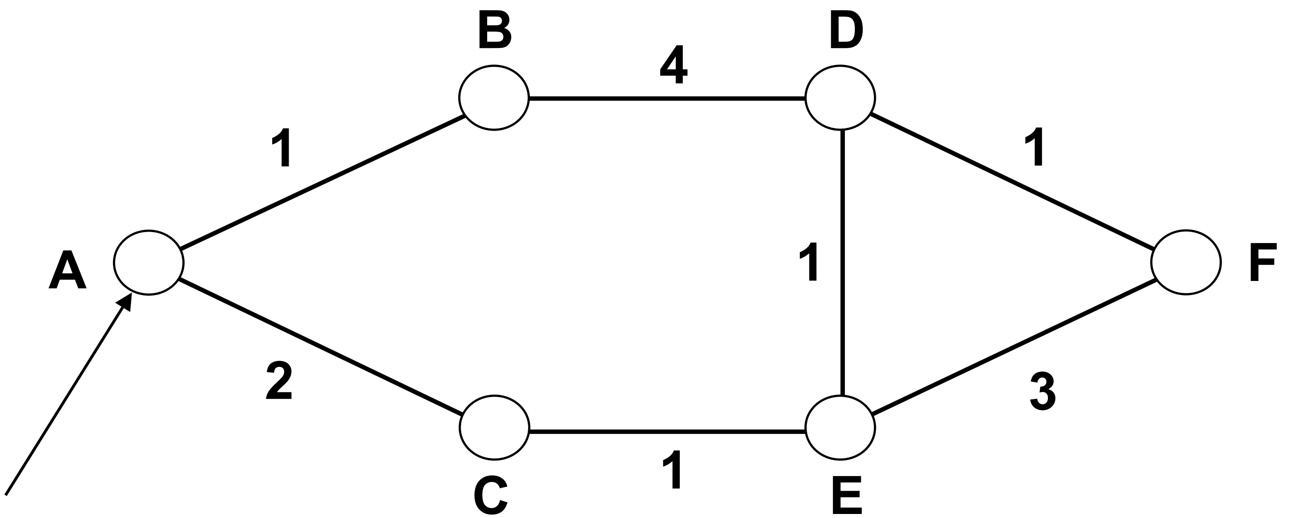
Initialization:

- define the set of nodes which are already considered (e.g. the nodes for which a shortest path is known) - at the beginning, this set only consists of the source node
- identify all reachable (e.g. all directly connected) nodes
- determine the path cost between these nodes and the source node

Repeat until all nodes are considered:

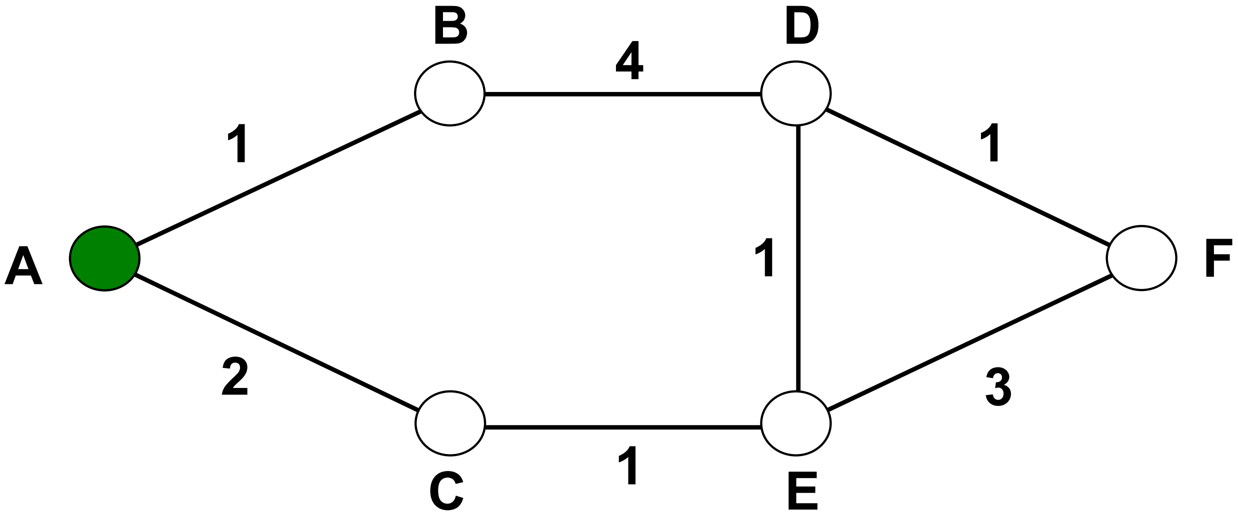
- chose the node with the lowest path cost, which is not yet in the set of considered nodes and add it to this set
- determine the shortest paths to all reachable nodes
- update the path cost between the nodes and the source node

Example: Calculation of the Shortest Paths Tree (Dijkstra)



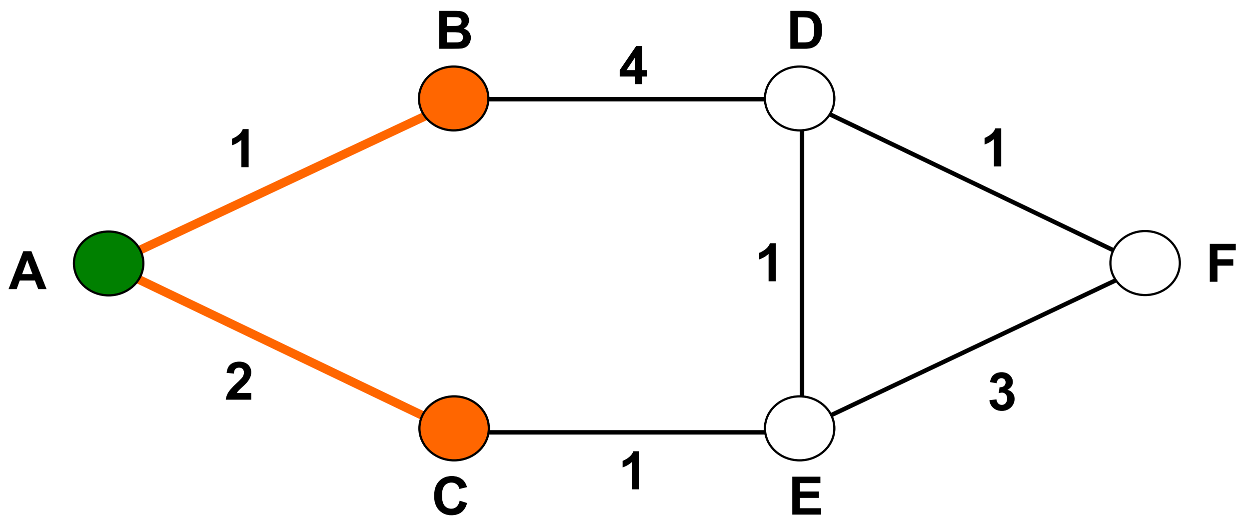
wanted: all shortest
paths between
source node A and
all other nodes

Example: Calculation of the Shortest Paths Tree (Dijkstra)



