Network Example 1

Simulation for 15 exchanges WITHOUT split-horizon heuristic of a network with 3 nodes

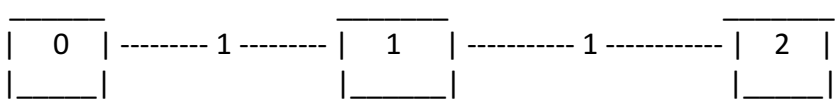
In this example I will try to demonstrate how the simulator handles a link fail.
Interesting exchanges:

- 2 - network reaches initial stable state
- 5 - link fails
- 6 - node1 and node2 notice link fail
- 7,8,9 - count-to-infinity
- 10 - recovery from count-to-infinity => network stable
- 13 - route-table entry timeout and network fully recovers from link fail

Input file:

```
-numOfNodes 3 -maxExchanges 15 -untilStability false -manual false -splitHorizon
off -infinity 10
0 1 1
1 2 1
## links changes
1 2 5 -1
## show best routes
0 2 0
0 2 1
0 2 2
0 2 3
0 2 4
0 2 5
0 2 6
0 2 7
0 2 8
0 2 9
## trace routing tables
0 0 15
1 0 15
2 0 15
```

Diagram of the network



Output and explanations:

Simulation start configuration:

Here you can how you have configured the simulator before the simulation actually starts. If you have defined "**-manual true**" the simulator will output this and start waiting for your command. From here you can the link cost matrix, as well as the initial states of all the nodes and their routing tables, as well as all the scheduled events and when will they happen.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!! Starting simulation !!!!!!!!!!!!!!!!!!!!!!!!!!!!
links costs:
       0      1      -1
       1      0      1
       -1     1      0

State of nodes:

NetworkNode 0:
neighbours: [node1 : cost 1;]
RouteTable for node : 0
       dest cost next
          0  0   null

NetworkNode 1:
neighbours: [node2 : cost 1;node0 : cost 1;]
RouteTable for node : 1
       dest cost next
          1  0   null

NetworkNode 2:
neighbours: [node1 : cost 1;]
RouteTable for node : 2
       dest cost next
          2  0   null


Scheduled events:
       At exchange 0 : [ShowBestRouteEvent from 0 to 2; TraceRouteTableEvent for
node 0; TraceRouteTableEvent for node 1; TraceRouteTableEvent for node 2; ]
       At exchange 1 : [ShowBestRouteEvent from 0 to 2; TraceRouteTableEvent for
node 0; TraceRouteTableEvent for node 1; TraceRouteTableEvent for node 2; ]
       At exchange 2 : [ShowBestRouteEvent from 0 to 2; TraceRouteTableEvent for
node 0; TraceRouteTableEvent for node 1; TraceRouteTableEvent for node 2; ]
       At exchange 3 : [ShowBestRouteEvent from 0 to 2; TraceRouteTableEvent for
node 0; TraceRouteTableEvent for node 1; TraceRouteTableEvent for node 2; ]
       At exchange 4 : [ShowBestRouteEvent from 0 to 2; TraceRouteTableEvent for
node 0; TraceRouteTableEvent for node 1; TraceRouteTableEvent for node 2; ]
       At exchange 5 : [LinkCostChangeEvent: 1, newCost: -1; ShowBestRouteEvent
from 0 to 2; TraceRouteTableEvent for node 0; TraceRouteTableEvent for node 1;
TraceRouteTableEvent for node 2; ]
       At exchange 6 : [ShowBestRouteEvent from 0 to 2; TraceRouteTableEvent for
node 0; TraceRouteTableEvent for node 1; TraceRouteTableEvent for node 2; ]
       At exchange 7 : [ShowBestRouteEvent from 0 to 2; TraceRouteTableEvent for
node 0; TraceRouteTableEvent for node 1; TraceRouteTableEvent for node 2; ]
       At exchange 8 : [ShowBestRouteEvent from 0 to 2; TraceRouteTableEvent for
node 0; TraceRouteTableEvent for node 1; TraceRouteTableEvent for node 2; ]
       At exchange 9 : [ShowBestRouteEvent from 0 to 2; TraceRouteTableEvent for
node 0; TraceRouteTableEvent for node 1; TraceRouteTableEvent for node 2; ]
```

Exchange 0

Exchange 0 is the first exchange in the network after it has been set-up. There are two types of events scheduled to happen at this exchange - show best route between node0 and node2 and traceRouteTable for all the nodes (0, 1, 2). This means that after the exchange is completed, the simulator will output the current best route from node0 to node2 as well as all the routing tables for all the nodes in the network.

From the output we can deduce that:

1. There ware changes in the routing tables of nodes 0, 1 and 2. Since this is the first exchange it means that initially all the nodes knew was a route to themselves. However, when the exchange happened new routes were included based on what the neighbors of a node have told the node. That is why we see changes in the route tables of all nodes.
2. After exchange 0 node0 knows a route to node1, but not to node2, node1 knows a route to both node1 and node2, and node2 knows a route to node1 and node0. This is because of the random, but sequential way the nodes exchange routing tables. What has happened is that node0 received node1 routing table before node2 has had a chance to tell node1 about itself. So when node0 receives node1 routing table it sees only a route to node1 and updates its own routing table to reflect that. Later node1 receives node0 routing table and appends a route to node0. After that node2 receives node1's routing table and sees routes both to node1 and node0.
3. There is no known route from node0 to node2. This is because from node0 point of view node2 does not exist in the network.

