CS423: Lecture 22, Route Computation

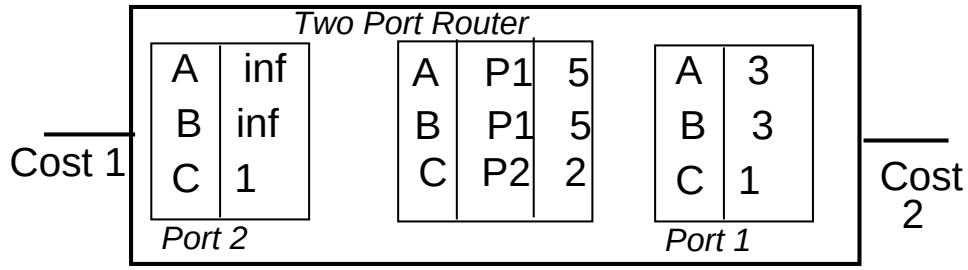
George Varghese

March 2, 2007

Distance Vector

- Two principal methods for Route Computation: distance vector (IP) and link state (OSI). We will study both.
- Distance vector: how can we use spanning tree protocol idea. We found distances to min ID node by "gossip". Use same idea for updating distance to all nodes.
- Previously we kept (Root, distance, parent). Now we keep a vector of (ID, distance). Hence called distance vector.

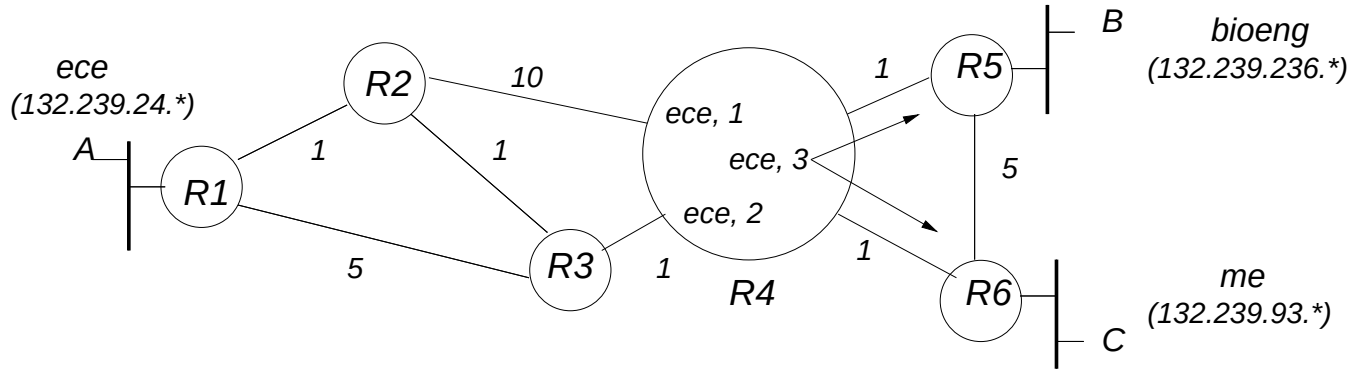
Distance Vector Databases

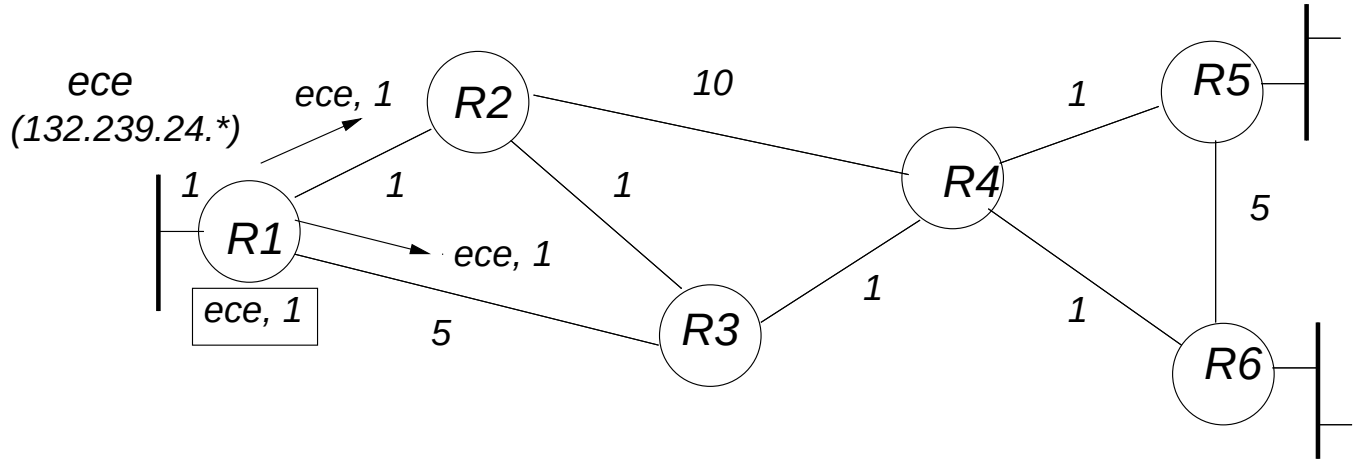


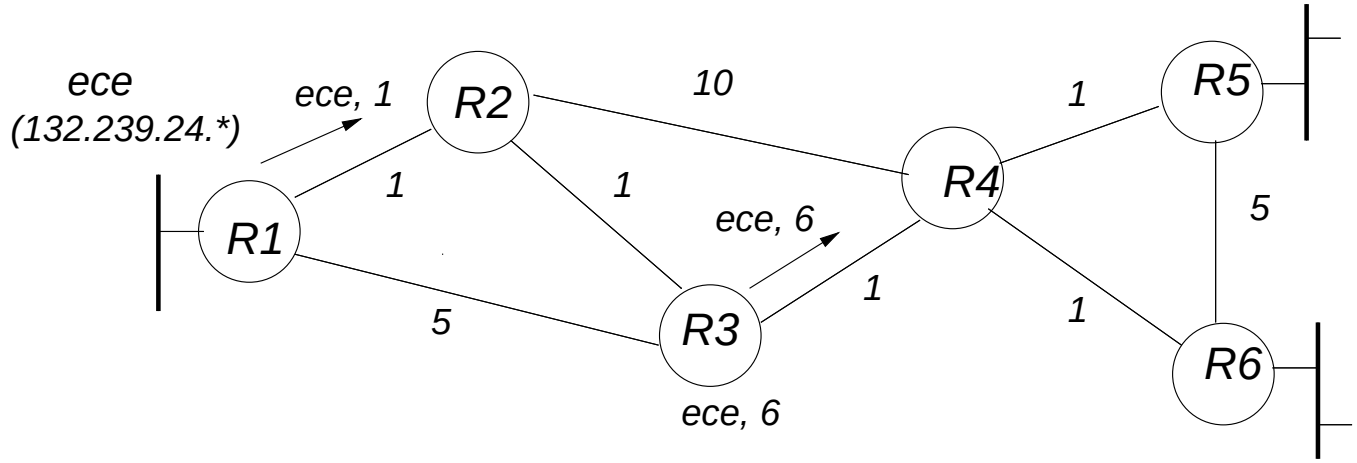
- As in Spanning Tree, we have port and central databases.
- Central is computed based on best port database.

