CS2031 - Telecommunications II

Assignment 2 Report

Routing

Introduction

Throughout this report I will discuss and explain the design and implementation of the routing methods as described in CS2031 Assignment 2. I will initially explain the design and implementation of the routing protocol. I will also explore the various obstacles and issues that I encountered throughout the assignment as well as various advantages and disadvantages associated with the approaches I decided to take.

Design and Implementation

On the overall, the program I designed is composed of various objects:

- Controller
- End Users
- Routers

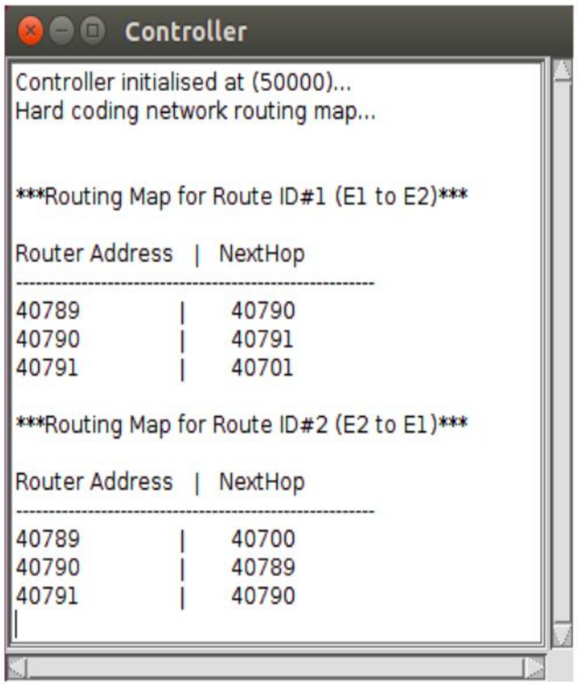
Within these classes I have designed and implemented various methods and protocols in order to achieve the desired output of having a network of end users and routers that are all interconnected. This is done with the desired outcome of an end user (node) being able to communicate with another end user (node) in the most efficient way possible, i.e via the shortest route. In ***Part One - Preconfigured Routing*** the controller has a hardcoded routing table for the network. In ***Part Two - Link State Routing*** the controller builds a routing table for the network using a modification of Dijkstra's algorithm to find the shortest route between all end users in the network.

Part One - Preconfigured Routing

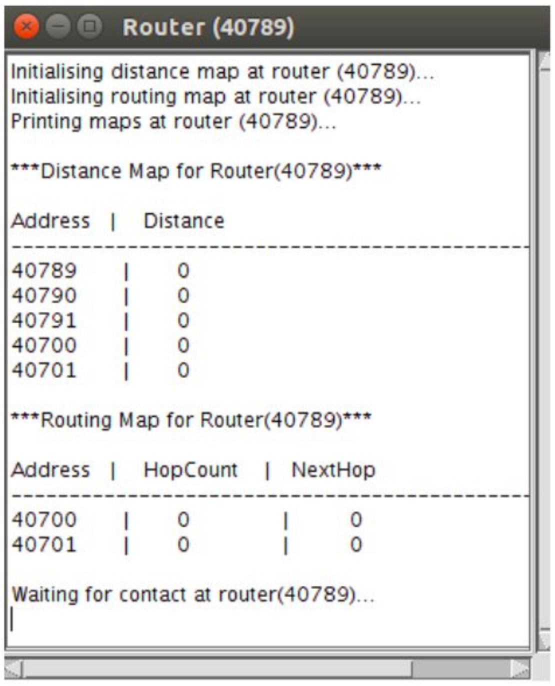
In this section, the **controller** is preconfigured (hardcoded) with information regarding how the network is connected. It has full knowledge of how node X should get to node Y. When the controller is established on a port on the network address (*localhost*) it begins by designing a routing table for the network. This is achieved using a nested *HashMap* data structure as follows:

HashMap<RouteID#, HashMap<NodeAddress, NextHop>> routingMap

Each **router** in the network is then established at their respective port on *localhost*. When a router is created it is initialised with an empty routing table containing all of the end users in the network. Routers use a **RoutingElementKey** object nested within a hashmap to model the routing table. Each key (node) within the HashMap (routing table) has a corresponding value (*RoutingElementKey)* and within the *RoutingElementKey* is the address of the *NextHop* for that respective key (node) and also the *HopCount* (distance in hops) to the node.

