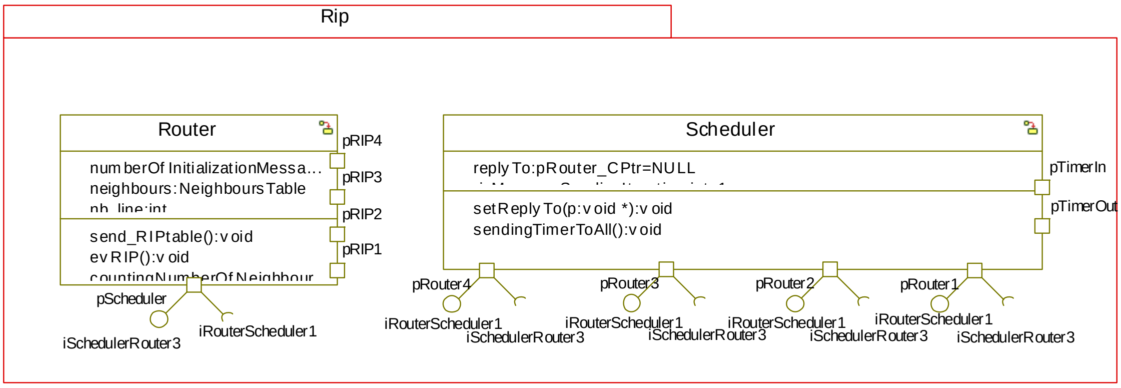
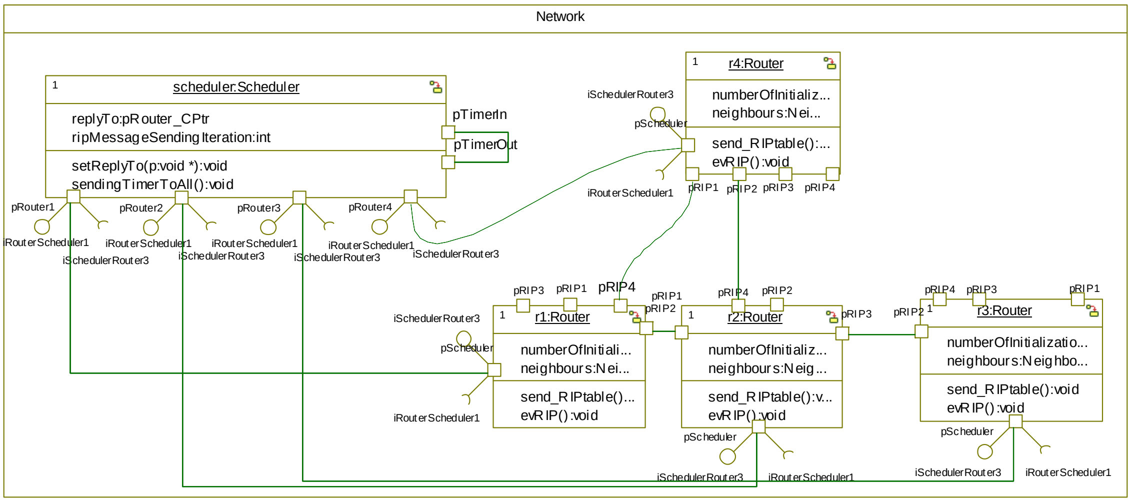
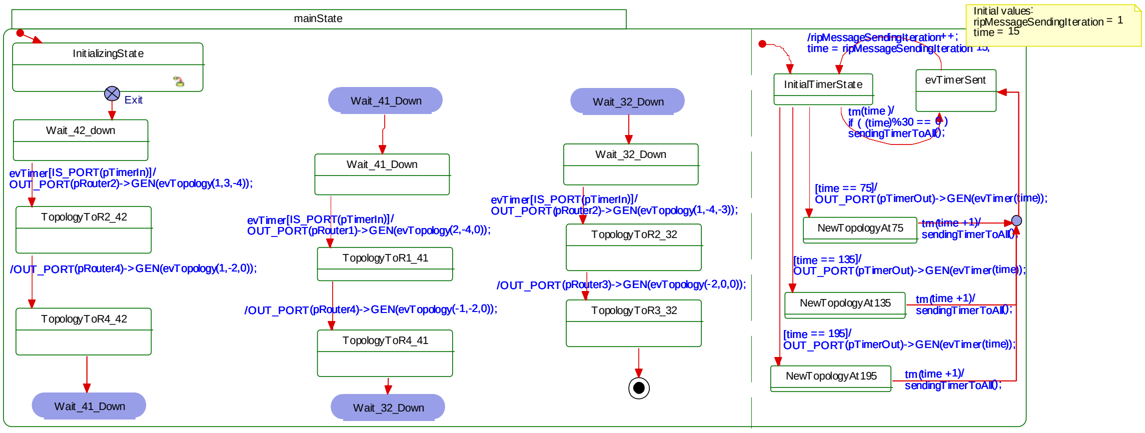
Object Model Diagram name: RIP