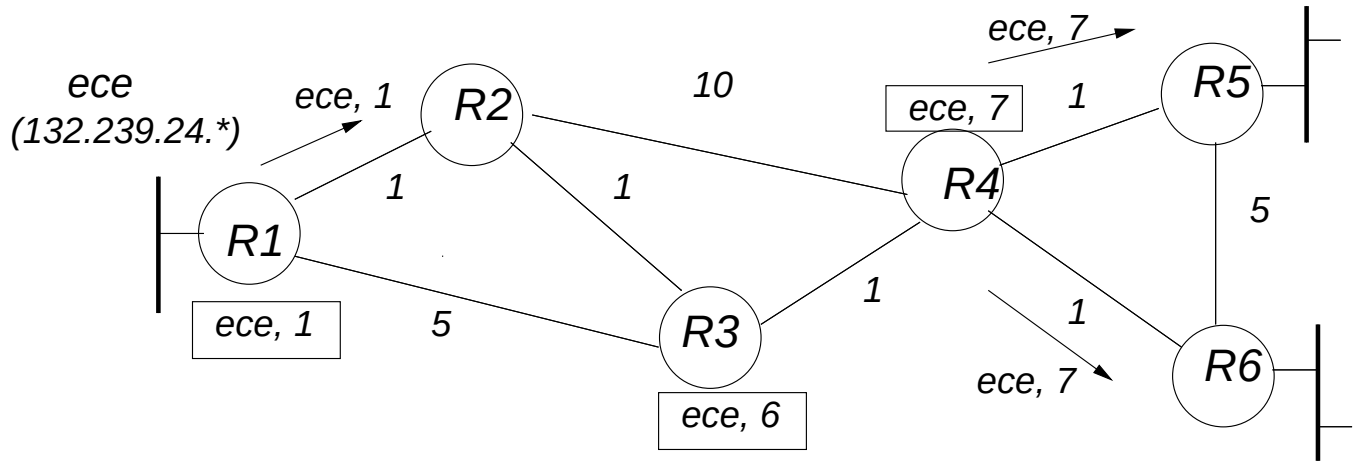
Link Failures and Distance Vector

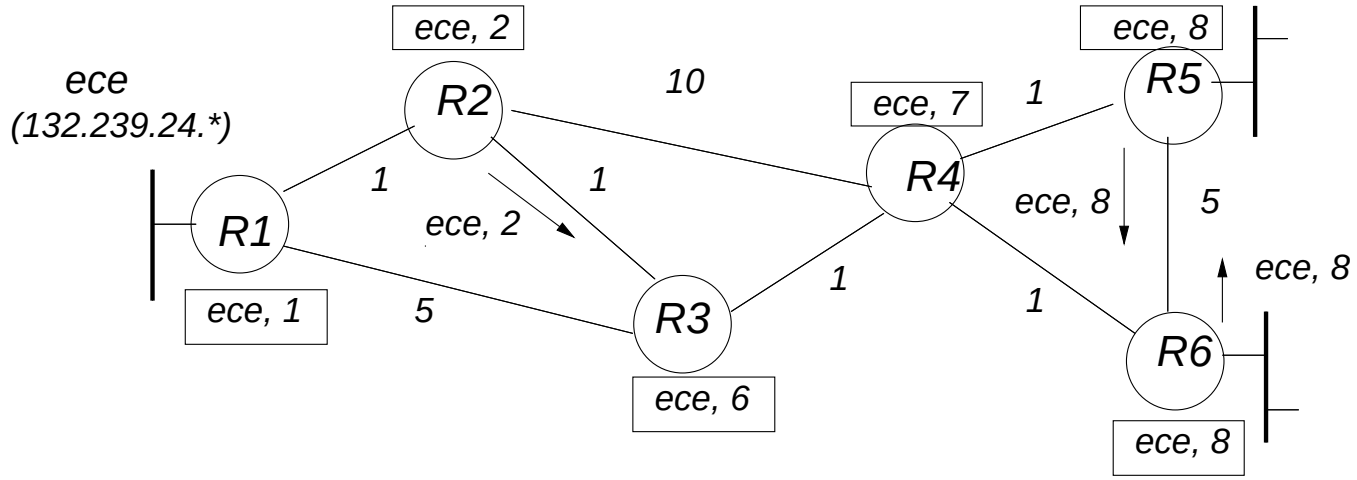
- On link failure, delete stored distance vector for that port. Link failure is reported by neighbor discovery (because we haven't received a hello from that neighbor for a while).
- Different from spanning tree in that there is no aging of information except the local aging used to detect link failures. Instead must rely on something call count-to-infinity.
- Also unlike spanning tree, you send whenever information changes. Periodic sending is a good idea for robustness but not strictly necessary.

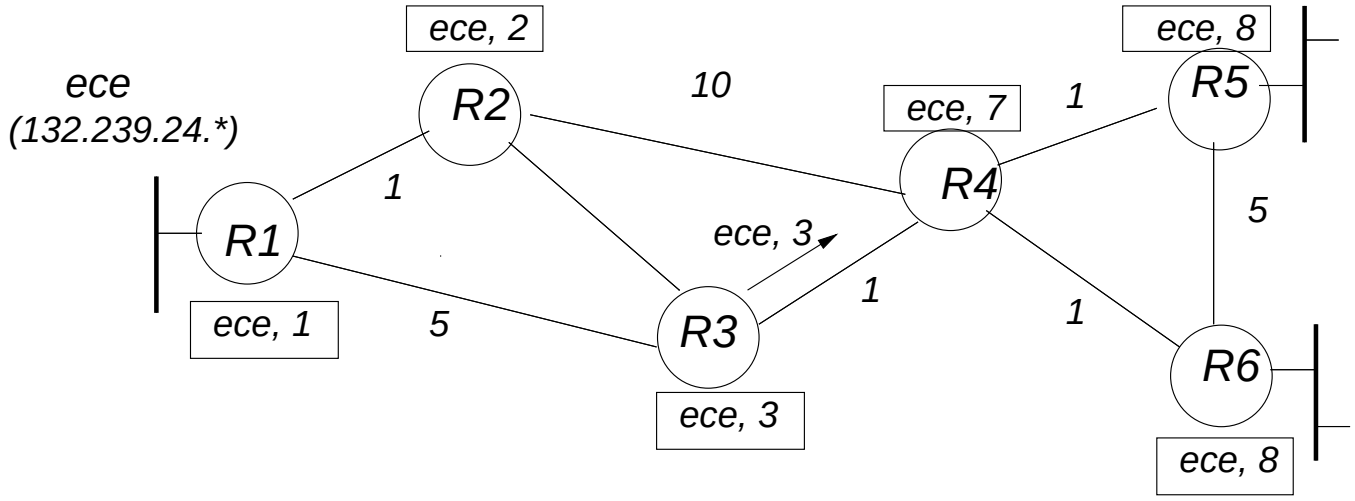

ece
(132.239.24.*)
A
R1
R2
R3
R4
R5
R6
B
bioeng
(132.239.236.*)
C
me
(132.239.93.*)
1
1
10
5
1
ece, 1
ece, 3
ece, 2
1
1
5