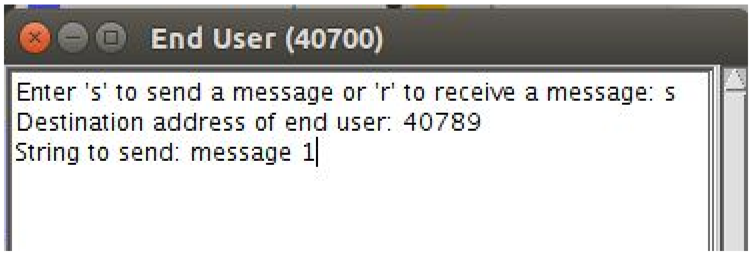


```
Controller initialised at (50000)...
Hard coding network routing map...


***Routing Map for Route ID#1 (E1 to E2)***

Router Address  |  NextHop
--------------------------------------------------
40789           |     40790
40790           |     40791
40791           |     40701

***Routing Map for Route ID#2 (E2 to E1)***

Router Address  |  NextHop
--------------------------------------------------
40789           |     40700
40790           |     40789
40791           |     40790
```



```
Initialising distance map at router (40789)...
Initialising routing map at router (40789)...
Printing maps at router (40789)...

***Distance Map for Router(40789)***

Address  |   Distance
--------------------------------------------------
40789    |     0
40790    |     0
40791    |     0
40700    |     0
40701    |     0

***Routing Map for Router(40789)***

Address  |   HopCount   |   NextHop
--------------------------------------------------
40700    |     0        |      0
40701    |     0        |      0

Waiting for contact at router(40789)...
```

Once the controller and all routers in the network have been created and initialised it is then time for the **end users** to be established. When an end user is created it knows the port of the router that it is connected to. End users are first given the option to either send/receive a message. Based on the users decision they will be placed in the respective mode. If the user selects to receive a message the node will remain idle waiting for contact at that end user's port on *localhost*. However, if the user selects to send a message they must then input the port of the destination end user to end a message to. They will then be prompted to input the text to be sent within the packet to the defined end user.

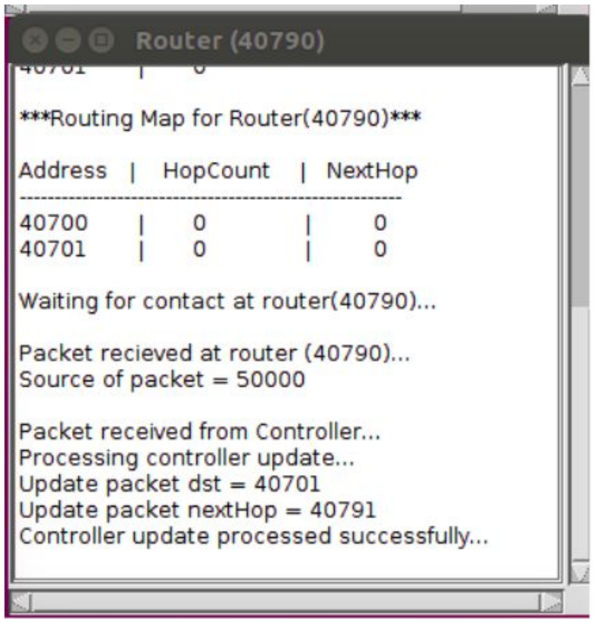


```
Enter 's' to send a message or 'r' to receive a message: s
Destination address of end user: 40789
String to send: message 1
```