```
======================== Round 0 ==========================
simulate network exchange start
simulate network exchange finish

------- !!! ------- There were a changes in node routing tables ------- !!! -------
Nodes with changed routing tables:
0 1 2

############ ScheduledEvents after exchange 0 ############
ShowBestRouteEvent from 0 to 2
       0 – >  next hop unknown yet

TraceRouteTableEvent for node 0
RouteTable for node : 0
       dest cost next
          0    0 null
          1    1    1

TraceRouteTableEvent for node 1
RouteTable for node : 1
       dest cost next
          0    1    0
          1    0 null
          2    1    2

TraceRouteTableEvent for node 2
RouteTable for node : 2
       dest cost next
          0    2    1
          1    1    1
          2    0 null

======================== End round 0 ==========================
```

Exchange 1

This is the second exchange. From the simulator's initial state output we know that the simulator will output the routing tables of all the nodes and the best route from node0 to node2.

From the output we can deduce:

1. There was something altered only in node0's routing table. This is expected. After exchange 0 we saw that only node0 does not know about node2, but node1 does. So when node0 receives node1's routing table it sees the entry for node2 and updates its own routing table. Because of the small size and lack of link cost changes node1 and node2 knew about node0 and knew the minimum route costs to all other nodes so they did not register any updates in their routing tables. If we look at the tables after exchange 0 and exchange 1 we will notice that only routing table 0 has changed by now including a route to node2.
2. The best route from node0 to node2 can now be traced since node0 now knows about node2. The best route is 0 -> 1 -> 2.

```
========================= Round 1 ==========================
simulate network exchange start
simulate network exchange finish

------- !!! ------- There were a changes in node routing tables ------- !!! -------
Nodes with changed routing tables:
0

########### ScheduledEvents after exchange 1 ###########
ShowBestRouteEvent from 0 to 2
       0 – > 1 –> 2

TraceRouteTableEvent for node 0
RouteTable for node : 0
       dest cost next
          0    0 null
          1    1    1
          2    2    1

TraceRouteTableEvent for node 1
RouteTable for node : 1
       dest cost next
          0    1    0
          1    0 null
          2    1    2

TraceRouteTableEvent for node 2
RouteTable for node : 2
       dest cost next
          0    2    1
          1    1    1
          2    0 null

========================= End round 1 ==========================
```

Exchange 2,3,4

I group exchanges 2, 3 and 4 because nothing really exciting happens to the network. Again we have scheduled to track the routing tables for all nodes in the network as well as try to find the best route between node0 and node1.

From the output we can deduce:

1. The network was in a stable state. This means no node has changed its routing table during the exchange to acknowledge a better route or link cost change
2. The best route between node0 and node2 is still 0 -> 1 -> 2
3. Routing tables have indeed not changed.