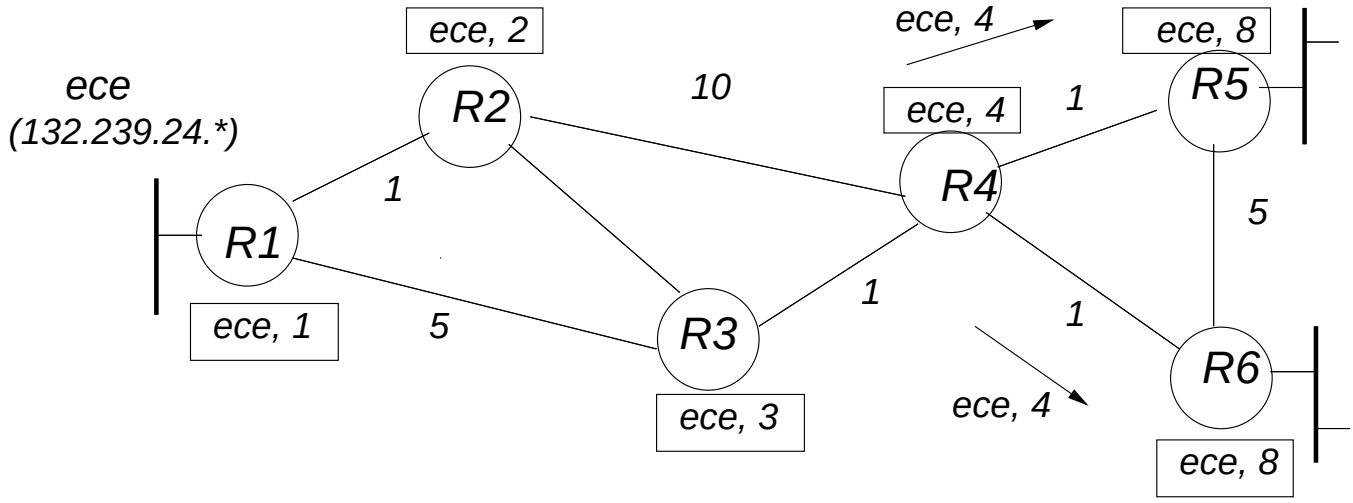

ece
(132.239.24.*)
ece, 1
R2
10
1
R5
1
1
1
R1
R4
5
ece, 1
ece, 1
1
1
5
R3
R6

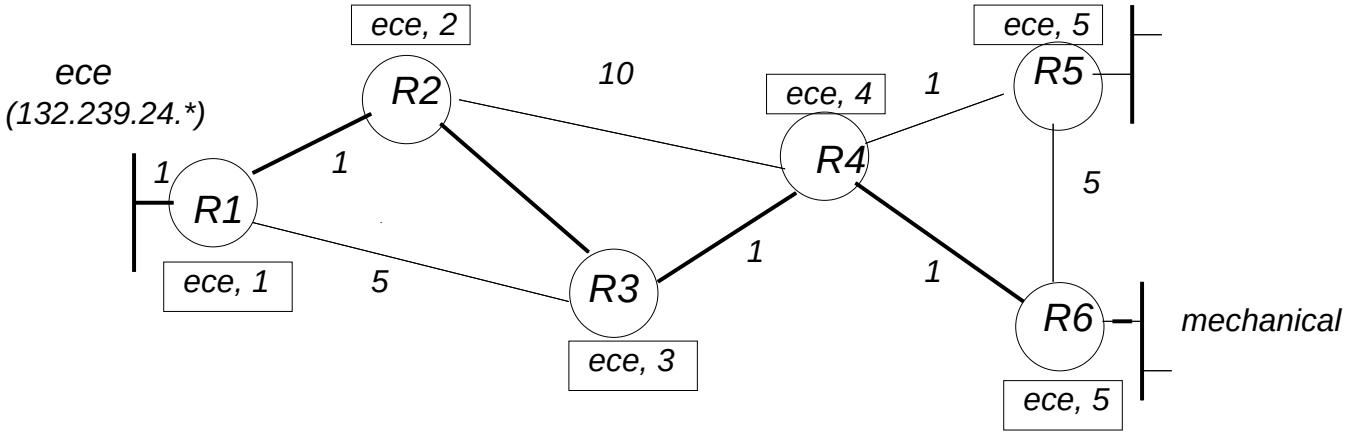

ece
(132.239.24.*)
ece, 1
R2
10
1
R5
R1
1
1
ece, 6
R4
5
5
1
1
R3
R6
ece, 6

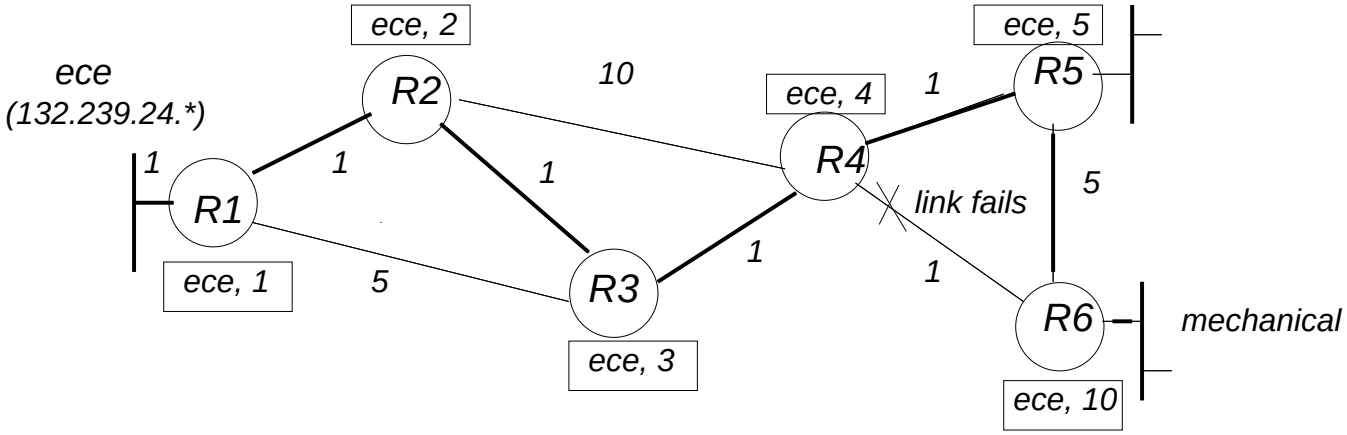

ece
(132.239.24.*)
ece, 1
R2
10
ece, 7
1
R5
ece, 7
R4
1
1
5
R1
ece, 1
5
R3
1
1
ece, 7
R6
ece, 6

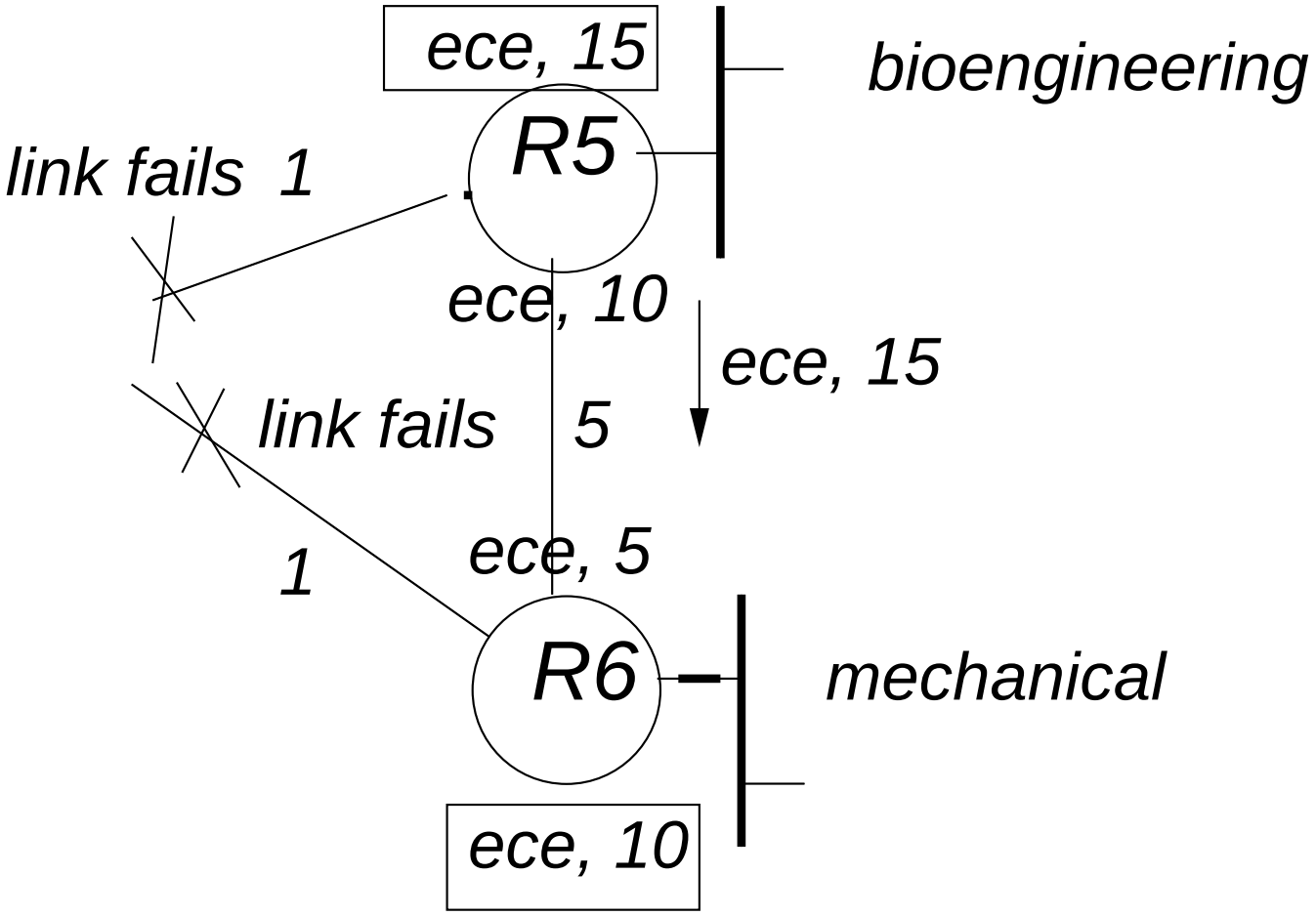

ece
(132.239.24.*)
ece, 2
R2
10
ece, 7
1
ece, 8
R5
R1
1
1
R4
ece, 8
5
ece, 2
ece, 8
ece, 1
5
R3
1
1
R6
ece, 6
ece, 8