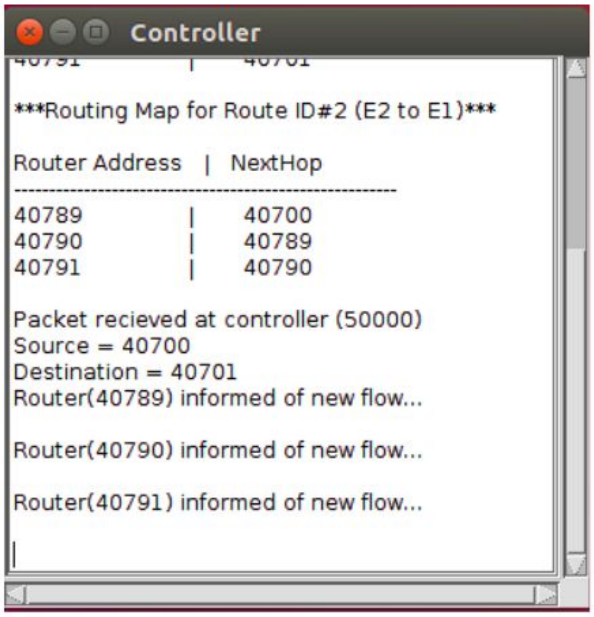
All of the data entered above is then encapsulated into a ***Packet*** which is to be sent to another end user via routers in the network.

This packet is then sent to the connected router for the given end user node using Datagram Packets and Sockets. When a packet is received at a router it first extracts the DST (destination) address from the packet, this is then cross referenced with the local *RoutingTable* for the router to check if it has knowledge of how to get to DST. If the router **does not** have knowledge of how to get to DST it then sends an **Update Request Packet** to the controller seeking information of how it should get to DST.

When the controller receives a packet it knows that it must be from a router seeking an update i.e a ***Update Request Packet***. It handles this accordingly and extracts the DST and SRC_NODE, it then cross-references this with its local *RoutingTable* and following this informs all nodes affected in the transmission route of their *NextHop* address when a packet is received that is addressed for DST. This is done using ***Update Response Packets***.



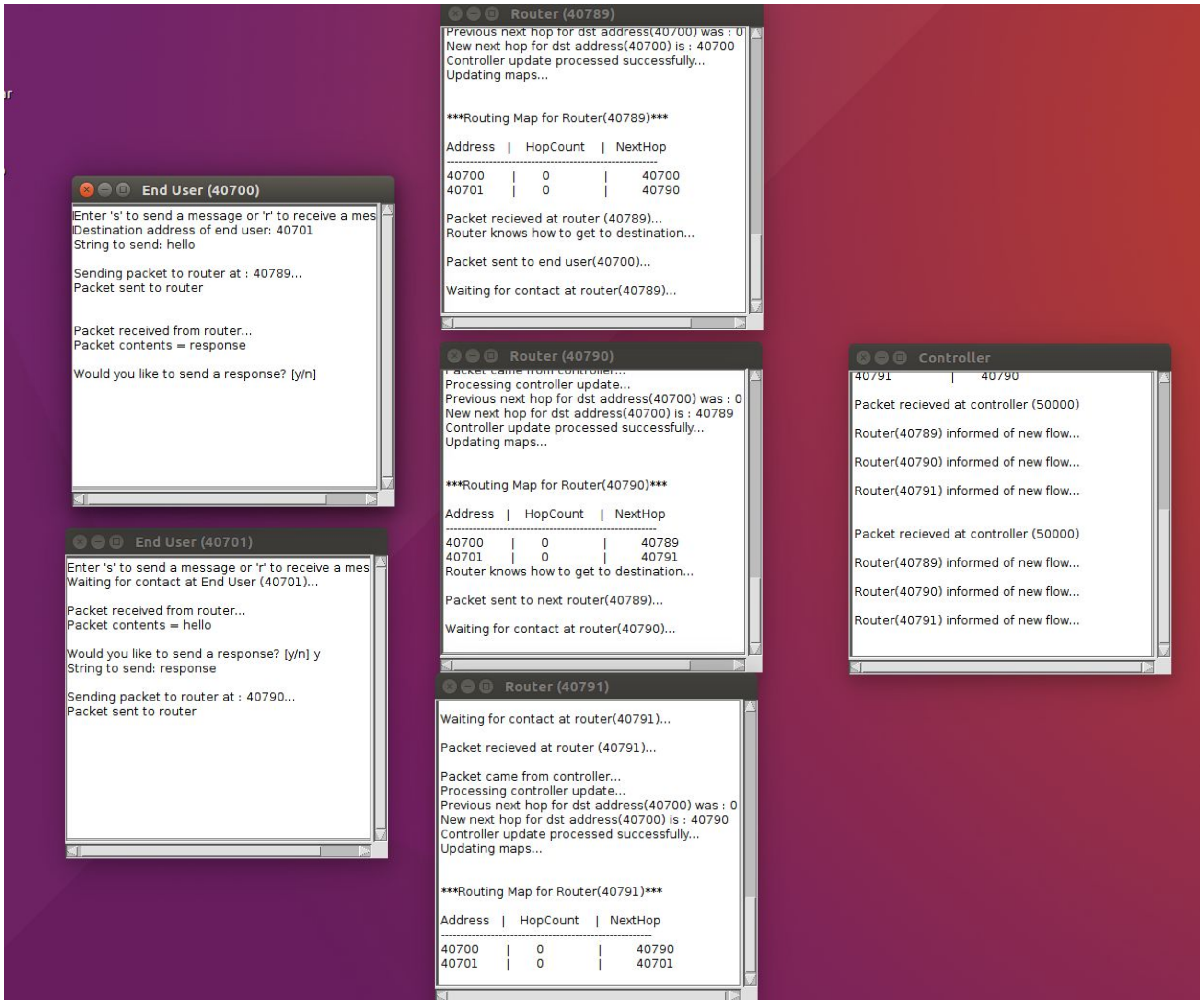
```
***Routing Map for Router(40790)***

Address  |  HopCount  |  NextHop
------------------------------------------------
40700    |     0      |     0
40701    |     0      |     0

Waiting for contact at router(40790)...

Packet recieved at router (40790)...
Source of packet = 50000

Packet received from Controller...
Processing controller update...
Update packet dst = 40701
Update packet nextHop = 40791
Controller update processed successfully...
```



```
***Routing Map for Route ID#2 (E2 to E1)***

Router Address  |  NextHop
------------------------------------------------
40789           |    40700
40790           |    40789
40791           |    40790

Packet recieved at controller (50000)
Source = 40700
Destination = 40701
Router(40789) informed of new flow...

Router(40790) informed of new flow...

Router(40791) informed of new flow...
```