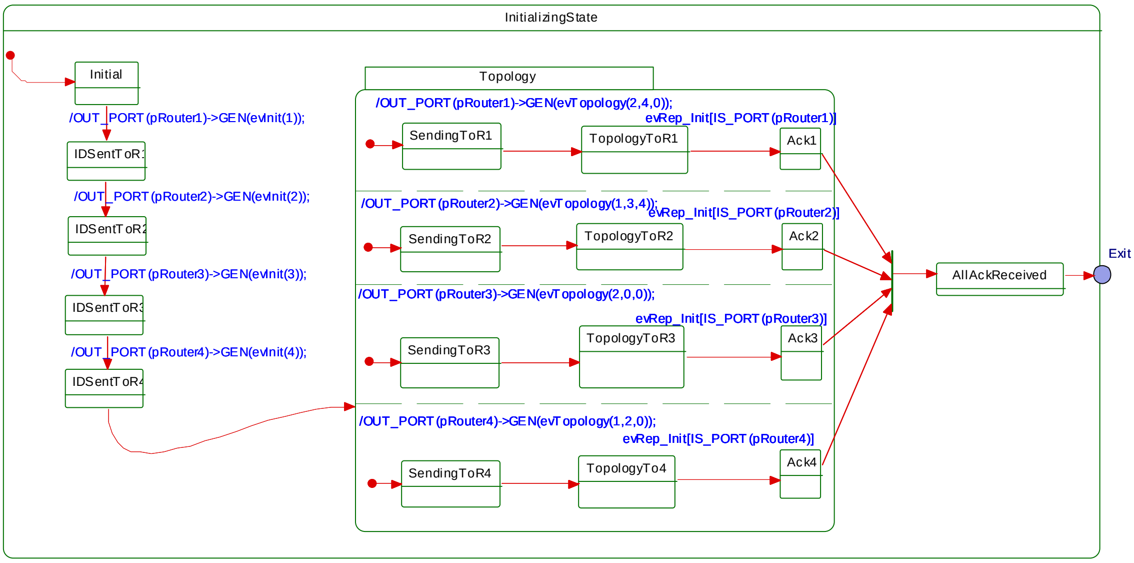
Structure Diagram name: structureDiagram



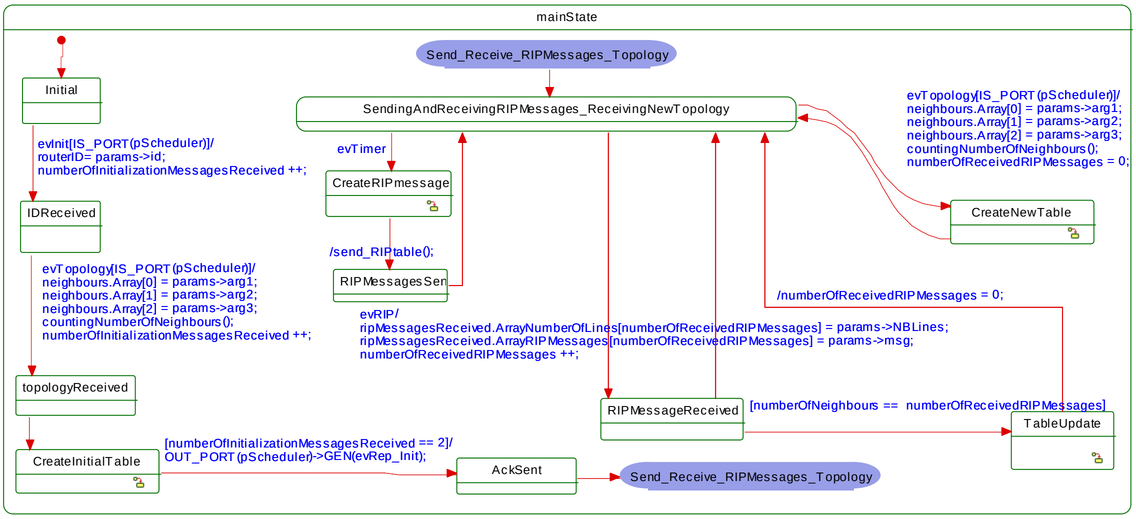
Statechart information for Class: Scheduler