```
======================== Round 2,3,4 ==========================
simulate network exchange start
simulate network exchange finish

------- !!! ------- Network stable for this round ------- !!! ------

########### ScheduledEvents after exchange 2,3,4 ###########
ShowBestRouteEvent from 0 to 2
       0 – > 1 –> 2

TraceRouteTableEvent for node 0
RouteTable for node : 0
       dest cost next
          0    0 null
          1    1    1
          2    2    1

TraceRouteTableEvent for node 1
RouteTable for node : 1
       dest cost next
          0    1    0
          1    0 null
          2    1    2

TraceRouteTableEvent for node 2
RouteTable for node : 2
       dest cost next
          0    2    1
          1    1    1
          2    0 null

======================== End round 2,3,4 ==========================
```

Exchange 5

This exchange yields the same results as exchanges 2,3 and 4. The only difference is that we have scheduled the link between node1 and node2 to change cost from 1 to -1. -1 reflects a failed link in the system so what we are basically saying is that there is now no connection between node1 and node2.

```
======================== Round 5 ==========================
simulate network exchange start
simulate network exchange finish

------- !!! ------- Network stable for this round ------- !!! ------

########### ScheduledEvents after exchange 5 ###########
LinkCostChangeEvent, link between 1 and 2 from 1 to -1

ShowBestRouteEvent from 0 to 2
       0 - > 1 -> 2

TraceRouteTableEvent for node 0
RouteTable for node : 0
       dest cost next
          0    0 null
          1    1    1
          2    2    1

TraceRouteTableEvent for node 1
RouteTable for node : 1
       dest cost next
          0    1    0
          1    0 null
          2    1    2

TraceRouteTableEvent for node 2
RouteTable for node : 2
       dest cost next
          0    2    1
          1    1    1
          2    0 null

======================== End round 5 ==========================
```

Exchange 6

During this exchange the network will have to propagate the changes in its topology.

From the output we can deduce:

1. During the exchange node1 and node2 notice that the link between them is gone. Both nodes update to reflect that they are not neighbors anymore. Each node also deletes any routes that have been learned from the other node for consistency.
2. The best route from node0 to node2 is now incomplete. There is no info what to do after node1 in the path.

```
======================== Round 6 ==========================
simulate network exchange start
Node 1 remove neighbour 2
Node 2 remove neighbour 1
Node 2 will remove entry because it came from neightbor 1 : RouteEntry{ dest: 0,
cost: 2, next:1}
simulate network exchange finish

------- !!! ------- There were a changes in node routing tables ------- !!! -------
Nodes with changed routing tables:
1 2

########### ScheduledEvents after exchange 6 ###########
ShowBestRouteEvent from 0 to 2
       0 – > 1 –>  next hop unknown yet

TraceRouteTableEvent for node 0
RouteTable for node : 0
       dest cost next
          0    0 null
          1    1    1
          2    2    1

TraceRouteTableEvent for node 1
RouteTable for node : 1
       dest cost next
          0    1    0
          1    0 null

TraceRouteTableEvent for node 2
RouteTable for node : 2
       dest cost next
          2    0 null

======================== End round 6 ==========================
```

Exchange 7, 8, 9

During those exchange we can see the counting to infinity problem. Node0 and node1 start to feed each other information about a route to node2. However because they send only destination -> minRouteLenght pairs there is no way for one of the nodes to notice that the advertised route from the other node includes the first node and thus there is no way to notice the cycle that has arose.

From the output we can deduce:

1. Routing tables for node1 and node0 are the only ones that get updated in during the exchanges. This is because they keep telling each other lies. The cost to node2 starts rising in both route tables. Node0 goes from 4 to 6 to 8 while node1 goes from 3 to 5 to 7.
2. The show-best-route can not find a route from node0 to node2. It also shows a cycle in the network.