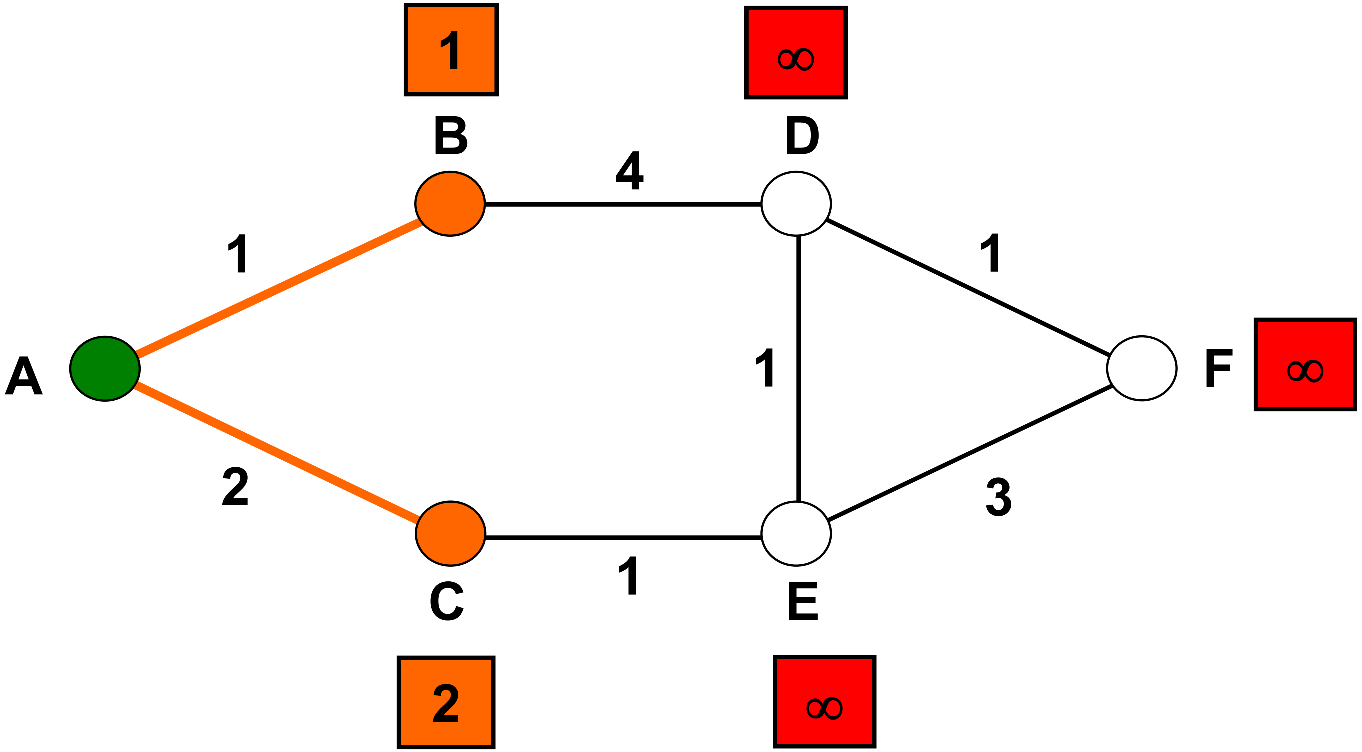
- start with source node (= starting set of considered nodes)

Example: Calculation of the Shortest Paths Tree (Dijkstra)



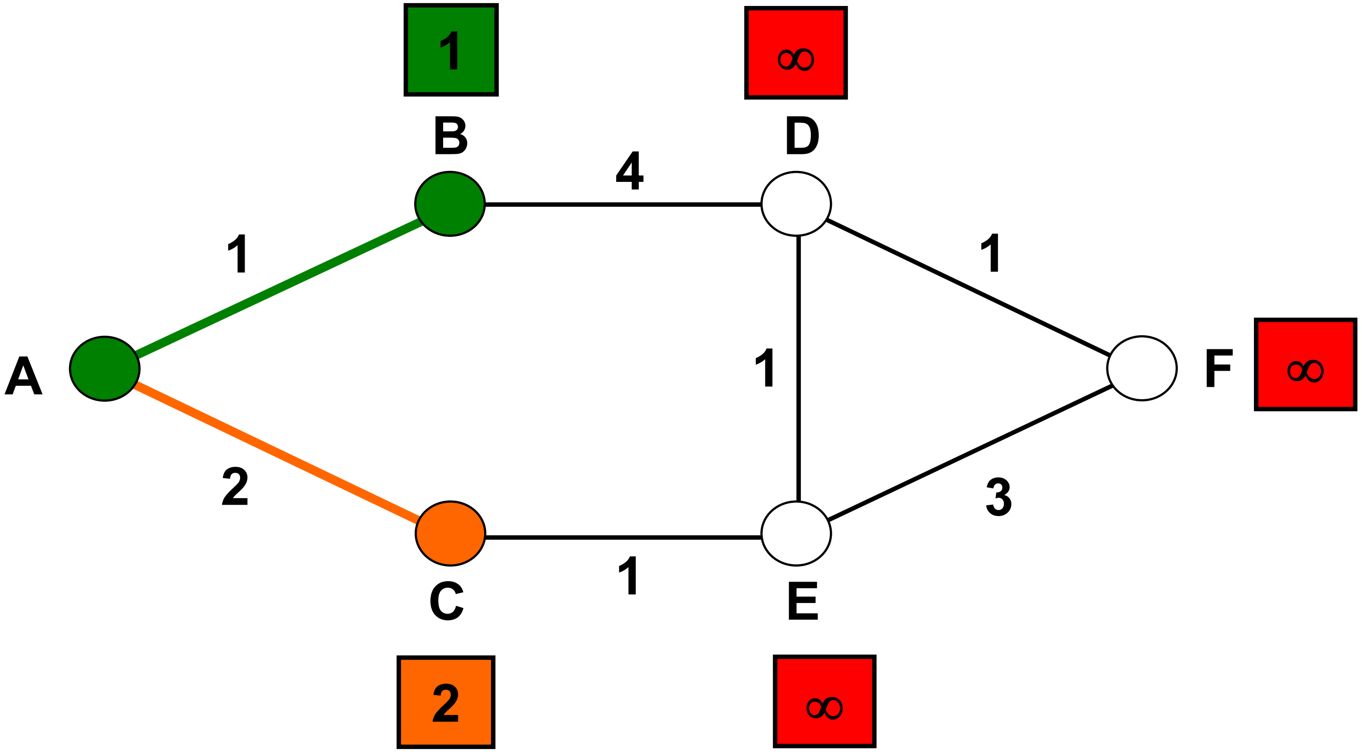
- start with source node
- identify all reachable nodes

Example: Calculation of the Shortest Paths Tree (Dijkstra)



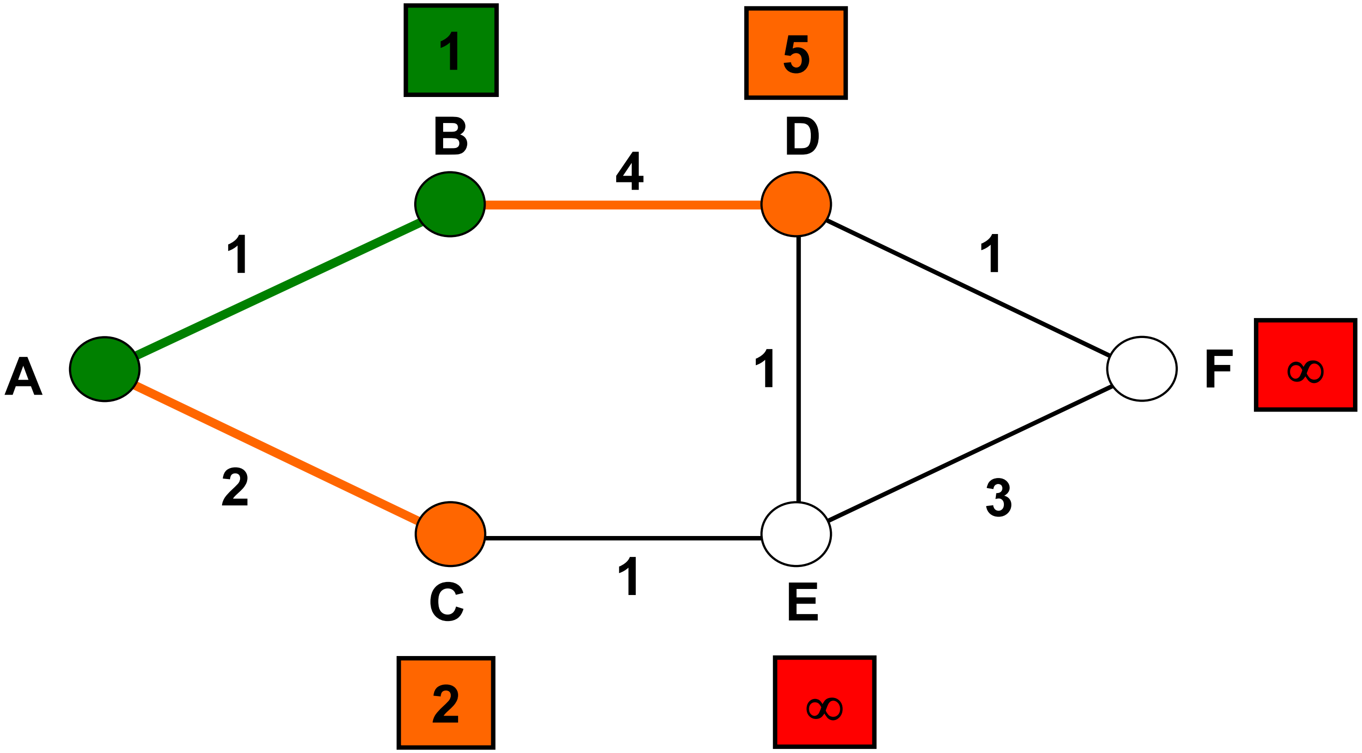
- start with source node
- identify all reachable nodes
- determine the path costs based on the source node