mainState
InitializingState
Exit
Wait_42_down
evTimer[IS_PORT(pTimerIn)]/
OUT_PORT(pRouter2)->GEN(evTopology(1,3,-4));
TopologyToR2_42
/OUT_PORT(pRouter4)->GEN(evTopology(1,-2,0));
TopologyToR4_42
Wait_41_Down
Wait_41_Down
Wait_41_Down
evTimer[IS_PORT(pTimerIn)]/
OUT_PORT(pRouter1)->GEN(evTopology(2,-4,0));
TopologyToR1_41
/OUT_PORT(pRouter4)->GEN(evTopology(-1,-2,0));
TopologyToR4_41
Wait_32_Down
Wait_32_Down
Wait_32_Down
evTimer[IS_PORT(pTimerIn)]/
OUT_PORT(pRouter2)->GEN(evTopology(1,-4,-3));
TopologyToR2_32
/OUT_PORT(pRouter3)->GEN(evTopology(-2,0,0));
TopologyToR3_32
Initial values:
ripMessageSendingIteration = 1
time = 15
/ripMessageSendingIteration++;
time = ripMessageSendingIteration*15;
InitialTimerState
evTimerSent
tm(time)/
if ((time)%30 == 0)
sendingTimerToAll();
[time == 75]/
OUT_PORT(pTimerOut)->GEN(evTimer(time));
NewTopologyAt75
tm(time +1)/
sendingTimerToAll();
[time == 135]/
OUT_PORT(pTimerOut)->GEN(evTimer(time));
NewTopologyAt135
tm(time +1)/
sendingTimerToAll();
[time == 195]/
OUT_PORT(pTimerOut)->GEN(evTimer(time));
NewTopologyAt195
tm(time +1)/
sendingTimerToAll();

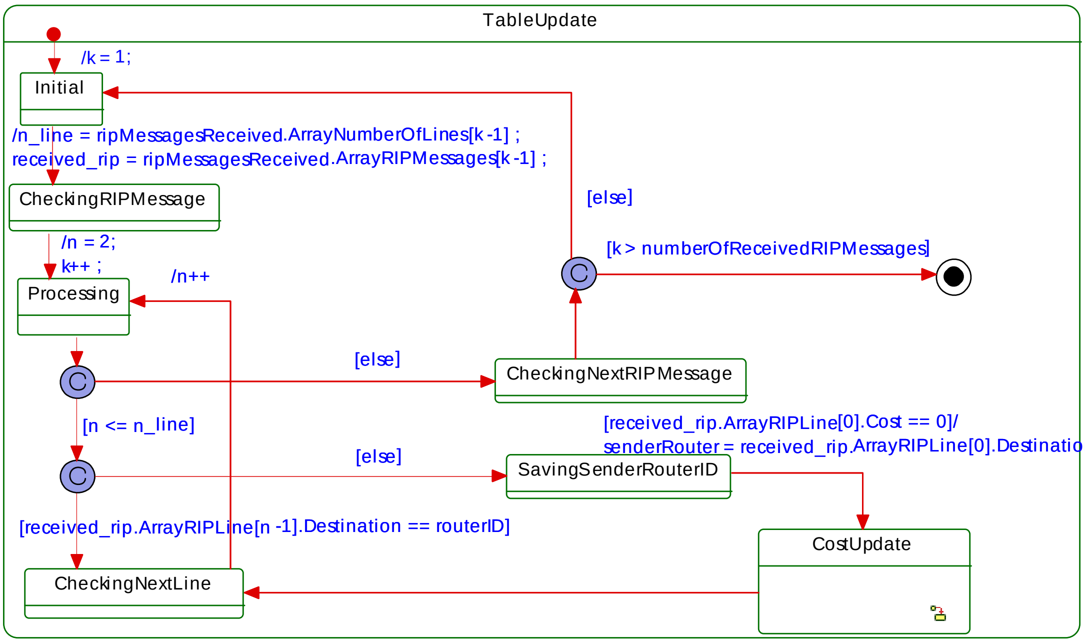
Sub-Statechart information for State InitializingState


