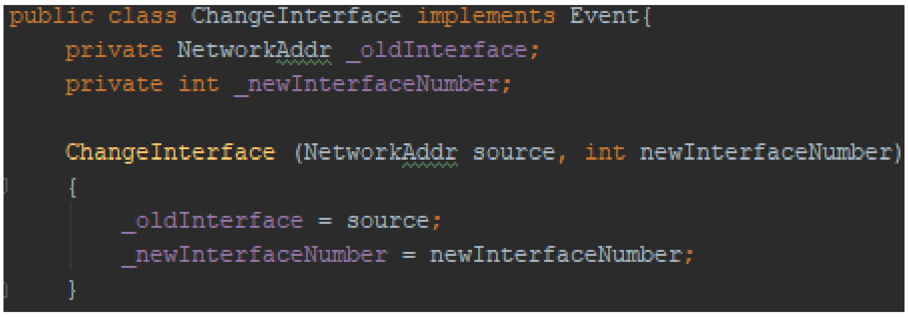
Laboratory 3

Introduction

With functionality implemented in this lab, a node can move between interfaces on one router without losing connection or dropping packets during the switch.

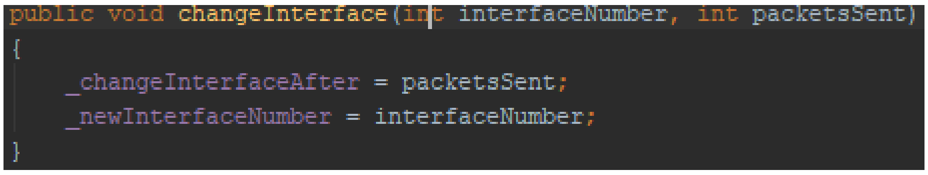
Method

To change interface of a node a new event called ChangeInterface is sent from a node to the router. This event contains the NetworkAddr for the current interface and the new interface number to change to.



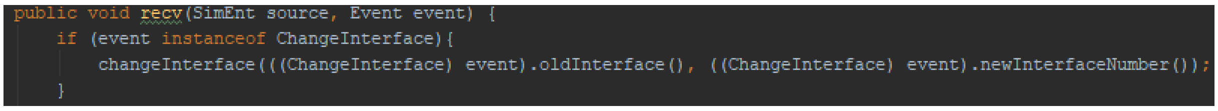
```java
public class ChangeInterface implements Event{
    private NetworkAddr _oldInterface;
    private int _newInterfaceNumber;

    ChangeInterface (NetworkAddr source, int newInterfaceNumber)
    {
        _oldInterface = source;
        _newInterfaceNumber = newInterfaceNumber;
    }
```

To request an interface change you can either use the nodes send function and pass a ChangeInterface or call for a new function in Node which will make the node change interface after a specified number of messages is sent from that node.

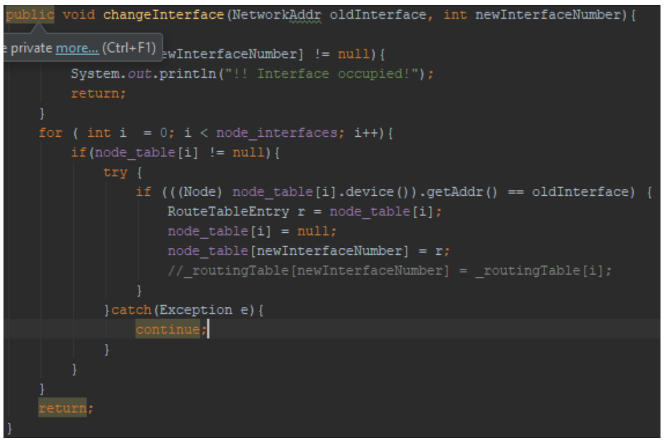


```java
public void changeInterface(int interfaceNumber, int packetsSent)
{
    _changeInterfaceAfter = packetsSent;
    _newInterfaceNumber = interfaceNumber;
}
```

This is then picked up by the Routers recv function which passes it onto a method that loops trough all the interfaces until it finds the old/current one where it sets it to null and instead assigns it to the new. If the new interface is already occupied, nothing happens.