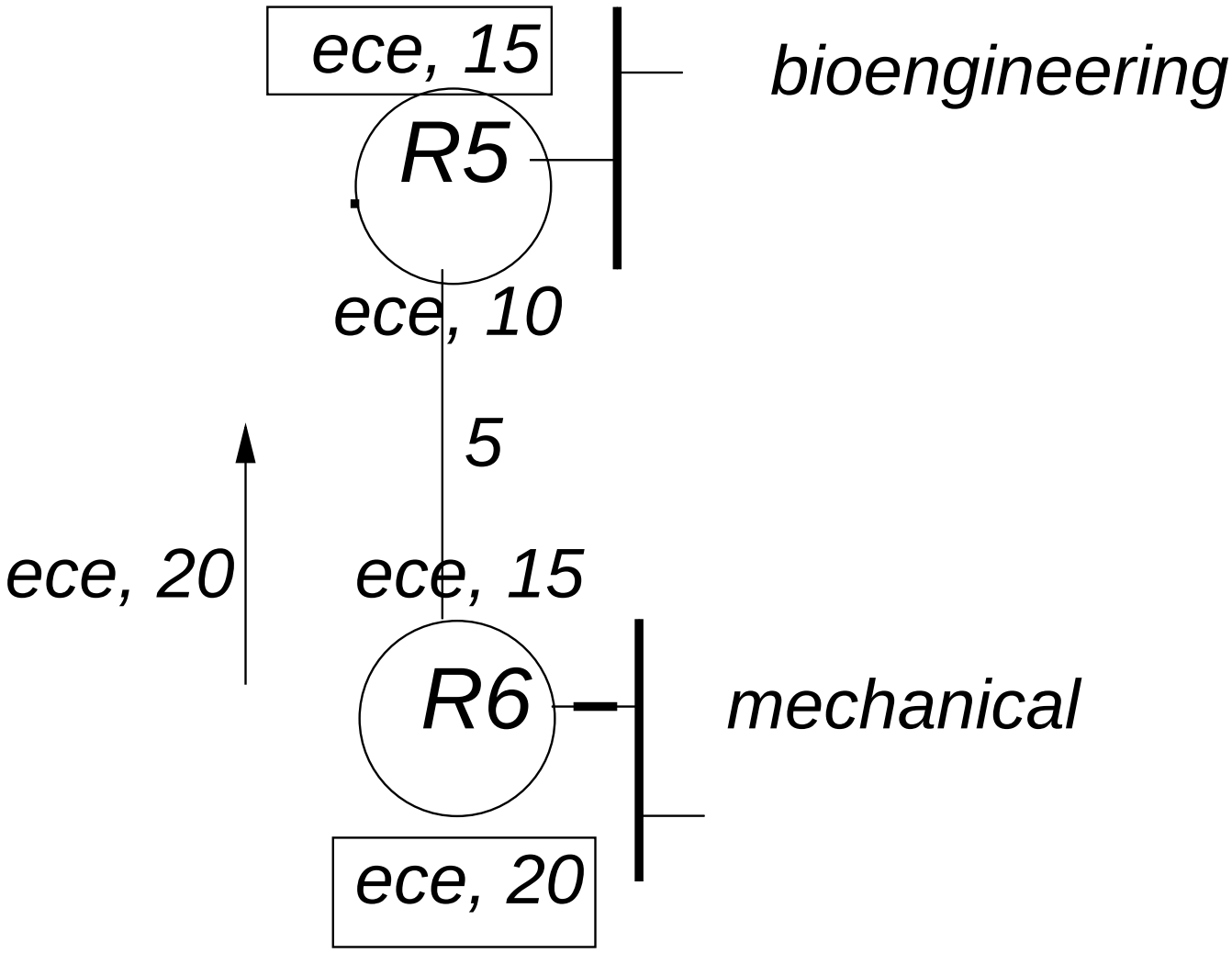

ece
(132.239.24.*)
ece, 2
R2
10
ece, 8
R5
ece, 7
1
R4
1
5
R1
ece, 3
ece, 1
5
R3
1
1
R6
ece, 3
ece, 8

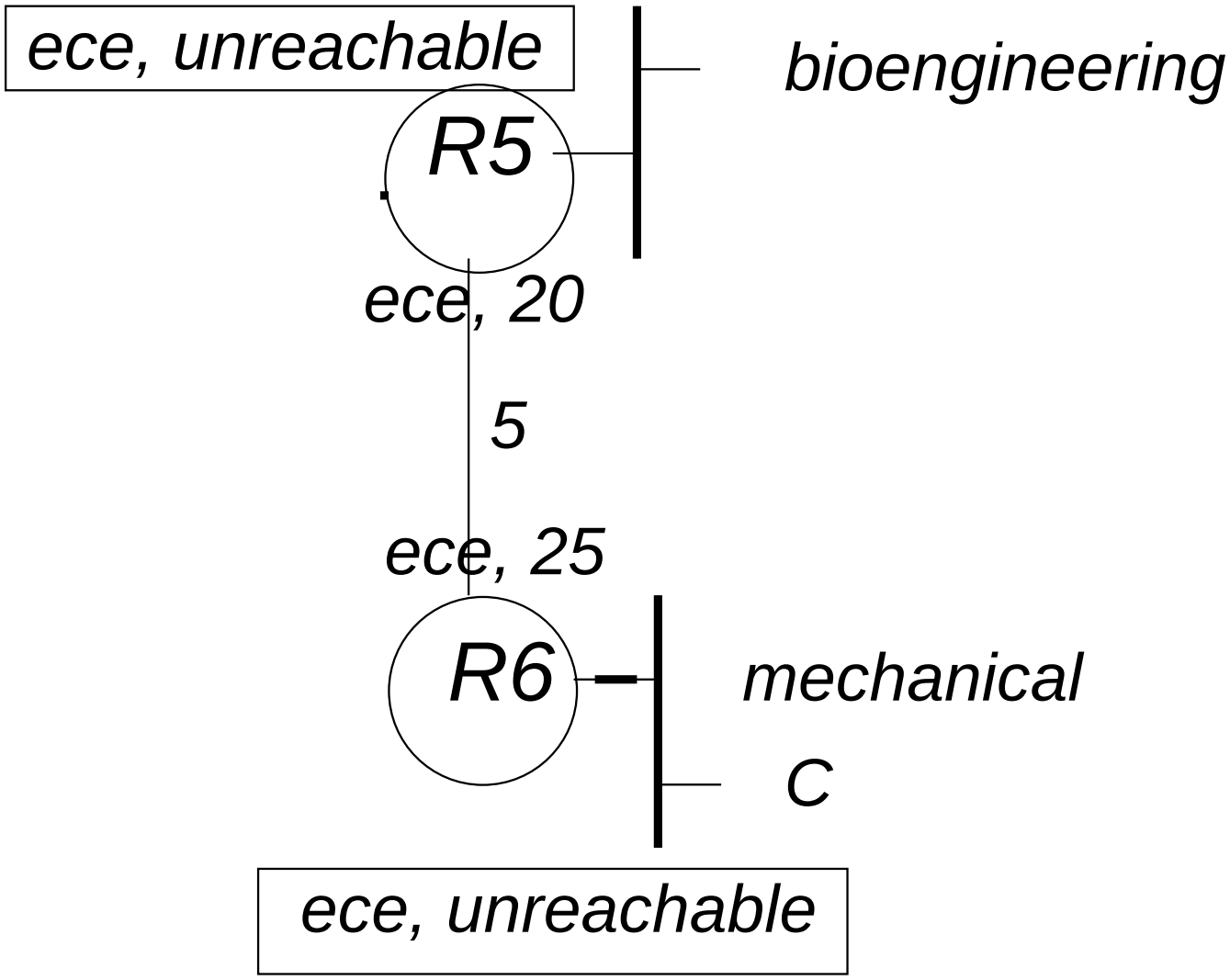

ece
(132.239.24.*)
ece, 2
R2
10
ece, 4
ece, 4
ece, 8
R5
1
R4
1
5
R1
ece, 1
5
R3
1
1
R6
ece, 4
ece, 3
ece, 8


ece
(132.239.24.*)
ece, 2
R2
10
ece, 4
1
ece, 5
R5
R4
1
1
5
R1
1
5
1
1
R3
R6
mechanical
ece, 1
ece, 3
ece, 5