Example: Calculation of the Shortest Paths Tree (Dijkstra)



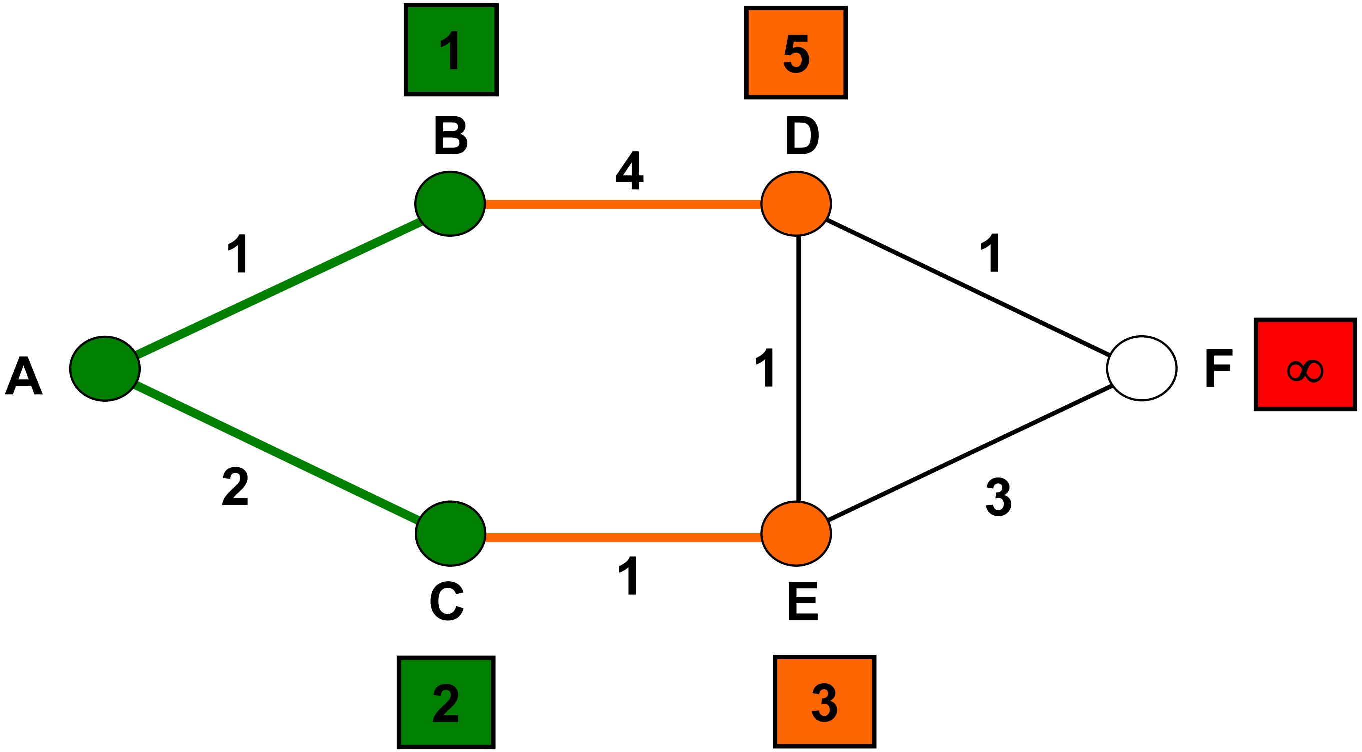
- choose the node with the lowest path cost, which is not yet in the set of considered nodes and add it to this set

Example: Calculation of the Shortest Paths Tree (Dijkstra)



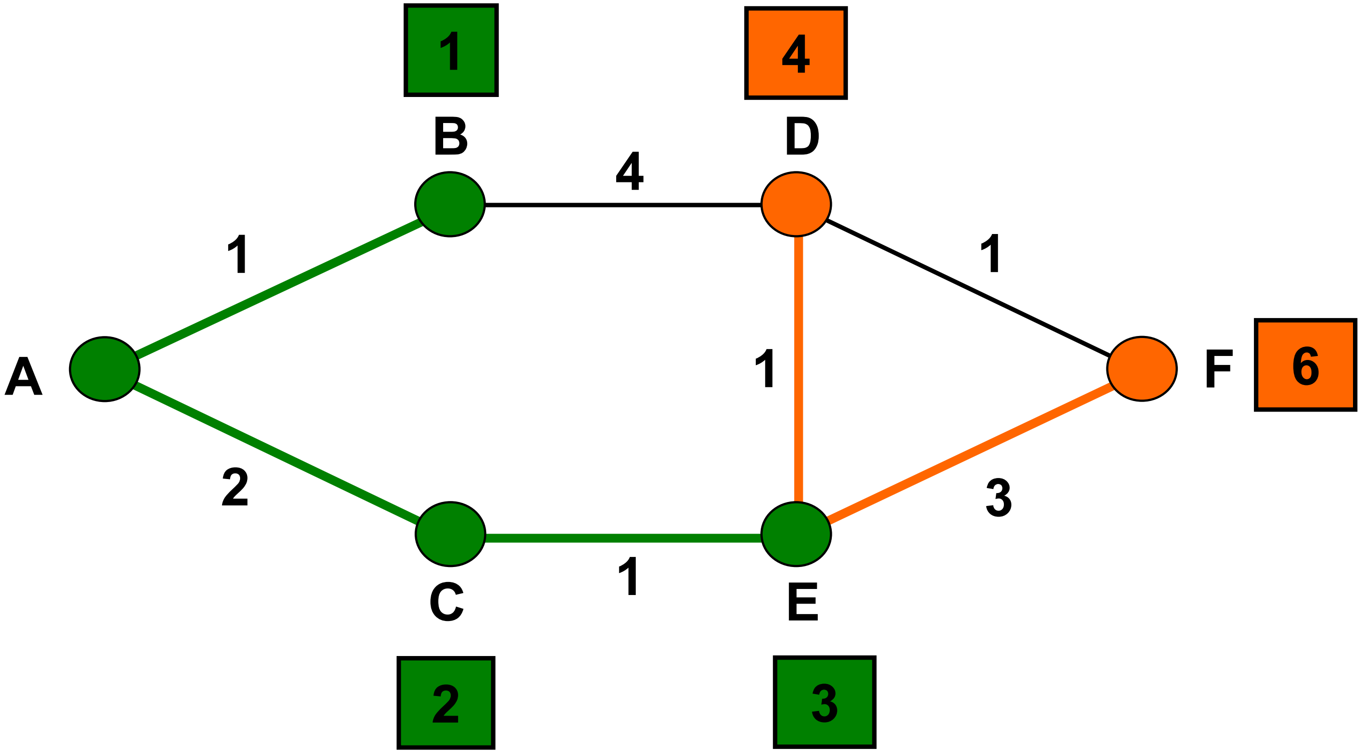
- choose the node with the lowest path cost, which is not yet in the set of considered nodes and add it to this set
- determine the shortest paths to all reachable nodes
- update the path costs based on the source node

