LULEÅ UNIVERITY OF TECHNOLOGY

COMPUTER SCIENCE

D0021E

NODE MOBILITY

Author
S. JONSSON
setjon-7@student.ltu.se
M. LARSSON ANDERSSON
magany-7@student.ltu.se
M. JONSSON
micjon-5@student.ltu.se

Supervisor
A. Prof. KARAN MITRA
SAGUNA SAGUNA

March 20, 2020

1 Introduction

This lab teaches us the principles of the mobility of nodes both in concepts and how to apply it practically to the simulator by moving them between different router interfaces.

2 Methods

The way we set up our simulator to test the handover process we constructed a new type of event, Migrate, that once a node had sent a certain amounts of packets were sent from the node. Once the router on the other end of the link would receive a event of the type Migrate it would attempt to create a new link connected to the source of the event from on a given interface and remove the old. Incase of a interface that was already in use it would keep the old link intact and not do any changes. Once it had attempted to change interface to the given interface, it attached to the event if it was successful or not in changing to the new interface and send back the event to its source.

```java
package Sim;
// This class implements a simple router

public class Router extends SimEnt{
  //redacted variable declarations for shorter code block
  //redacted constructor for shorter code block

  // This method connects links to the router and also informs the
  // router of the host connects to the other end of the link

  public void connectInterface(int interfaceNumber, SimEnt link, SimEnt node)
  {
    if (interfaceNumber<_interfaces && _routingTable[interfaceNumber] == null)
    {
      _routingTable[interfaceNumber] = new RouteTableEntry(link, node);
    }
    else
      System.out.println("Trying to connect to port not in router");

    ((Link) link).setConnector(this);
  }

  /**
   * Return if could change interface, @param node, @param newInterface, @return
   */
  private boolean moveInterface(SimEnt node, int newInterface) {
    if(_routingTable[newInterface] != null) {
      return false;
    }
    Link link = (Link)removeFromInterface(node);
    Link newLink = new Link();
    newLink.setConnector(link._connectorA);
    newLink.setConnector(link._connectorB);
    link._connectorA = null;
    link._connectorB = null;
    ((Node)node).setPeer(newLink);
    connectInterface(newInterface, newLink , node);
    return true;
  }

  /**
   * Disconnects from link
   * @param node which is removed
   */
  private SimEnt removeFromInterface(SimEnt node) {
    SimEnt link = null;
    for(int i = 0; i<_routingTable.length; i++)
    {
      if(_routingTable[i] != null)
      if(_routingTable[i].node() == node)
      {
        link = _routingTable[i].link();
        _routingTable[i] = null;

        ((Link) link).removeConnector(this);
      }

    }
    return link;
  }

  // This method searches for an entry in the routing table that matches
  // the network number in the destination field of a messages. The link
  // represents that network number is returned

  private SimEnt getInterface(int networkAddress)
  {
    SimEnt routerInterface=null;
    for(int i=0; i<_interfaces; i++)
      if (_routingTable[i] != null)
      {
        if (((Node) _routingTable[i].node()).getAddr().networkId() == networkAddress)
        {
          routerInterface = _routingTable[i].link();
        }
      }
    return routerInterface;
  }


  // When messages are received at the router this method is called

  public void recv(SimEnt source, Event event)
  {
    if(event instanceof Migrate)
    {
      System.out.println("Router attempts to change interface for node " +((Migrate)
            event).source().getAddr().networkId() + " to  interface " +((Migrate)
            event).newInterface());
      ((Migrate) event).isSuccess(moveInterface(((Migrate) event).source(),((Migrate)
            event).newInterface()));
      send(getInterface(((Migrate) event).source().getAddr().networkId()),event,0);
    }
  }
```

```java
package Sim;

// This class implements a node (host) it has an address, a peer that it communicates with
// and it count messages send and received.

public class Node extends SimEnt {
  //redacted variable declarations for shorter code block

  public Node (int network, int node)
  {
    super();
    _id = new NetworkAddr(network, node);
  }


  // Sets the peer to communicate with. This node is single homed

  public void setPeer (SimEnt peer)
  {
    _peer = peer;

    if(_peer instanceof Link )
    {
        ((Link) _peer).setConnector(this);
    }
  }


  public NetworkAddr getAddr()
  {
    return _id;
  }

//*******************************************************************************
  // Just implemented to generate some traffic for demo.
  // In one of the labs you will create some traffic generators


  //redacted startSending variable and function declaration for shorter code block

  private int swapInterfaceAfter = 0;
  private int swapTo;
  /**
   * After non zero number of messages it will attempt swap to given interface
   * @param numberOfMessages
   * @param swapToInterface
   */
  public void changeInterfaceAfter(int numberOfMessages, int swapToInterface) {
    swapInterfaceAfter= numberOfMessages;
    swapTo = swapToInterface;
  }


//*******************************************************************************

  // This method is called upon that an event destined for this node triggers.
  public void recv(SimEnt src, Event ev)
  {
    if (ev instanceof TimerEvent)
    {
      if (_stopSendingAfter > _sentmsg && (_sentmsg != swapInterfaceAfter||swapInterfaceAfter ==
          0))
      {
        _sentmsg++;
        send(_peer, new Message(_id, new NetworkAddr(_toNetwork, _toHost),_seq),0);
        send(this, new TimerEvent(),_timeBetweenSending);
        System.out.println("Node "+_id.networkId()+ "." + _id.nodeId() +" sent message with seq:
            "+_seq + " at time "+SimEngine.getTime());
        _seq++;
      }
      else if(_sentmsg == swapInterfaceAfter)
      {
        _sentmsg++;
        System.out.println("Node "+_id.networkId()+ "." + _id.nodeId() + " sends a request to
            change interface to interface " + swapTo);
        send(_peer, new Migrate(this, swapTo),0);
        send(this, new TimerEvent(), _timeBetweenSending);

      }
    }
    if (ev instanceof Message)
    {
      System.out.println("Node "+_id.networkId()+ "." + _id.nodeId() +" receives message with seq:
          "+((Message) ev).seq() + " at time "+SimEngine.getTime());
    }
    if(ev instanceof Migrate)
    {
      System.out.println("Node "+_id.networkId()+ "." + _id.nodeId()+ " moved to new interface: "
          + ((Migrate)ev).success());
    }
  }
}
```

```java
package Sim;
public class Migrate implements Event{
  private Node _source;
  private int _newInterface;
  private boolean _success = false;
  public Migrate (Node from, int newInterface)
  {
    _source = from;
    _newInterface = newInterface;
  }
  /// set if migrate was success or not
  public void isSuccess(boolean success) {
    _success = success;
  }
  ///returns the new wished interface
  public int newInterface()
  {
    return _newInterface;
  }
  /// returns the source node of the migrate request
  public Node source()
  {
    return _source;
  }
  /// returns if the migrate process was successful or not
  public boolean success() {
    return _success;
  }

  public void entering(SimEnt locale)
  {
  }
}
```