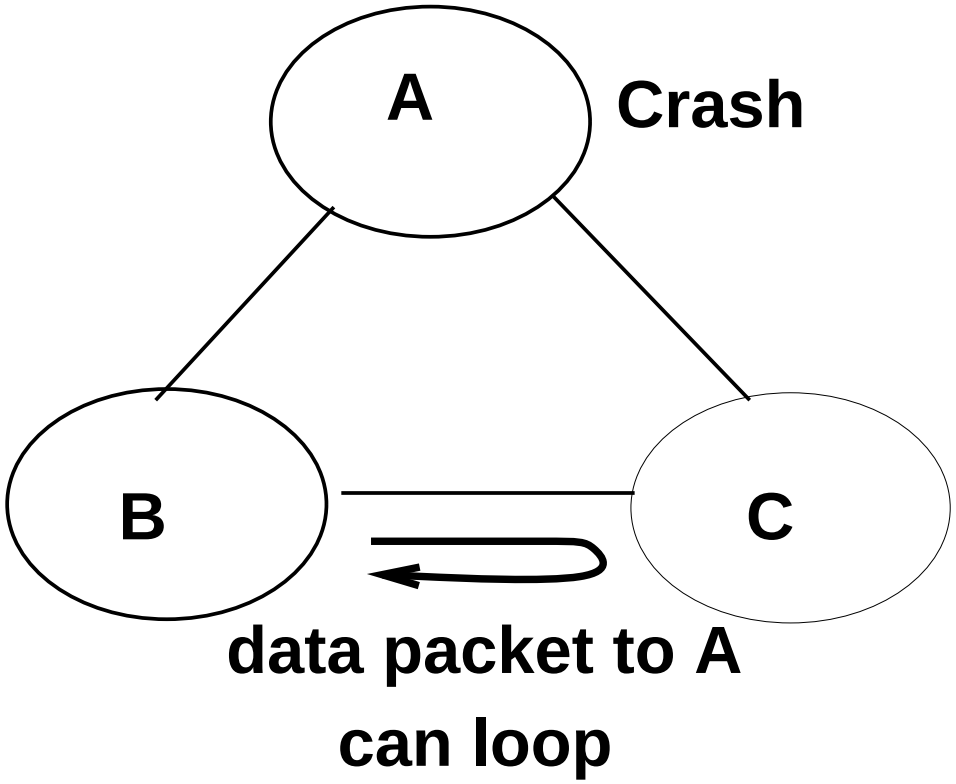

ece
(132.239.24.*)
ece, 2
R2
10
ece, 4
1
R5
ece, 5
1
1
1
R4
R1
link fails
5
1
1
5
ece, 1
R3
R6
mechanical
ece, 3
ece, 10


ece, 15
bioengineering
R5
link fails 1
ece, 10
ece, 15
link fails 5
1
ece, 5
R6
mechanical
ece, 10


ece, 15
bioengineering
R5
ece, 10
5
ece, 20
ece, 15
R6
mechanical
ece, 20


ece, unreachable
bioengineering
R5
ece, 20
5
ece, 25
R6
mechanical
C
ece, unreachable

Data Packet Looping



- After A crashes, B and C keep thinking the best way to get to A is through each other.
- Thus a data packet destined to A will keep looping until either the hop-count in the packet reaches its maximum value or B and C finally decide that A is down.

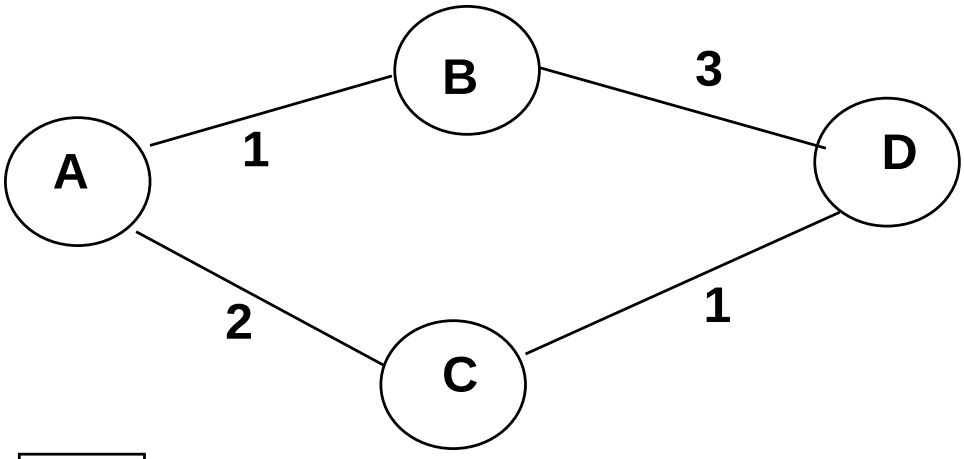
Link State History

- The ARPANET is a large national network that is part of the global Internet. (ARPANET has a net number). Classic network in a historic sense.
- Originally, ARPANET used distance vector. However, failure recovery times were very slow after node failures because of count-to-infinity problem. Also data packets kept looping during this period.
- New ARPANET moved to link state routing which has quicker response to failures and no count-up problem. Similar design used by OSI Routing. packets

Link State: the basic idea

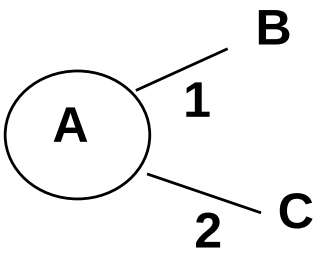
- Each node knows the default (or manager settable) cost of its outgoing links. Neighbor discovery is used to compile a list of neighbors that are UP. This information, along with link costs, is placed in a Link State Packet (LSP).
- Each source broadcasts its LSP to *all* other nodes using a primitive flooding mechanism called intelligent flooding.
- After the LSP propagation process stabilizes. each node has a complete and identical picture of the network graph. Then each node S uses any shortest path algorith (i.e., Dijkstra's) to compute the next node on the shortest path from S to every other node D.