Example: Calculation of the Shortest Paths Tree (Dijkstra)



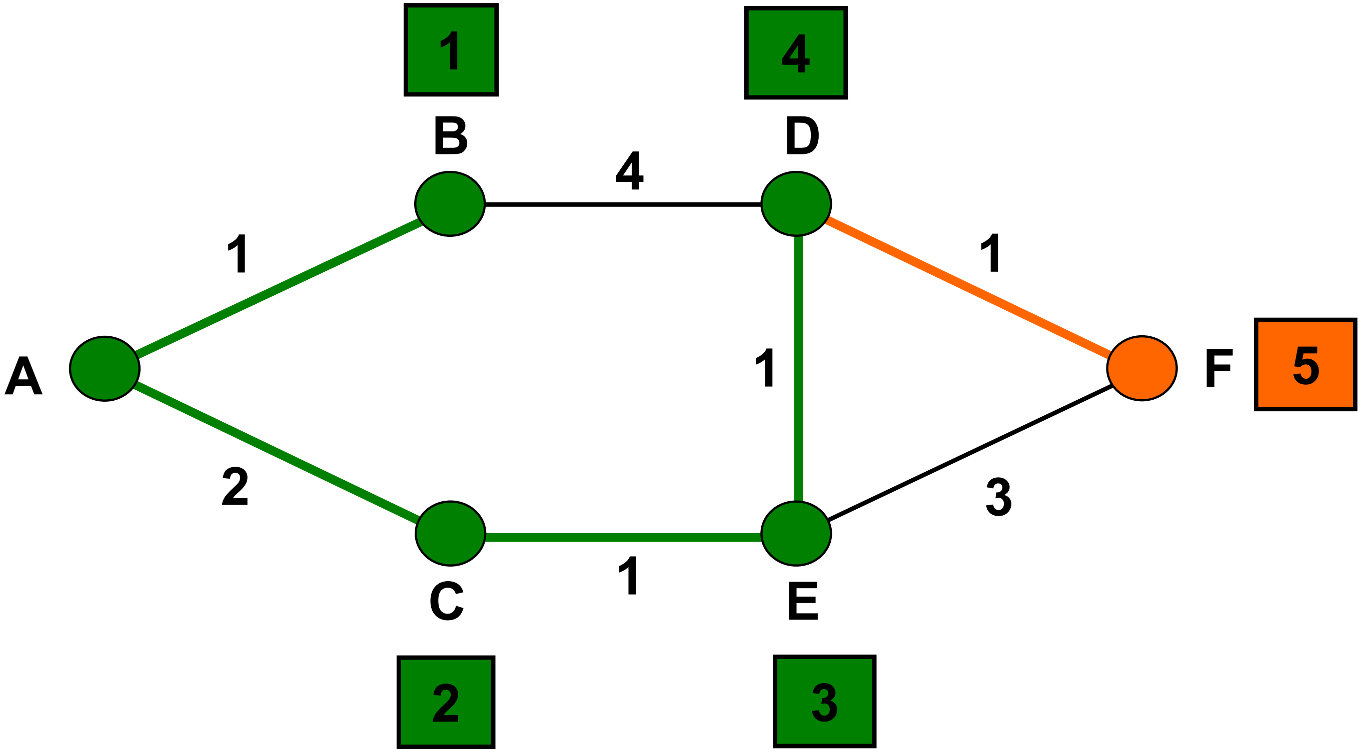
- choose the node with the lowest path cost, which is not yet in the set of considered nodes and add it to this set
- determine the shortest paths to all reachable nodes
- update the path costs based on the source node

Example: Calculation of the Shortest Paths Tree (Dijkstra)



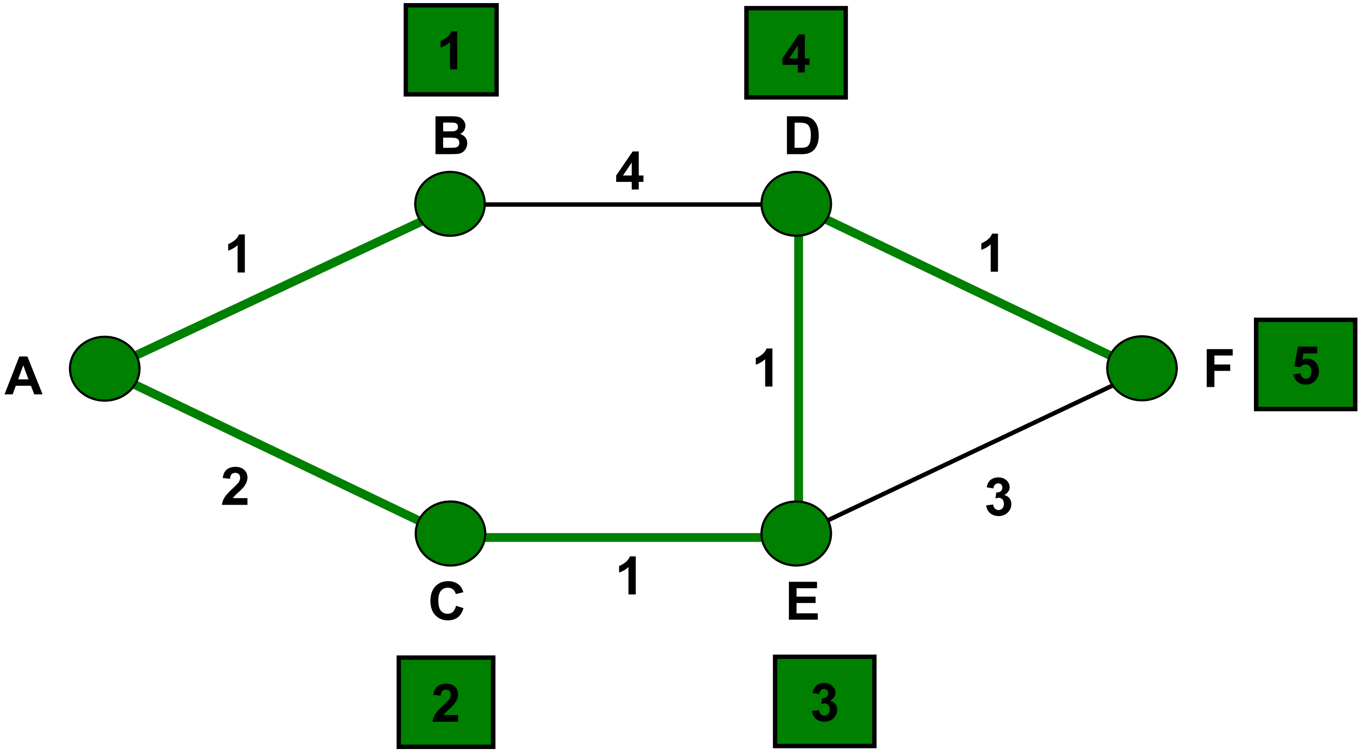
- choose the node with the lowest path cost, which is not yet in the set of considered nodes and add it to this set
- determine the shortest paths to all reachable nodes
- update the path costs based on the source node

Example: Calculation of the Shortest Paths Tree (Dijkstra)



- choose the node with the lowest path cost, which is not yet in the set of considered nodes and add it to this set
- determine the shortest paths to all reachable nodes
- update the path costs based on the source node