Figure 3: migrate

3 Results

```
Console
<terminated> Run [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (21 feb. 2020 13:10:44)
Node 1.1 sent message with seq: 30 at time 10.0
Node 2.1 sends a request to change interface to interface 2
Link recv msg, passes it through
Link recv msg, passes it through
Router handles packet with seq: 30 from node: 1.1
Router sends to node: 2.2
Router attempts to change interface for node 2 to  interface 2
Link dropped packet
Link recv msg, passes it through
Node 2.1 moved to new interface: true
Node 1.1 sends a request to change interface to interface 2
Node 2.1 sent message with seq: 10 at time 11.0
Link recv msg, passes it through
Link recv msg, passes it through
Router attempts to change interface for node 1 to  interface 2
Router handles packet with seq: 10 from node: 2.1
Router sends to node: 1.1
Link recv msg, passes it through
Link recv msg, passes it through
Node 1.1 moved to new interface: false
Node 1.1 receives message with seq: 10 at time 11.0
```

Figure 4: Result of run with two nodes trying to change to interface

Result of trying to make two nodes on same router change interface can be seen in figure 4, red underlines show the messages of the attempts. As figure show only the first node were allowed to change interface as when the other attempted the interface was already taken. Underlined in green is old link dropping a packet due to no connection. By making sure that handover process happened before router gets any messages no packets would be drop, see figure 5.

```
Console
<terminated> Run [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (21 feb. 2020 13:41:42)
Node 2.1 sends a request to change interface to interface 2
Node 1.1 sent message with seq: 30 at time 10.0
Link recv msg, passes it through
Link recv msg, passes it through
Router attempts to change interface for node 2 to  interface 2
Router handles packet with seq: 30 from node: 1.1
Router sends to node: 2.2
Link recv msg, passes it through
Link recv msg, passes it through
Node 2.1 moved to new interface: true
Node 2.1 receives message with seq: 30 at time 10.0
```

Figure 5: Run with handover before message

4 Discussion

Our solution for the handover proccess is simple and instant. This is easy to observer when you compare the to figures 4 and 5. In figure 4 a packet is dropped while in figure 5 the packet is not dropped. In reality a handover can take seconds to perform so our simulation in the time aspect is not a good representation of that aspect. However as we can see in figure 4 we drop a packet that is sent on a dead link which do happen in reality and is good example of showing that even an instantly setting up a new link on a new interface is not enough to protect a node from packet loss.