LSP Generation

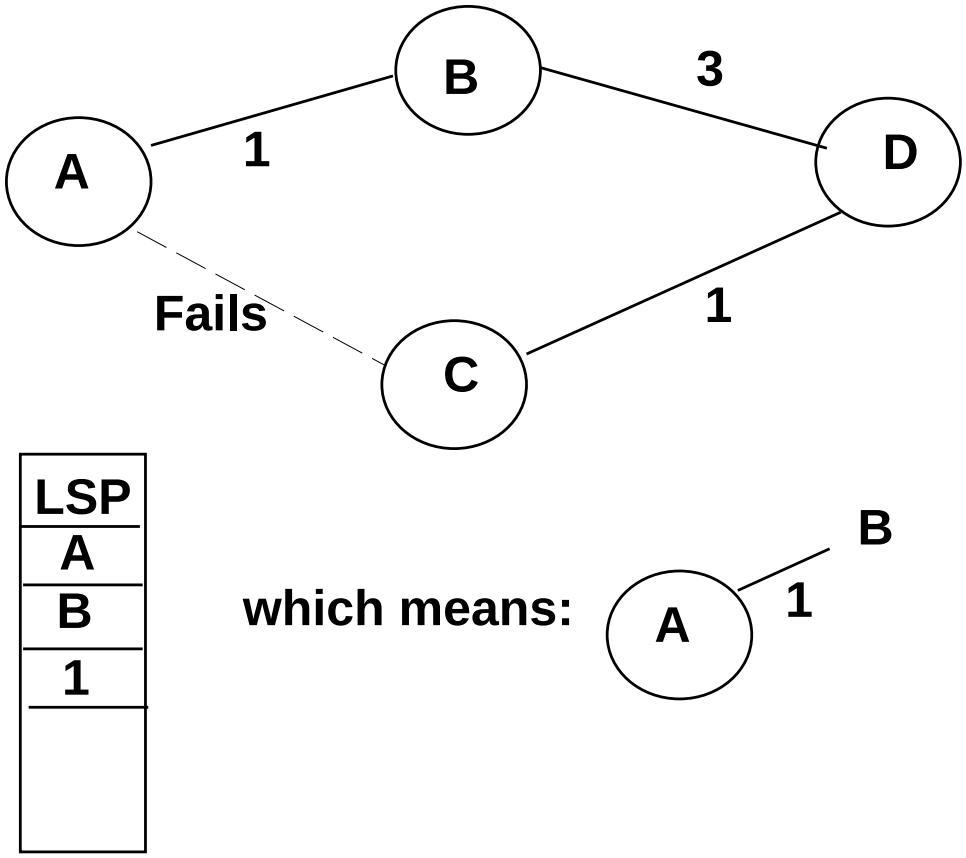


LSP
A
B
1
C
2

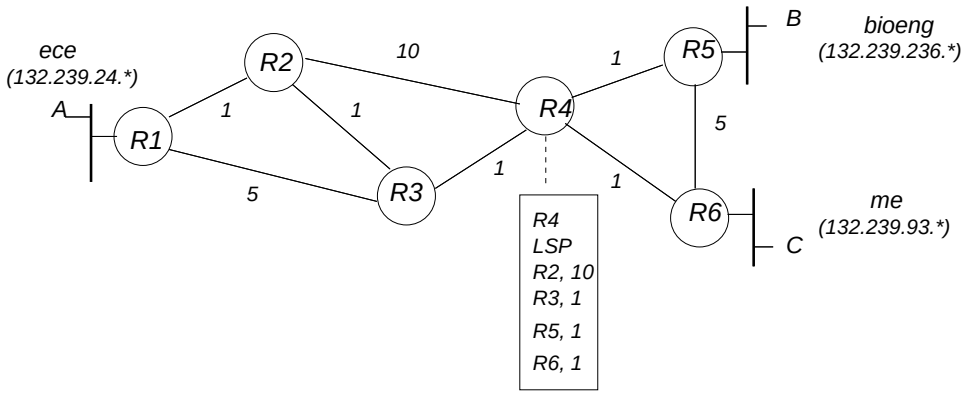
which means:

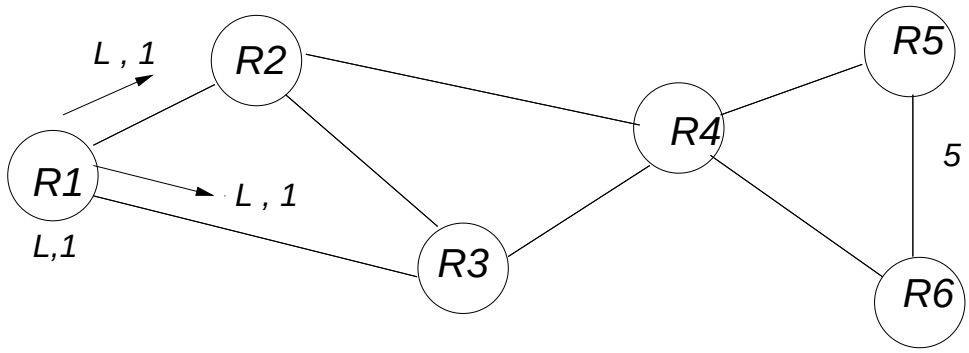


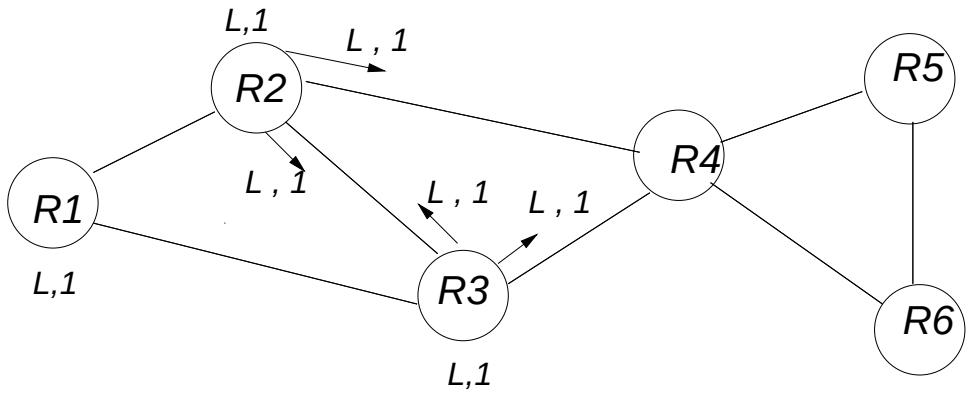
LSP Generation on Failure

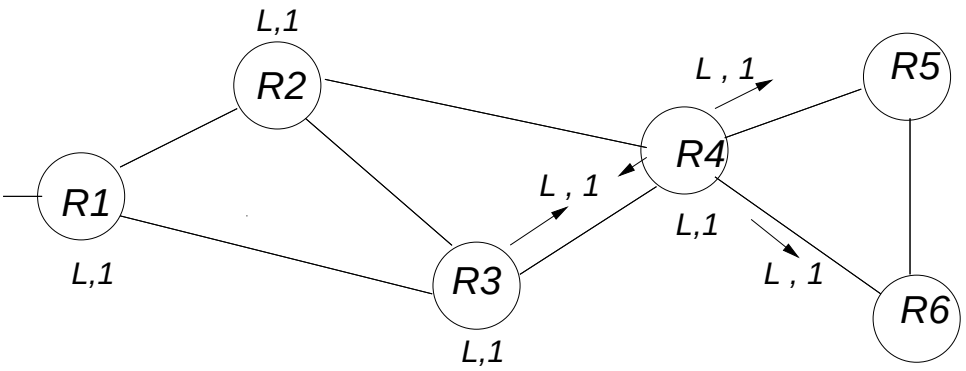


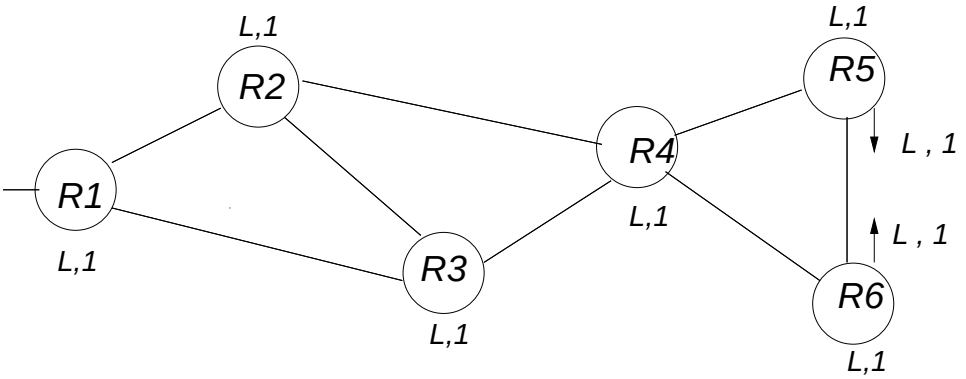
- If link AC fails, neighbor discovery in A and C will eventually detect failure.
- Only A and C recompute their LSP values and broadcast their LSPs again to all other nodes. Other nodes do not recompute or rebroadcast their LSPs.