```
======================== Round 7,8,9 ==========================
simulate network exchange start
simulate network exchange finish

------- !!! ------- There were a changes in node routing tables ------- !!! -------
Nodes with changed routing tables:
0 1

########### ScheduledEvents after exchange 7,8,9 ###########
ShowBestRouteEvent from 0 to 2
There is a cycle in the routes meaning the end node (2) has become unreachable!
       0 – > 1 –> 0

TraceRouteTableEvent for node 0
RouteTable for node : 0
       dest cost     next
          0    0     null
          1    1        1
          2    4,6,8    1

TraceRouteTableEvent for node 1
RouteTable for node : 1
       dest cost     next
          0    1        0
          1    0     null
          2    3,5,7    0

TraceRouteTableEvent for node 2
RouteTable for node : 2
       dest cost     next
          2    0     null

======================== End round 7,8,9 ==========================
```

Exchange 10

During this exchange we will be expecting to see the cost from node0 to node2 in node0's routing table to reach infinity defined as 10 in the configuration file.

From the output we can deduce:

1. Node0 notices that node2 is no longer reachable. This happens because the cost for reaching node2 from node0 becomes equal to the defined infinity cost 10. This causes node0 to drop the route from its routing table and not advertise it to its neighbors.

```
========================= Round 10 ==========================
simulate network exchange start
Node 0 dropping route for dest (2) because infinity (10) was reached.
simulate network exchange finish

------- !!! ------- There were a changes in node routing tables ------- !!! -------
Nodes with changed routing tables:
0 1

########### ScheduledEvents after exchange 10 ###########
TraceRouteTableEvent for node 0
RouteTable for node : 0
       dest cost next
          0    0 null
          1    1    1

TraceRouteTableEvent for node 1
RouteTable for node : 1
       dest cost next
          0    1    0
          1    0 null
          2    9    0

TraceRouteTableEvent for node 2
RouteTable for node : 2
       dest cost next
          2    0 null

========================= End round 10 ==========================
```

Exchange 11,12

They are not interesting at all.

From the output we can deduce:

1. The network is stable.
2. Node1's cost to node2 does not go up.

```
========================= Round 11,12 =========================
simulate network exchange start
simulate network exchange finish

------- !!! ------- Network stable for this round ------- !!! ------

########### ScheduledEvents after exchange 11,12 ###########
TraceRouteTableEvent for node 0
RouteTable for node : 0
       dest cost next
          0    0 null
          1    1    1

TraceRouteTableEvent for node 1
RouteTable for node : 1
       dest cost next
          0    1    0
          1    0 null
          2    9    0

TraceRouteTableEvent for node 2
RouteTable for node : 2
       dest cost next
          2    0 null

========================= End round 11,12 =========================
```

Exchange 13

During this round we get to see a timeout in the routing table of node1. After this exchange the network has finally recovered from the link fail during exchange 5.

From the output we can deduce:

1. The entry for node2 in node1's routing table expires. This happens because node1 has not heard anything about node2 from anywhere for the past 3 exchanges - 11,12,13.
2. The network has finally fully recovered from the link fail during exchange 5. The routing tables of all nodes represent the current topography of the network without any redundant/outdated info.

```
======================== Round 13 ========================
simulate network exchange start
Node 1 removing timeout entry: RouteEntry{ dest: 2, cost: 9, next:0}
simulate network exchange finish

------- !!! ------- There were a changes in node routing tables ------- !!! -------
Nodes with changed routing tables:
1

########### ScheduledEvents after exchange 13 ###########
TraceRouteTableEvent for node 0
RouteTable for node : 0
       dest cost next
          0    0 null
          1    1    1

TraceRouteTableEvent for node 1
RouteTable for node : 1
       dest cost next
          0    1    0
          1    0 null

TraceRouteTableEvent for node 2
RouteTable for node : 2
       dest cost next
          2    0 null

======================== End round 13 ========================
```

Exchange 14

Final exchange. Nothing interesting.