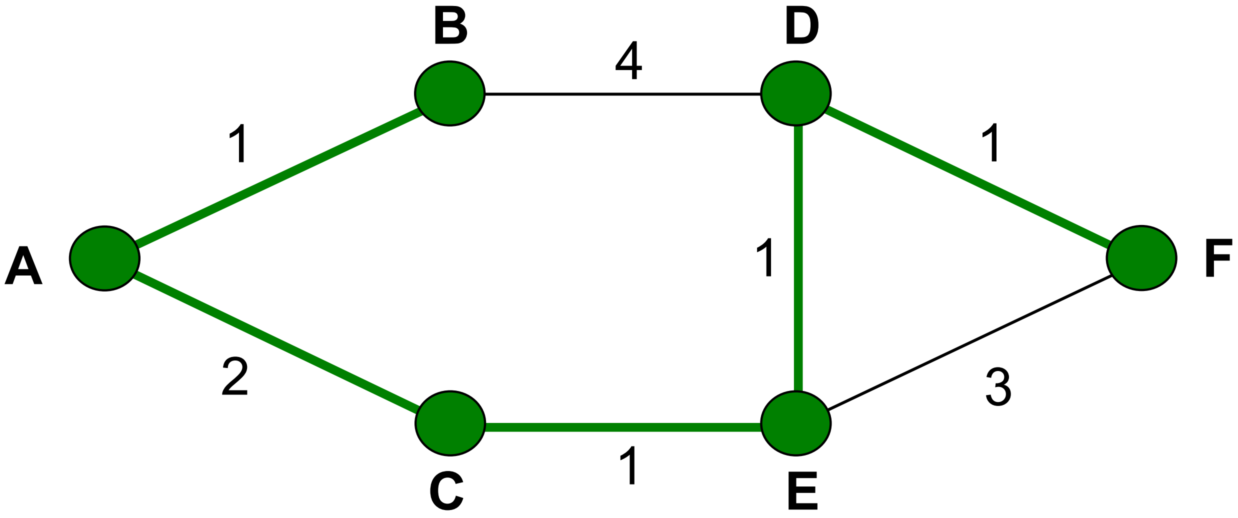
Example: Calculation of the Shortest Paths Tree (Dijkstra)



- if all nodes are considered → stop

Example: Calculation of the Shortest Paths Tree (Dijkstra)

Shortest Paths Tree for source node A



⇒ **routing table for node A:**

Dest.	path cost	next hop
B	1	B
C	2	C
D	4	C
E	3	C
F	5	C

Link State Routing - Summary

Properties of LS routing:

- routers have knowledge of the complete network topology
- changes in the network topology (link states) are flooded into the whole network
- the route calculation is performed autonomously by every router (i.e. without the help of other routers) using a shortest path algorithm (e.g. the Dijkstra algorithm)

Advantages of LS routing:

- fast convergence of routing tables without routing loops
- bandwidth savings (only link state changes are sent)
- support of multiple link cost metrics
- equal cost multi path routing (with load balancing) possible

Disadvantages of LS routing:

- the calculation of shortest paths requires much processing power (CPU)
- complex implementation

LS Protocol Example: OSPF (Open Shortest Path First)

Properties of OSPF (RFC1247, RFC2328):

- link checking and neighbour router detection (Hello)
- initial exchange of complete database content between neighbouring routers (Exchange)
- route calculation with Dijkstra's shortest path algorithm
- predefined link cost metric (Cisco): $\frac{10^8 \text{ (bit/s)}}{\text{link bandwidth (bit/s)}}$ (value range: [1, 65535])
- flooding of link state updates: periodically and after topology changes (triggered updates) (Flooding)
- authentication of OSPF messages
- support of equal cost multi path routing
- for large networks: hierarchical OSPF (area concept)

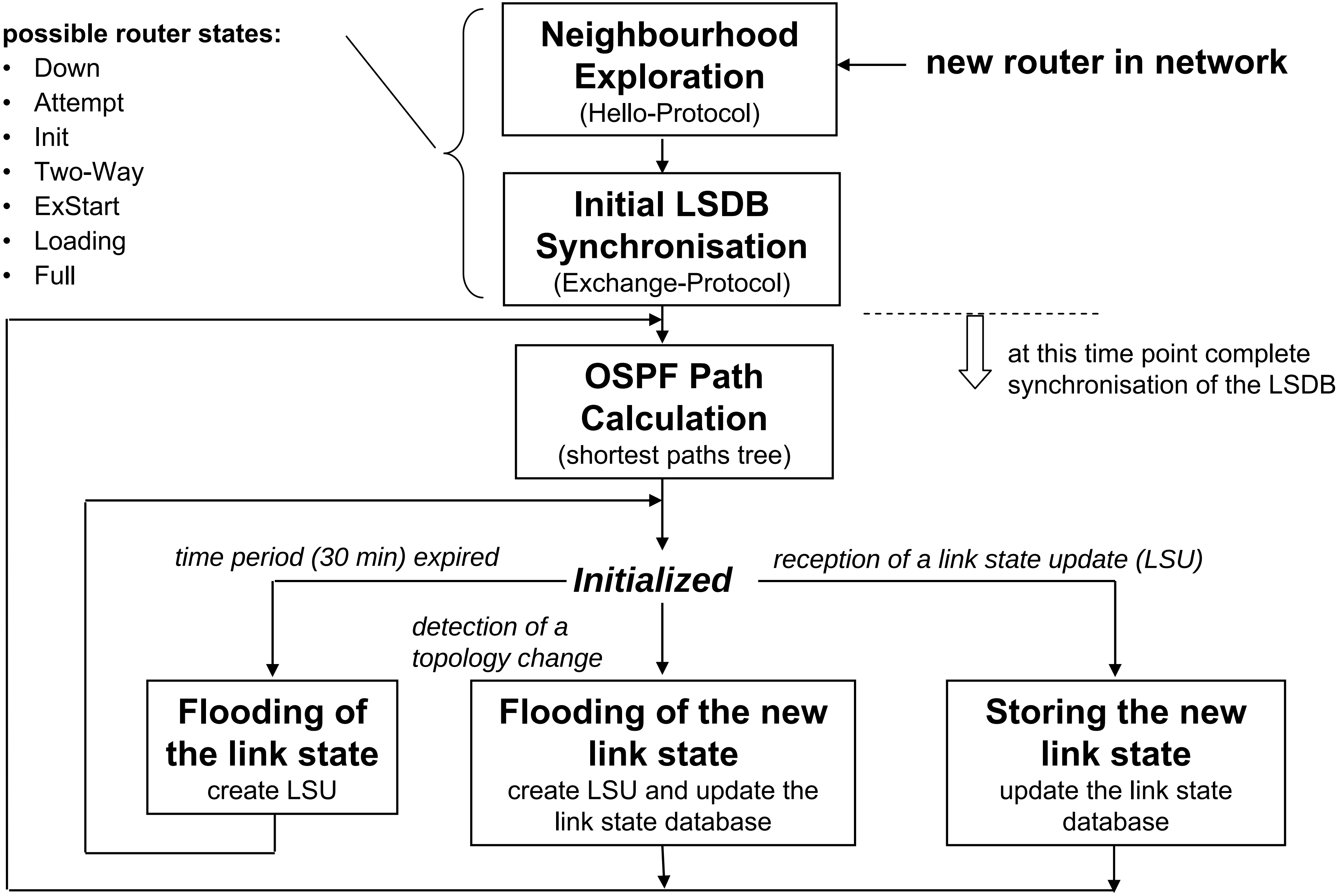
Protocols within OSPF:

- **Hello Protocol**: link checking, neighbour router detection
- **Exchange Protocol**: initial synchronization of link state databases between neighbouring routers, i.e. exchange of the complete link state databases
- **Flooding Protocol**: sending of link state update messages (every 30 min)

OSPF Mode of Operation - Overview

possible router states:

- Down
- Attempt
- Init
- Two-Way
- ExStart
- Loading
- Full



Neighbourhood Exploration
(Hello-Protocol)

new router in network

Initial LSDB Synchronisation
(Exchange-Protocol)

at this time point complete synchronisation of the LSDB

OSPF Path Calculation
(shortest paths tree)

Initialized

time period (30 min) expired

detection of a topology change

reception of a link state update (LSU)