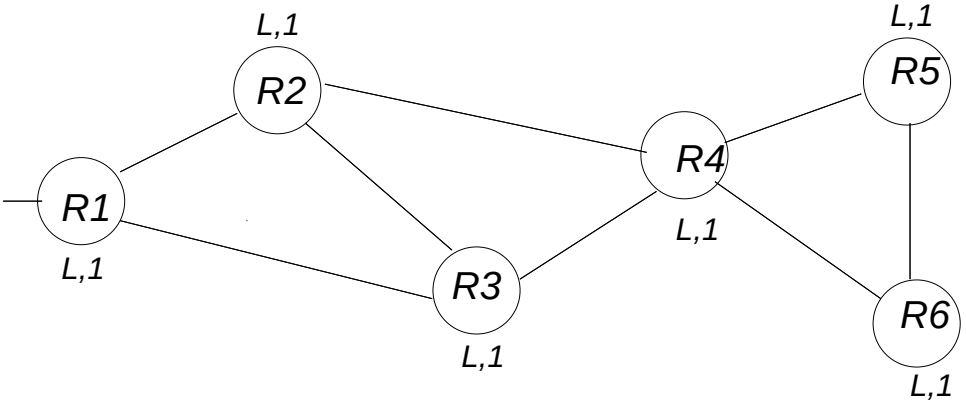

ece
(132.239.24.*)
A
R1
R2
R3
R4
R5
R6
B
bioeng
(132.239.236.*)
C
me
(132.239.93.*)
10
1
1
5
1
1
1
5
R4
LSP
R2, 10
R3, 1
R5, 1
R6, 1