InitializingState
Initial
/OUT_PORT(pRouter1)->GEN(evInit(1));
IDSentToR1
/OUT_PORT(pRouter2)->GEN(evInit(2));
IDSentToR2
/OUT_PORT(pRouter3)->GEN(evInit(3));
IDSentToR3
/OUT_PORT(pRouter4)->GEN(evInit(4));
IDSentToR4
Topology
/OUT_PORT(pRouter1)->GEN(evTopology(2,4,0));
evRep_Init[IS_PORT(pRouter1)]
SendingToR1
TopologyToR1
Ack1
/OUT_PORT(pRouter2)->GEN(evTopology(1,3,4));
evRep_Init[IS_PORT(pRouter2)]
SendingToR2
TopologyToR2
Ack2
/OUT_PORT(pRouter3)->GEN(evTopology(2,0,0));
evRep_Init[IS_PORT(pRouter3)]
SendingToR3
TopologyToR3
Ack3
/OUT_PORT(pRouter4)->GEN(evTopology(1,2,0));
evRep_Init[IS_PORT(pRouter4)]
SendingToR4
TopologyTo4
Ack4
AllAckReceived
Exit

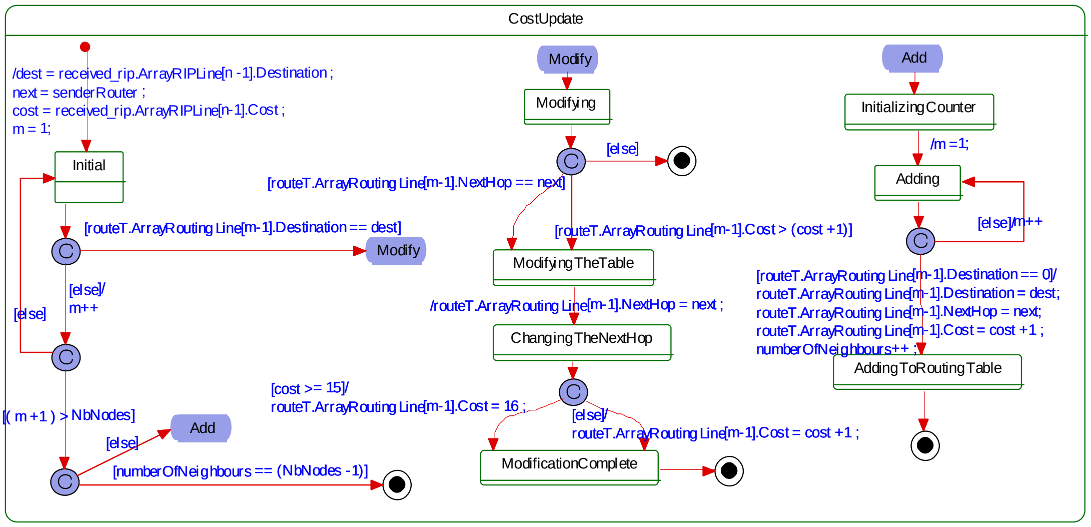
Statechart information for Class: Router


mainState
Send_Receive_RIPMessages_Topology
Initial
SendingAndReceivingRIPMessages_ReceivingNewTopology
evInit[IS_PORT(pScheduler)]/
routerID= params->id;
numberOfInitializationMessagesReceived ++;
evTimer
CreateRIPmessage
evTopology[IS_PORT(pScheduler)]/
neighbours.Array[0] = params->arg1;
neighbours.Array[1] = params->arg2;
neighbours.Array[2] = params->arg3;
countingNumberOfNeighbours();
numberOfReceivedRIPMessages = 0;
CreateNewTable
IDReceived
/send_RIPtable();
RIPMessagesSen
evTopology[IS_PORT(pScheduler)]/
neighbours.Array[0] = params->arg1;
neighbours.Array[1] = params->arg2;
neighbours.Array[2] = params->arg3;
countingNumberOfNeighbours();
numberOfInitializationMessagesReceived ++;
/numberOfReceivedRIPMessages = 0;
evRIP/
ripMessagesReceived.ArrayNumberOfLines[numberOfReceivedRIPMessages] = params->NBLines;
ripMessagesReceived.ArrayRIPMessages[numberOfReceivedRIPMessages] = params->msg;
numberOfReceivedRIPMessages ++;
topologyReceived
RIPMessageReceived
[numberOfNeighbours == numberOfReceivedRIPMessages]
TableUpdate
CreateInitialTable
[numberOfInitializationMessagesReceived == 2]/
OUT_PORT(pScheduler)->GEN(evRep_Init);
AckSent
Send_Receive_RIPMessages_Topology