Flooding of the link state
create LSU

Flooding of the new link state
create LSU and update the link state database

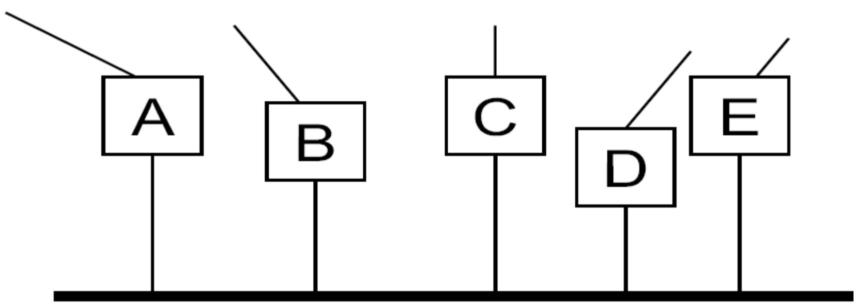
Storing the new link state
update the link state database

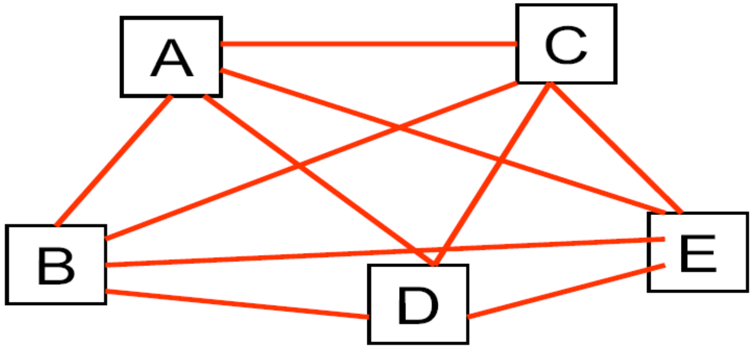
OSPF Mode of Operation - Neighbourhood Exploration

- Neighbours (according to OSPF) are routers, with which a router exchanges information directly; **Neighbourhoods = Adjacencies**
- **Hello Protocol:**
  - periodic transmission of **OSPF Hello messages** (default: every 10s)
  - a Hello message contains the router ID, information about router configuration and a list of all known neighbouring routers
  - usually Hello messages use as a destination IP addresses known multicast IP addresses (AllSPFRouters: 224.0.0.5 or AllDRouters: 224.0.0.6) (exception: NBMA configuration)
  - if a Hello message of a neighbouring router is not received within a certain time interval (default: 40s), this router is considered as down
- Purpose of the Hello Protocol:
  - ensuring the bidirectional communication between neighbouring routers
  - agreement of parameters between neighbouring routers

OSPF Mode of Operation - Neighbourhood Exploration

- **Problem:** inefficient communication between neighbouring routers in multipoint configuration (LAN) because of pairwise neighbourships

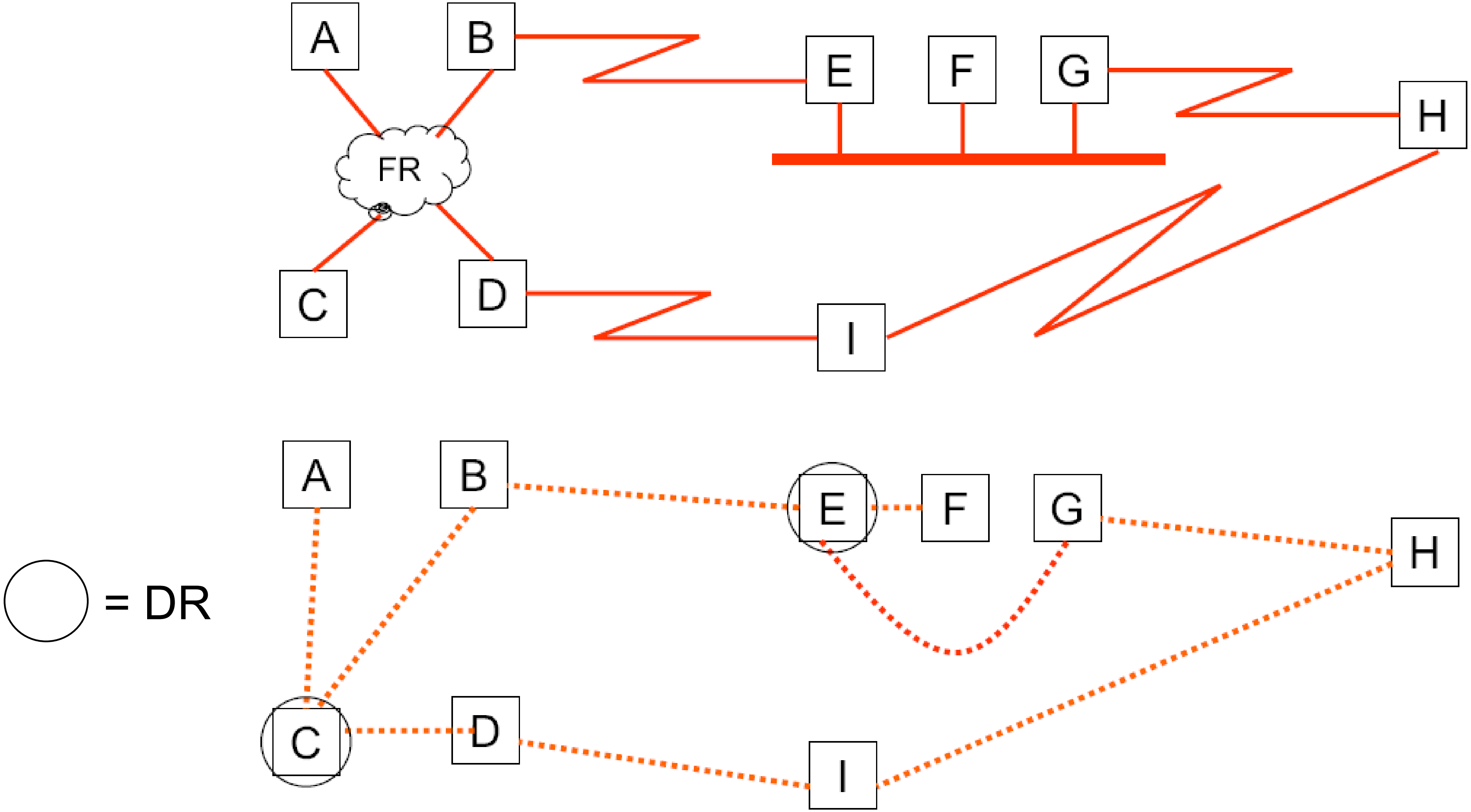




- **Solution:** one router in the LAN becomes **Designated Router (DR)**; all routers in the LAN maintain only neighbourships with the DR (and not between each other anymore); for redundancy reasons also a **Backup Designated Router (BDR)** is chosen; the negotiation about the DR and BDR is handled via the Hello Protocol