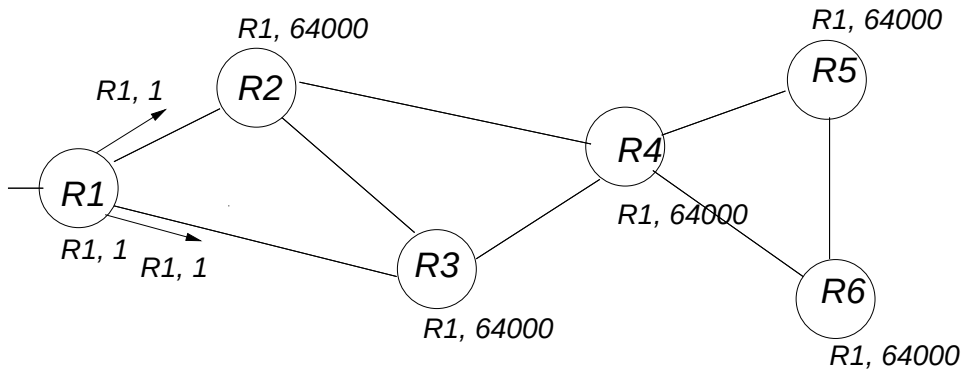

L , 1
R1
L , 1
L,1
R2
R3
R4
R5
5
R6

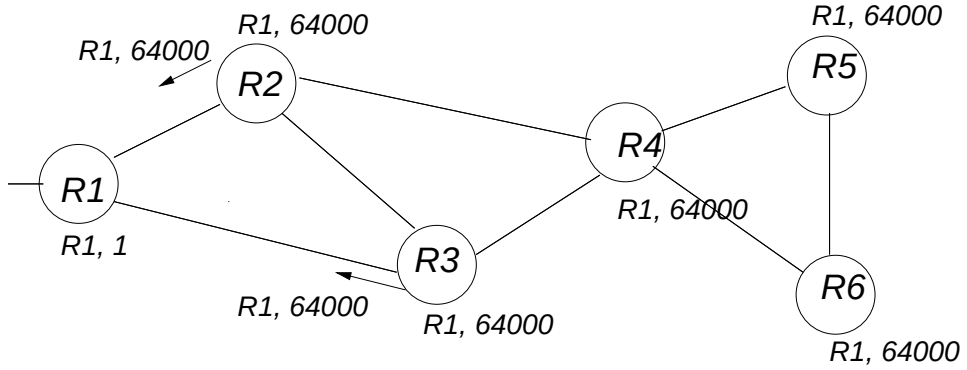

L,1
L , 1
R2
L , 1
R1
L,1
L , 1
L , 1
R3
L,1
R4
R5
R6

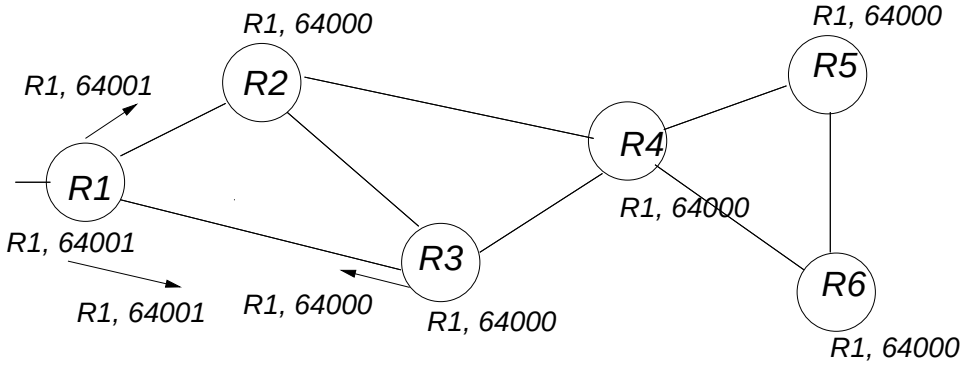

L,1
R2
L , 1
R5
R4
R1
L , 1
L,1
L , 1
R3
R6
L,1

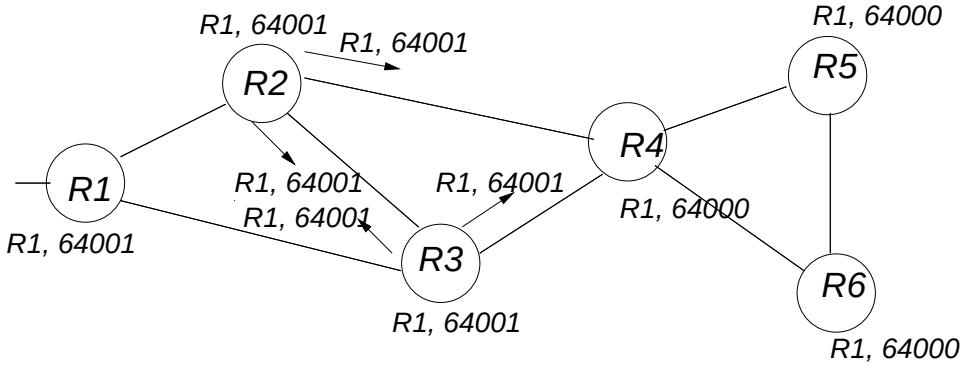

L,1
L,1
R5
R2
R4
L , 1
R1
L,1
L , 1
R3
R6
L,1
L,1

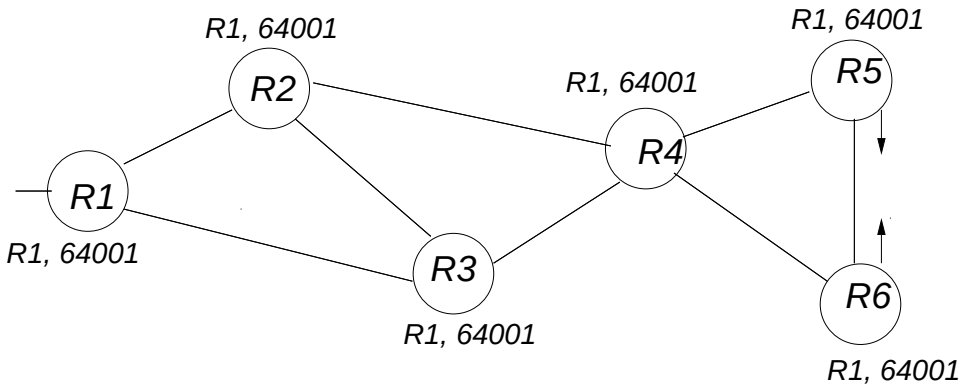

L,1
L,1
R2
R5
R4
R1
L,1
L,1
R3
R6
L,1
L,1

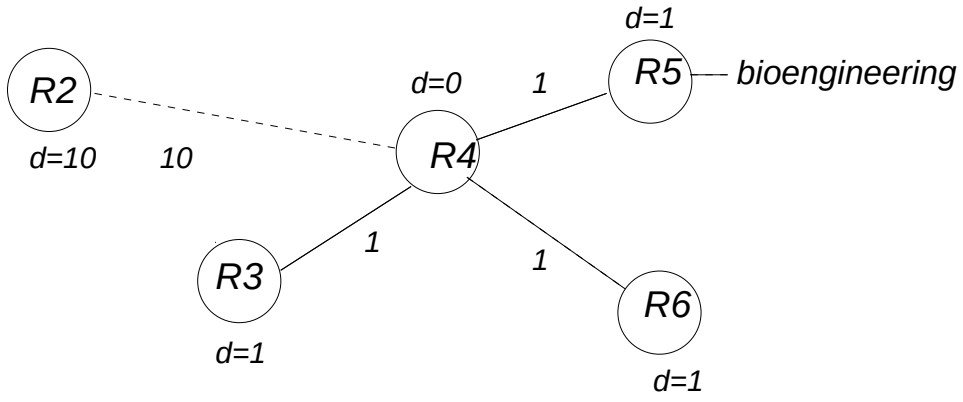

R1, 64000
R1, 64000
R1, 1
R2
R5
R4
R1
R1, 64000
R1, 1
R1, 1
R3
R6
R1, 64000
R1, 64000

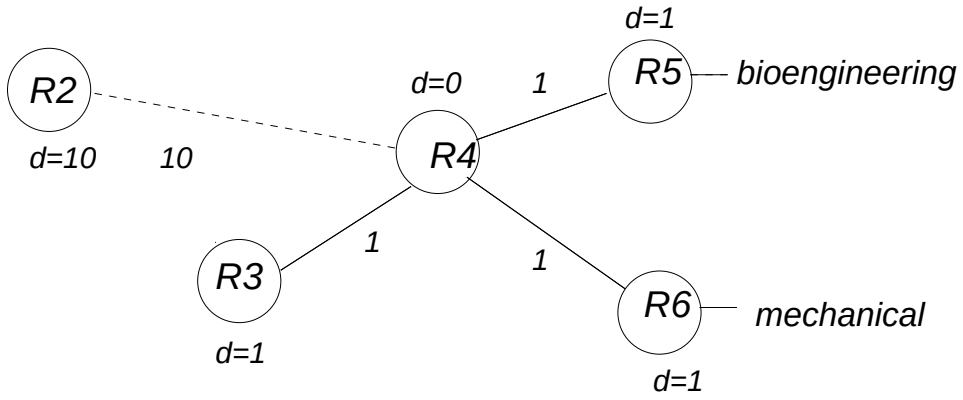

R1, 64000
R1, 64000
R1, 64000
R2
R5
R4
R1
R1, 64000
R1, 1
R3
R6
R1, 64000
R1, 64000
R1, 64000

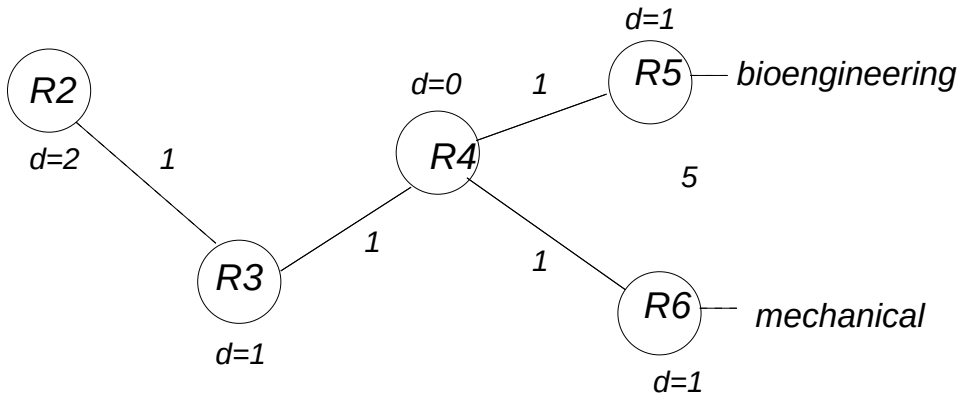

R1, 64000
R1, 64000
R1, 64001
R2
R5
R4
R1
R1, 64000
R1, 64001
R3
R6
R1, 64001
R1, 64000
R1, 64000
R1, 64000

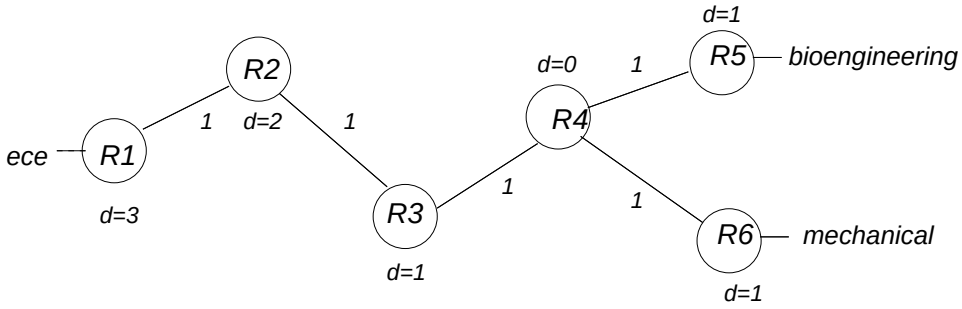

R1, 64001
R1, 64001
R1, 64000
R2
R5
R4
R1, 64001
R1, 64001
R1, 64000
R1
R1, 64001
R3
R6
R1, 64001
R1, 64000

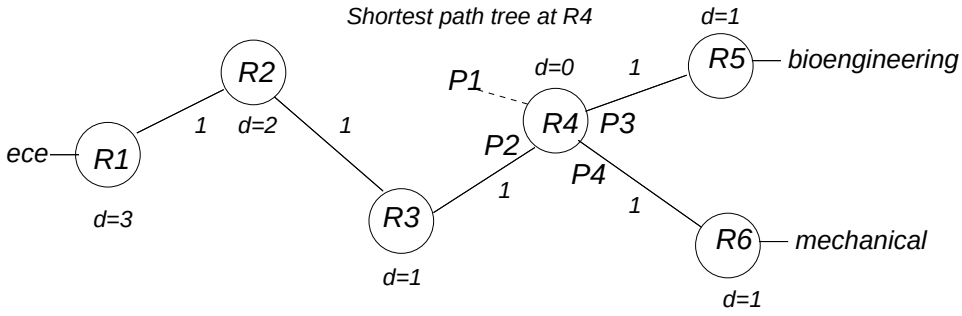

R1, 64001
R2
R1, 64001
R5
R1, 64001
R4
R1
R1, 64001
R3
R6
R1, 64001
R1, 64001


d=1
R2
d=0
1
R5
bioengineering
R4
d=10
10
1
1
R3
R6
d=1
d=1


d=1
R2
d=0
1
R5
bioengineering
R4
d=10
10
1
1
R3
R6
mechanical
d=1
d=1



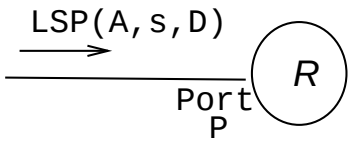




Prefix	Next hop interface
ece	P2
bioengineering	P3
mechanical	P4

Forwarding table at R4

LINK STATE CODE



RECEIVE LSP(A,s,D) on PORT P

```
   IF s < SEQ(A) THEN
      SEND STORED−LSP(A) ON PORT P
      ACK([A,P] = TRUE
   ELSEIF s > SEQ (A) THEN
     IF A = ME THEN (*source must jump*)
       SEQ(ME) = s + 1;
       SEND STORED−LSP(A) ON ALL PORTS
       FOR ALL PORTS Q,ACK[ME,Q] = TRUE
     ELSE
       STORED−LSP(A) = LSP(A,s, D)
       SEND ACK(A,s) ON PORT P
       SEND STORED−LSP(A) ON ALL PORTS P<>Q
       FOR ALL PORTS Q<>P,ACK[A,Q] = TRUE
   ELSE
      SEQ(ME) = SEQ(ME) + 1;
      SEND ACK(A,s) ON PORT P
      ACK([A,P] = FALSE
```

PERIODICALLY

```
  FOR ALL PORTS P and A with ACK[A,P]= TRUE
     SEND STORED−LSP(A) ON PORT P
```

RECEIVE ACK (A, s)

```
   IF s = SEQ(A) THEN
     ACK([A,P] = FALSE
```

LINK ON PORT P COMES UP

```
  FOR ALL SOURCES A DO
     SET ACK[A,P]= TRUE (*send all LSPs on P*)
```