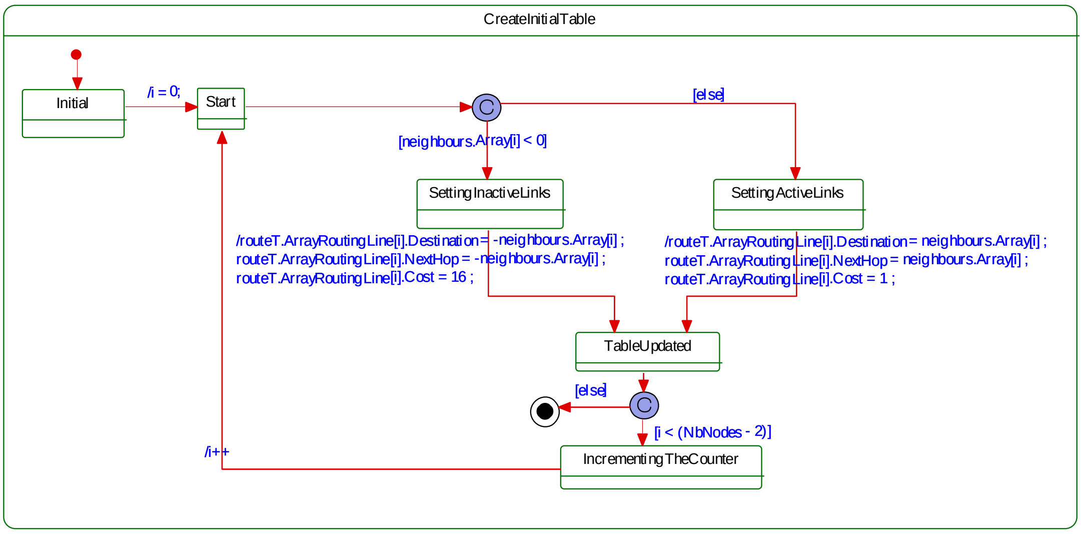
Sub-Statechart information for State TableUpdate


TableUpdate
/k = 1;
Initial
/n_line = ripMessagesReceived.ArrayNumberOfLines[k -1] ;
received_rip = ripMessagesReceived.ArrayRIPMessages[k -1] ;
CheckingRIPMessage
/n = 2;
k++ ;
Processing
/n++
[else]
[k > numberOfReceivedRIPMessages]
C
[else]
CheckingNextRIPMessage
C
[n <= n_line]
[received_rip.ArrayRIPLine[0].Cost == 0]/
senderRouter = received_rip.ArrayRIPLine[0].Destination;
[else]
C
SavingSenderRouterID
[received_rip.ArrayRIPLine[n -1].Destination == routerID]
CostUpdate
CheckingNextLine

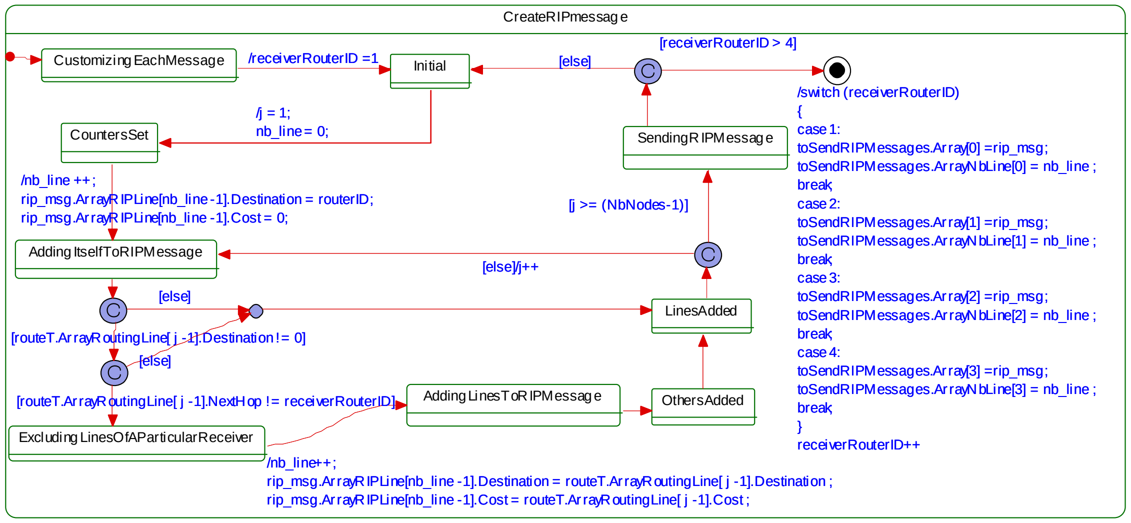
Sub-sub-Statechart information for State CostUpdate