OSPF Mode of Operation - Neighbourhood Exploration

- **Example:** neighbourships in case of multipoint configuration

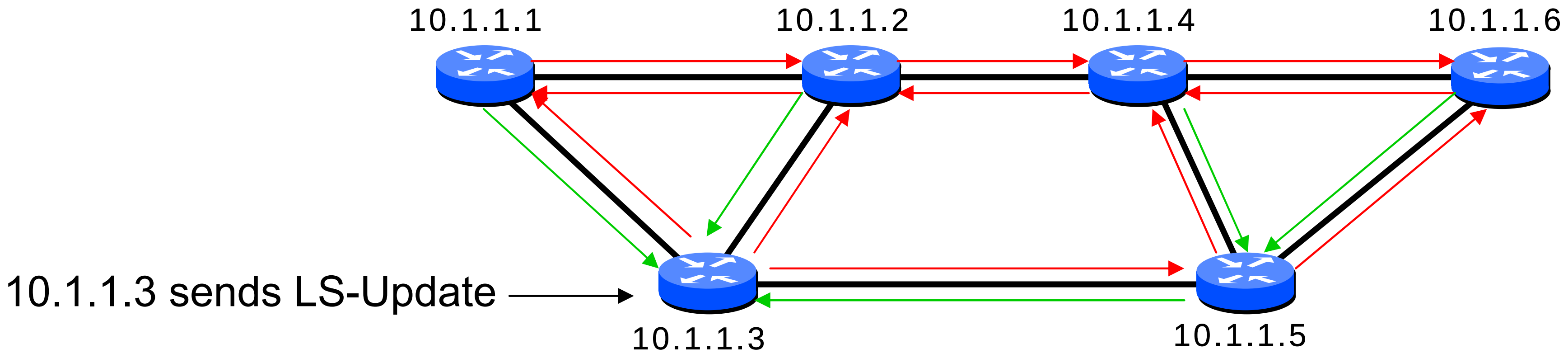


OSPF Mode of Operation - LSDB Synchronisation

- Initial LSDB synchronisation (**Exchange Protocol**)
    - Step 1: exchange of the LSDB contents between neighbouring routers after a neighbourship is established (via **OSPF Database Description messages**)
    - Step 2: a neighbouring router requests missing LSDB elements (LSAs) via **OSPF LS-Request** or **LS-Update messages;** the correct reception of a LS-Update message is acknowledged with a **OSPF LS-Ack** message
- Continuous LSDB synchronisation (**Flooding Protocol**)
    - LSDB synchronisation takes place periodically (default value: 30min) and after a topology change
    - for that, the respective LSAs are propagated in the whole network via **OSPF LS-Update messages**; this is a **reliable flooding** method, since the LS-Update messages are always acknowledged between neighbouring routers via OSPF LS-ACK messages
    - for the flooding of LSAs special timer values have to be considered (for example LSAs are updated at most every 5s)

OSPF Mode of Operation - Reliable Flooding

- **Example:** reliable flooding of LS-Update messages



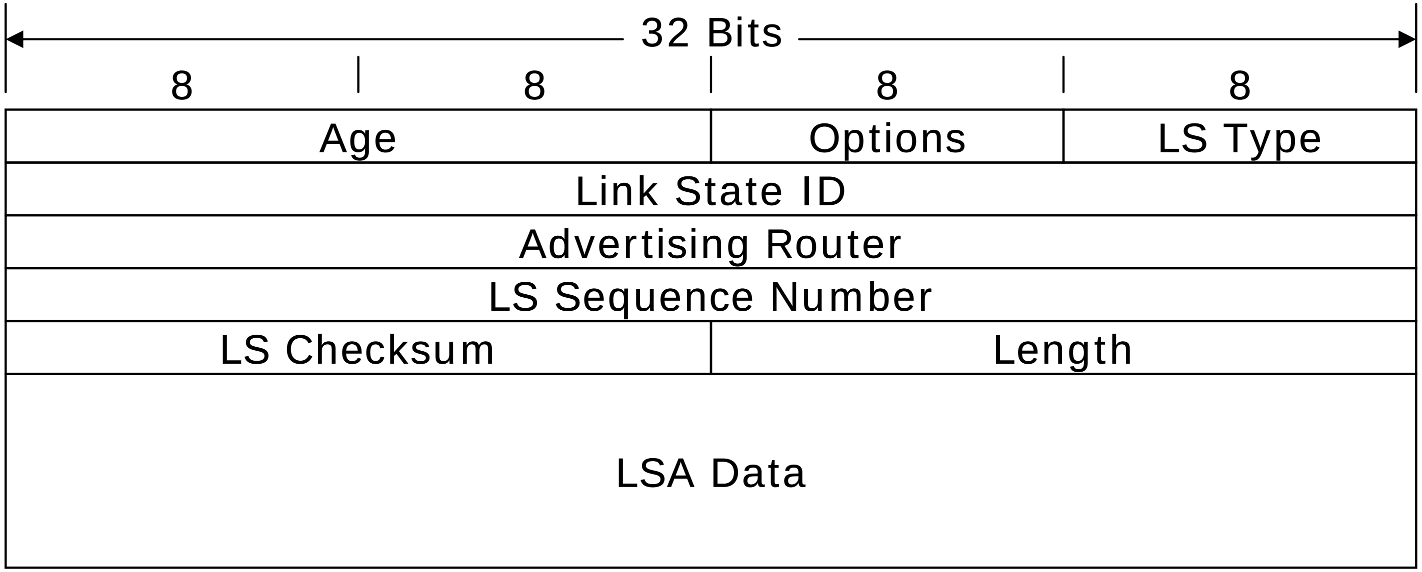
- Reliable flooding:
  - flooding over all links (not only over the spanning tree!)
  - all LSAs have to be acknowledged implicitly or explicitly:
    - implicit Ack: through reception of an identical copy of the LSA
    - explicit Ack: via OSPF LS-Ack (normally delayed)
  - transmission errors are detected via the LS checksum: affected LSAs are not acknowledged
  - further functions: LSA refresh (after 30min), LSA update at most every 5s, refusing of LSA which were accepted less than 1s ago

OSPF Link State Data Base (LSDB) and LSAs

- The **Link State Data Base (LSDB)** represents the description of the network (topology, link states and link costs) → base for calculating the routing table
- The LSDB contains as entries the so called **Link State Advertisements (LSAs)**; LSAs are exchanged via OSPF LS-Update messages between neighbouring routers; LSAs can be uniquely identified via the triple *LS type*, *link state ID* and *advertising router*
- Depending of the type of connection between neighbouring routers different LSA formats are used:
  - Router LSA: in case of point-to-point connections
  - Network LSA: in case of multipoint connections (LAN)
  - Network Summary LSA: in case of connections between OSPF Areas
  - AS external LSA: in case of external connections (to other ASes)

OSPF Router LSAs

- In case of point-to-point connections a router sends Router LSAs (on each connected link) to all its neighbouring routers (for example as part of OSPF LS-Update messages); by that, the router announces the part of the network topology, which is visible for him
- Router LSA format:

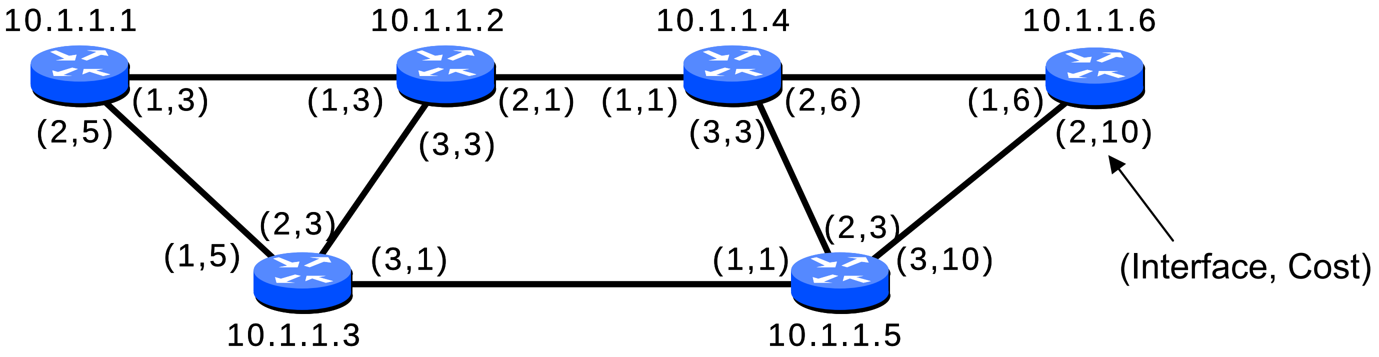


32 Bits

8	8	8	8
Age		Options	LS Type
Link State ID			
Advertising Router			
LS Sequence Number			
LS Checksum		Length	
LSA Data			