```
======================== Round 14 ========================
simulate network exchange start
simulate network exchange finish

------- !!! ------- Network stable for this round ------- !!! ------

########### ScheduledEvents after exchange 14 ###########
 . . .
======================== End round 14 ========================
```

Simulation for 15 exchanges WITH split-horizon heuristic of a network with 3 nodes

This is the same network as the above but with split-horizon heuristic turned on.

Interesting exchanges:

- 2 - network reaches initial stable state
- 5 - link fails
- 6 - node1 and node2 notice link fail
- 7 - network back in stable state
- 9 - route-table entry timeout and network fully recovers from link fail

This breakdown shows how much of a difference can split-horizon have on network convergence. It takes 4 exchanges less for the network to fully recover from the link fail. And the difference will be even greater if we used the RIP default infinity cost of 16 instead of 10.

Input:

```
-numOfNodes 3 -maxExchanges 15 -untilStability false -manual false -splitHorizon on
-infinity 10
0 1 1
1 2 1
## links changes
1 2 5 -1
## show best routes
0 2 0
0 2 1
0 2 2
0 2 3
0 2 4
0 2 5
0 2 6
0 2 7
0 2 8
0 2 9
## trace routing tables
0 0 15
1 0 15
2 0 15
```

Output:

Exchange 0,1,2,3,4,5,6

During exchanges 0, 1, 2, 3, 4 and 5 the same things happen as before.
At exchange 2 the network becomes stable.
At exchange 5 the link between node1 and node2 fails.
At exchange 6 node1 and node2 register that they are no longer neighbors and update accordingly.

Exchange 7

This is the exchange which demonstrates very clearly how split-horizon saves the day.

From the output we can deduce:

1. Network is in stable state. This is because split-horizon is turned on. This means that while node0 and node1 still exchange routing tables, those routing tables do not include routes learned from the other node. This helps because it breaks the count-to-infinity and the network is stable only 1 exchange after the link has failed.

```
======================== Round 7 ==========================
simulate network exchange start
simulate network exchange finish

------- !!! ------- Network stable for this round ------- !!! ------

########### ScheduledEvents after exchange 7 ###########
ShowBestRouteEvent from 0 to 2
       0 – > 1 –>  next hop unknown yet

TraceRouteTableEvent for node 0
RouteTable for node : 0
       dest cost next
          0    0 null
          1    1    1
          2    2    1

TraceRouteTableEvent for node 1
RouteTable for node : 1
       dest cost next
          0    1    0
          1    0 null

TraceRouteTableEvent for node 2
RouteTable for node : 2
       dest cost next
          2    0 null

======================== End round 7 ==========================
```

Exchange 8

Network stable. Nothing interesting.

```
======================== Round 8 ========================
simulate network exchange start
simulate network exchange finish

------- !!! ------- Network stable for this round ------- !!! ------

########### ScheduledEvents after exchange 8 ###########
. . .
======================== End round 8 ========================
```

Exchange 9

Entry in node0's routing table timeouts after not hearing anything about node2 for 4 cycles. Network has finally fully recovered from link loss.

From the output we can deduce:

1. Node0's entry to node2 timeouts. This happens because at exchange 5 the link between node1 and node2 fails. Because of this node1 removes its entry to node2 from its routing table. At exchange 6,7,8 node1 and node0 exchange routing tables . Node1's routing table does not contain a route for node2 and because of the split-horizon when node0 does not include the route to node2 when it sends its routing table to node1 because node0 has learned the route to node2 from node1. This means that node0 stops receiving any updates for node2 so after the default 4 cycles the entry timeouts.
2. Network has finally fully recovered from link loss since there is no more redundant/outdated info.

```
======================== Round 9 ========================
simulate network exchange start
Node 0 removing timeout entry: RouteEntry{ dest: 2, cost: 2, next:1}
simulate network exchange finish

------- !!! ------- There were a changes in node routing tables ------- !!! -------
Nodes with changed routing tables:
0

########### ScheduledEvents after exchange 9 ###########
ShowBestRouteEvent from 0 to 2
       0 - >  next hop unknown yet

TraceRouteTableEvent for node 0
RouteTable for node : 0
       dest cost next
          0    0 null
          1    1    1

TraceRouteTableEvent for node 1
RouteTable for node : 1
       dest cost next
          0    1    0
          1    0 null

TraceRouteTableEvent for node 2
RouteTable for node : 2
       dest cost next
          2    0 null
```

========== End round 9 ==========

Exchange 10,11,12,13,14

Nothing interesting.