CostUpdate
/dest = received_rip.ArrayRIPLine[n -1].Destination ;
next = senderRouter ;
cost = received_rip.ArrayRIPLine[n-1].Cost ;
m = 1;
Initial
[routeT.ArrayRouting Line[m-1].Destination == dest]
C
Modify
[else]/
m++
[else]
C
[(m +1) > NbNodes]
Add
[else]
C
[numberOfNeighbours == (NbNodes -1)]
Modify
Modifying
C
[else]
[routeT.ArrayRouting Line[m-1].NextHop == next]
[routeT.ArrayRouting Line[m-1].Cost > (cost +1)]
ModifyingTheTable
/routeT.ArrayRouting Line[m-1].NextHop = next ;
ChangingTheNextHop
C
[cost >= 15]/
routeT.ArrayRouting Line[m-1].Cost = 16 ;
[else]/
routeT.ArrayRouting Line[m-1].Cost = cost +1 ;
ModificationComplete
Add
InitializingCounter
/m =1;
Adding
C
[else]/m++
[routeT.ArrayRouting Line[m-1].Destination == 0]/
routeT.ArrayRouting Line[m-1].Destination = dest;
routeT.ArrayRouting Line[m-1].NextHop = next;
routeT.ArrayRouting Line[m-1].Cost = cost +1 ;
numberOfNeighbours++ ;
AddingToRoutingTable

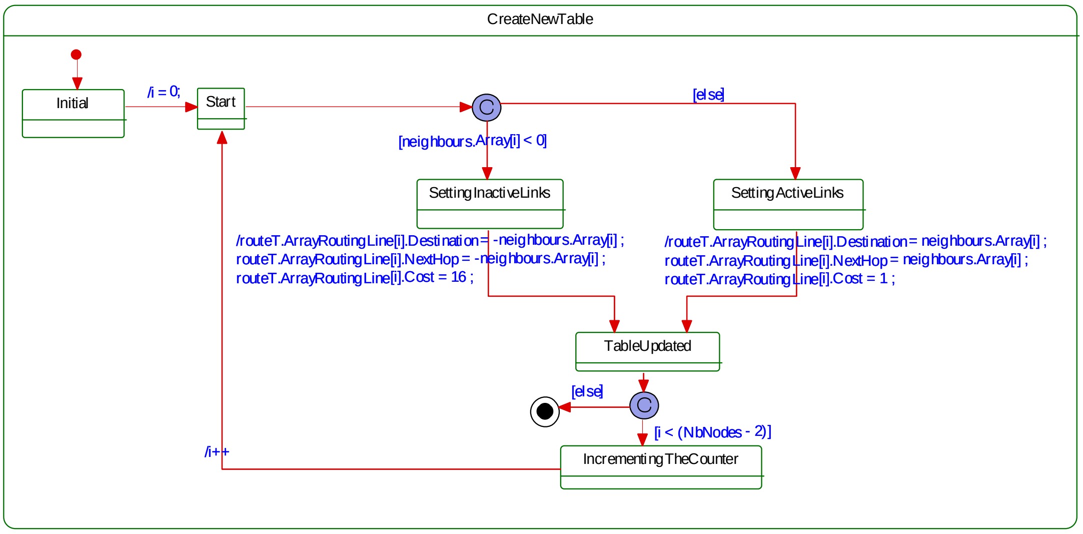
Sub-Statechart information for State CreateInitialTable


CreateInitialTable
Initial
/i = 0;
Start
C
[else]
[neighbours.Array[i] < 0]
SettingInactiveLinks
SettingActiveLinks
/routeT.ArrayRoutingLine[i].Destination= -neighbours.Array[i] ;
routeT.ArrayRoutingLine[i].NextHop = -neighbours.Array[i] ;
routeT.ArrayRoutingLine[i].Cost = 16 ;
/routeT.ArrayRoutingLine[i].Destination= neighbours.Array[i] ;
routeT.ArrayRoutingLine[i].NextHop= neighbours.Array[i] ;
routeT.ArrayRoutingLine[i].Cost = 1 ;
TableUpdated
[else]
C
[i < (NbNodes - 2)]
/i++
IncrementingTheCounter