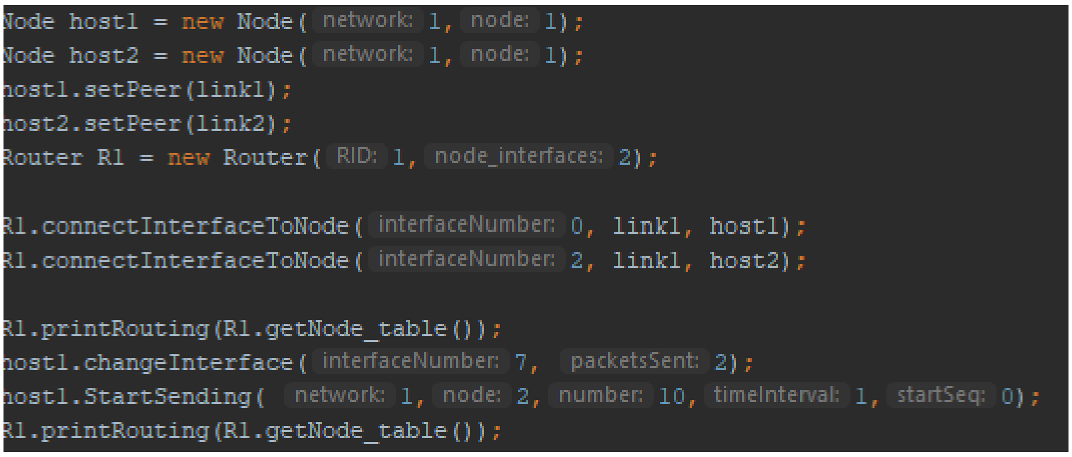
```java
public void recv(SimEnt source, Event event) {
    if (event instanceof ChangeInterface){
        changeInterface(((ChangeInterface) event).oldInterface(), ((ChangeInterface) event).newInterfaceNumber());
    }
```



```
public void changeInterface(NetworkAddr oldInterface, int newInterfaceNumber){
: private more... (Ctrl+F1) ewInterfaceNumber] != null){
        System.out.println("!! Interface occupied!");
        return;
    }
    for ( int i  = 0; i < node_interfaces; i++){
        if(node_table[i] != null){
            try {
                if (((Node) node_table[i].device()).getAddr() == oldInterface) {
                    RouteTableEntry r = node_table[i];
                    node_table[i] = null;
                    node_table[newInterfaceNumber] = r;
                    //_routingTable[newInterfaceNumber] = _routingTable[i];
                }
            }catch(Exception e){
                continue;
            }
        }
    }
    return;
}
```

Results

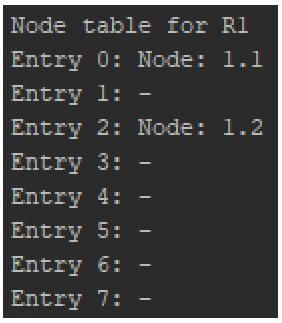
To test, create one (or more) router and connect two nodes to it. Set up a change interface after some number of messages then start sending.



```
Node host1 = new Node( network: 1, node: 1);
Node host2 = new Node( network: 1, node: 1);
host1.setPeer(link1);
host2.setPeer(link2);
Router R1 = new Router( RID: 1, node_interfaces: 2);

R1.connectInterfaceToNode( interfaceNumber: 0, link1, host1);
R1.connectInterfaceToNode( interfaceNumber: 2, link1, host2);

R1.printRouting(R1.getNode_table());
host1.changeInterface( interfaceNumber: 7, packetsSent: 2);
host1.StartSending( network: 1, node: 2, number: 10, timeInterval: 1, startSeq: 0);
R1.printRouting(R1.getNode_table());
```

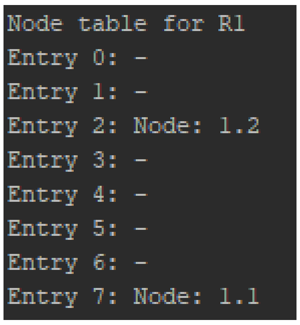
The results can then be showed by printing the interfaces of the router.

Before:



```
Node table for R1
Entry 0: Node: 1.1
Entry 1: -
Entry 2: Node: 1.2
Entry 3: -
Entry 4: -
Entry 5: -
Entry 6: -
Entry 7: -
```

After:



```
Node table for R1
Entry 0: -
Entry 1: -
Entry 2: Node: 1.2
Entry 3: -
Entry 4: -
Entry 5: -
Entry 6: -
Entry 7: Node: 1.1
```

As can be seen, the node has moved from entry 0 to 7.