OSPF Router LSAs

- **Example:**

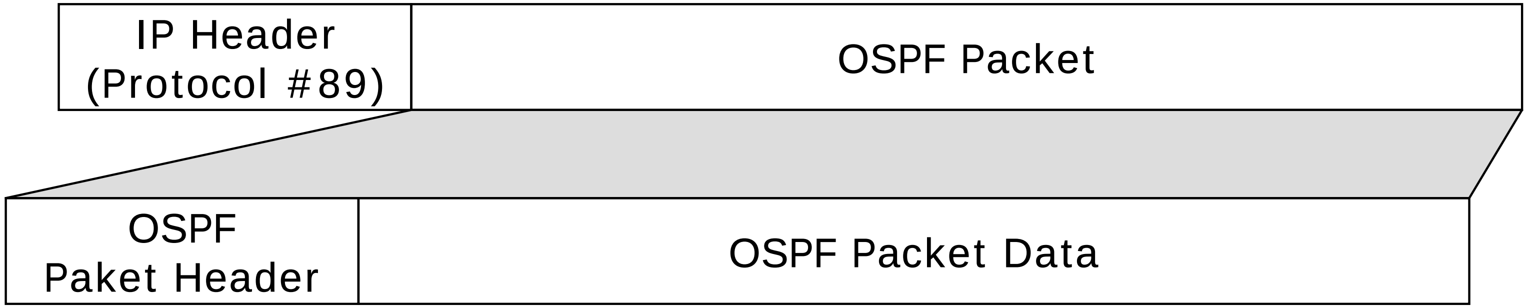


Router LSA (from 10.1.1.1)

32 Bits

8	8	8	8
Age = 0		Options	LS Type = 1
Link State ID = 10.1.1.1			
Advertising Router = 10.1.1.1			
LS Sequence Number = 0x80000006			
LS Checksum = 0x9b47		Length = 60	
00000 0 0 0	0x00	Number Links = 3	
Link ID = 10.1.1.2			
Link Data = Interface Index 1			
Link Type = 1	# TOS = 0	Link Cost = 3	
Link ID = 10.1.1.3			
Link Data = Interface Index 2			
Link Type = 1	# TOS = 0	Link Cost = 5	
Link ID = 10.1.1.1			
Link Data = 255.255.255.255			
Link Type = 3	# TOS = 0	Link Cost = 0	

OSPF Messages - Overview



- 5 OSPF message types:
  - Hello
  - Database Description
  - Link State Request / Update / Ack
- OSPF messages are carried directly over IP with protocol number 89
- OSPF message exchange only between neighbouring routers (TTL=1)
- Destination address: unicast or multicast address, depending on the network configuration (point-to-point, multipoint (LAN), NBMA)

OSPF Messages - OSPF Packet Format

32 Bits

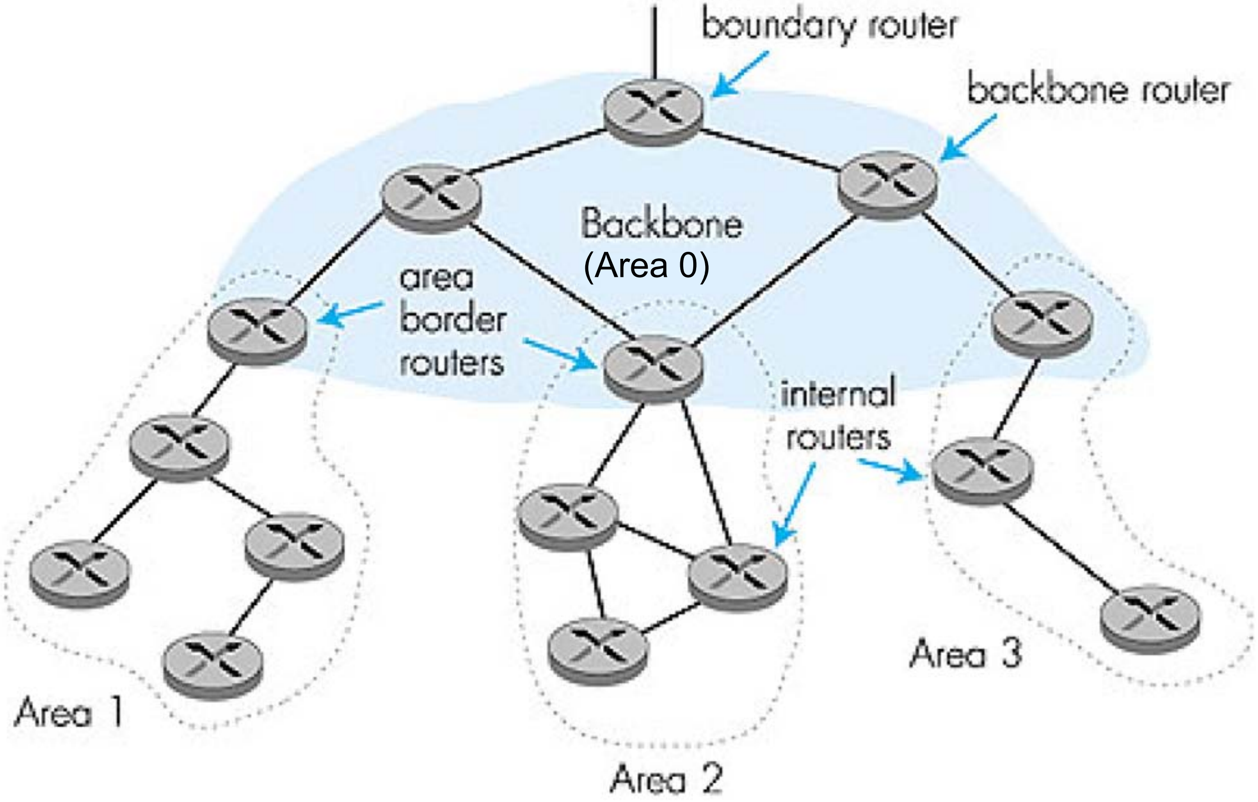
8	8	8	8
Version	Type	Length	
Router ID			
Area ID			
Checksum		AuType	
Authentication*)			
Authentication*)			
Packet Data			

*) if AuType = 2:

0x0000		Key ID	Auth. Length
Cryptogr. Sequence Number			

Hierarchical OSPF

- **Problem:** large IP networks require large LSDB and routing tables → large memory and high processing load in routers as well as high bandwidth requirement for flooding of OSPF messages
- **Solution:** partitioning of the network into separate **areas** and use of hierarchical routing:



Hierarchical OSPF - Principle

- Two hierarchy levels: backbone area (area 0) and areas; all areas have to be physically or logically (via virtual links) connected to the backbone area; flat routing within each area; routing between areas is only possible via the backbone area (strict hierarchy)
- Every area has its own LSDB
- Details of the area topology are only known within the area (i.e. are not known outside the area)
- Router and network LSAs are not flooded over area boundaries
- Routers, which belong to multiple area are called **Area Border Routers (ABR)**
- ABRs exchange information about areas via **Network Summary LSAs**: flooding of Summary LSAs of their areas into the backbone area; in a Summary LSAs the addresses of the area are aggregated to a longest prefix; link state ID = prefix address, cost = path cost to the most expensive destination network within the area

OSPF - Handling of external Routing Information

- OSPF is used as a routing protocol within an autonomous system (AS); **AS Boundary Routers (ASBR)** are located at the edge of an OSPF routing domain towards the outside world (other ASes)
- Routing information of other routing protocols can be inserted into the own network via **AS External LSAs** in order to find the best path out of the own network; AS External LSAs are also flooded over area boundaries

Comparison of different LS Routing Protocols

Properties	RIPv1	RIPv2
Area Concept (hierarchy)	X	X
Route Summarization	X	X
Equal Cost Multi Path	X	X
VLSM Support	X	X
Routing Algorithm	Dijkstra	IS-IS specific
Metric	arbitrary cost metric	arbitrary cost metric
Hop Limit	unlimited	1024
Scalability	good	excellent

Link State vs. Distance Vector Routing

- Control complexity:
  - link state: changes have to be sent to all other routers (flooding)
  - distance vector: changes are send to neighbouring routers
- Convergence speed:
  - link state: fast convergence, loop-free, oscillations are possible
  - distance vector: slow convergence, loops are possible, oscillations are possible (count to infinity problem)
- Robustness:
  - link state: separated route calculation → certain degree of robustness
  - distance vector: joint distributed route calculation (Bellmann-Ford) → a router might distribute incorrect paths to all destinations
- Conclusion:
  - link state routing shows a better convergence and robustness
  - distance vector routing is easier to implement