Sub-Statechart information for State CreateRIPmessage


CreateRIPmessage
CustomizingEachMessage
/receiverRouterID =1
Initial
[else]
[receiverRouterID > 4]
C
/switch (receiverRouterID)
{
case 1:
toSendRIPMessages.Array[0] =rip_msg;
toSendRIPMessages.ArrayNbLine[0] = nb_line ;
break;
case 2:
toSendRIPMessages.Array[1] =rip_msg;
toSendRIPMessages.ArrayNbLine[1] = nb_line ;
break;
case 3:
toSendRIPMessages.Array[2] =rip_msg;
toSendRIPMessages.ArrayNbLine[2] = nb_line ;
break;
case 4:
toSendRIPMessages.Array[3] =rip_msg;
toSendRIPMessages.ArrayNbLine[3] = nb_line ;
break;
}
receiverRouterID++
CountersSet
/j = 1;
nb_line= 0;
SendingRIPMessage
/nb_line ++;
rip_msg.ArrayRIPLine[nb_line-1].Destination = routerID;
rip_msg.ArrayRIPLine[nb_line-1].Cost= 0;
[j >= (NbNodes-1)]
AddingItselfToRIPMessage
C
[else]/j++
[else]
C
LinesAdded
[routeT.ArrayRoutingLine[j -1].Destination != 0]
[else]
C
[routeT.ArrayRoutingLine[j -1].NextHop != receiverRouterID]
AddingLinesToRIPMessage
OthersAdded
ExcludingLinesOfAParticularReceiver
/nb_line++;
rip_msg.ArrayRIPLine[nb_line -1].Destination = routeT.ArrayRoutingLine[j -1].Destination ;
rip_msg.ArrayRIPLine[nb_line -1].Cost = routeT.ArrayRoutingLine[j -1].Cost ;

Sub-Statechart information for State CreateNewTable


CreateNewTable
Initial
/i = 0;
Start
C
[else]
[neighbours.Array[i] < 0]
SettingInactiveLinks
SettingActiveLinks
/routeT.ArrayRoutingLine[i].Destination= -neighbours.Array[i] ;
routeT.ArrayRoutingLine[i].NextHop = -neighbours.Array[i] ;
routeT.ArrayRoutingLine[i].Cost = 16 ;
/routeT.ArrayRoutingLine[i].Destination= neighbours.Array[i] ;
routeT.ArrayRoutingLine[i].NextHop = neighbours.Array[i] ;
routeT.ArrayRoutingLine[i].Cost = 1 ;
TableUpdated
[else]
C
[i < (NbNodes - 2)]
/i++
IncrementingTheCounter

Operation name: send_RIPtable

Body:

```
for (int i=0 ; i<3 ; i++)
{
        switch (neighbours.Array[i])
        //Sending RIP messages through the respective ports
        {
        case 1:
                        for (int x =4 ; x > toSendRIPMessages.ArrayNbLine[0] ; x--)
                        {
                                toSendRIPMessages.Array[0].ArrayRIPLine[x-1].Destination = 88;
                                toSendRIPMessages.Array[0].ArrayRIPLine[x-1].Cost = 88 ;
                        }

                        OUT_PORT(pRIP1)->GEN(evRIP(
                        toSendRIPMessages.ArrayNbLine[0] , toSendRIPMessages.Array[0],
                        toSendRIPMessages.Array[0].ArrayRIPLine[0].Destination ,
                        toSendRIPMessages.Array[0].ArrayRIPLine[0].Cost ,
                        toSendRIPMessages.Array[0].ArrayRIPLine[1].Destination ,
                        toSendRIPMessages.Array[0].ArrayRIPLine[1].Cost  ,
                        toSendRIPMessages.Array[0].ArrayRIPLine[2].Destination ,
                        toSendRIPMessages.Array[0].ArrayRIPLine[2].Cost   ,
                        toSendRIPMessages.Array[0].ArrayRIPLine[3].Destination ,
                        toSendRIPMessages.Array[0].ArrayRIPLine[3].Cost
                        ));
                        break;
        case 2:
                        for (int x =4 ; x > toSendRIPMessages.ArrayNbLine[1] ; x--)
                        {
                                toSendRIPMessages.Array[1].ArrayRIPLine[x-1].Destination = 88;
                                toSendRIPMessages.Array[1].ArrayRIPLine[x-1].Cost = 88 ;
                        }

                        OUT_PORT(pRIP2)->GEN(evRIP(
                        toSendRIPMessages.ArrayNbLine[1] , toSendRIPMessages.Array[1],
                        toSendRIPMessages.Array[1].ArrayRIPLine[0].Destination ,
                        toSendRIPMessages.Array[1].ArrayRIPLine[0].Cost ,
                        toSendRIPMessages.Array[1].ArrayRIPLine[1].Destination ,
                        toSendRIPMessages.Array[1].ArrayRIPLine[1].Cost  ,
                        toSendRIPMessages.Array[1].ArrayRIPLine[2].Destination ,
                        toSendRIPMessages.Array[1].ArrayRIPLine[2].Cost   ,
                        toSendRIPMessages.Array[1].ArrayRIPLine[3].Destination ,
                        toSendRIPMessages.Array[1].ArrayRIPLine[3].Cost
                        ));
                        break;
        case 3:
                        for (int x =4 ; x > toSendRIPMessages.ArrayNbLine[2] ; x--)
                        {
                                toSendRIPMessages.Array[2].ArrayRIPLine[x-1].Destination = 88;
                                toSendRIPMessages.Array[2].ArrayRIPLine[x-1].Cost = 88 ;
                        }

                        OUT_PORT(pRIP3)->GEN(evRIP(
                        toSendRIPMessages.ArrayNbLine[2] , toSendRIPMessages.Array[2],
                        toSendRIPMessages.Array[2].ArrayRIPLine[0].Destination ,
                        toSendRIPMessages.Array[2].ArrayRIPLine[0].Cost ,
                        toSendRIPMessages.Array[2].ArrayRIPLine[1].Destination ,
                        toSendRIPMessages.Array[2].ArrayRIPLine[1].Cost  ,
                        toSendRIPMessages.Array[2].ArrayRIPLine[2].Destination ,
                        toSendRIPMessages.Array[2].ArrayRIPLine[2].Cost   ,
                        toSendRIPMessages.Array[2].ArrayRIPLine[3].Destination ,
                        toSendRIPMessages.Array[2].ArrayRIPLine[3].Cost
                        ));
                        break;
        case 4:
                        for (int x =4 ; x > toSendRIPMessages.ArrayNbLine[3] ; x--)
                        {
                                toSendRIPMessages.Array[3].ArrayRIPLine[x-1].Destination = 88;
                                toSendRIPMessages.Array[3].ArrayRIPLine[x-1].Cost = 88 ;
                        }

                        OUT_PORT(pRIP4)->GEN(evRIP(
                        toSendRIPMessages.ArrayNbLine[3] , toSendRIPMessages.Array[3],
                        toSendRIPMessages.Array[3].ArrayRIPLine[0].Destination ,
                        toSendRIPMessages.Array[3].ArrayRIPLine[0].Cost ,
                        toSendRIPMessages.Array[3].ArrayRIPLine[1].Destination ,
                        toSendRIPMessages.Array[3].ArrayRIPLine[1].Cost  ,
                        toSendRIPMessages.Array[3].ArrayRIPLine[2].Destination ,
                        toSendRIPMessages.Array[3].ArrayRIPLine[2].Cost   ,
                        toSendRIPMessages.Array[3].ArrayRIPLine[3].Destination ,
                        toSendRIPMessages.Array[3].ArrayRIPLine[3].Cost
                        ));
                        break;
        }
}

numberOfNeighbours = 0;
for (int i=0 ; i<3 ; i++)
        if (neighbours.Array[i] > 0)
            numberOfNeighbours++;
```

Operation name: countingNumberOfNeighbours

```
Body: numberOfNeighbours = 0;
for (int i=0 ; i<3 ; i++)
    if (neighbours.Array[i] > 0)
        numberOfNeighbours++;
```

Operation name: sendingTimerToAll

```
Body:
OUT_PORT(pRouter1)->GEN(evTimer(time));
OUT_PORT(pRouter2)->GEN(evTimer(time));
OUT_PORT(pRouter3)->GEN(evTimer(time));
OUT_PORT(pRouter4)->GEN(evTimer(time));
```