This way, all routers retain the knowledge of how a packet should be directed from A to B which prevents the controller needing to be contacted every hop of the transmission. Once a packet has been sent once between all end users the routing table for the network is then fully populated.

If on the other hand, a packet is received at a router and it already has knowledge of how to get to DST, the packet is then forwarded to the NEXT_HOP address taken from the local routing table at each router respectively all the way until the packet reaches the DST address.



```
End User (40700)
Enter 's' to send a message or 'r' to receive a mes
Destination address of end user: 40701
String to send: hello

Sending packet to router at : 40789...
Packet sent to router


Packet received from router...
Packet contents = response

Would you like to send a response? [y/n]
```

```
End User (40701)
Enter 's' to send a message or 'r' to receive a mes
Waiting for contact at End User (40701)...

Packet received from router...
Packet contents = hello

Would you like to send a response? [y/n] y
String to send: response

Sending packet to router at : 40790...
Packet sent to router
```

```
Router (40789)
Previous next hop for dst address(40700) was : 0
New next hop for dst address(40700) is : 40700
Controller update processed successfully...
Updating maps...


***Routing Map for Router(40789)***

Address  |   HopCount   |  NextHop
-------------------------------------------------
40700      |      0             |       40700
40701      |      0             |       40790

Packet recieved at router (40789)...
Router knows how to get to destination...

Packet sent to end user(40700)...

Waiting for contact at router(40789)...
```

```
Router (40790)
Packet came from controller...
Processing controller update...
Previous next hop for dst address(40700) was : 0
New next hop for dst address(40700) is : 40789
Controller update processed successfully...
Updating maps...


***Routing Map for Router(40790)***

Address  |   HopCount   |  NextHop
-------------------------------------------------
40700      |      0             |       40789
40701      |      0             |       40791
Router knows how to get to destination...

Packet sent to next router(40789)...

Waiting for contact at router(40790)...
```

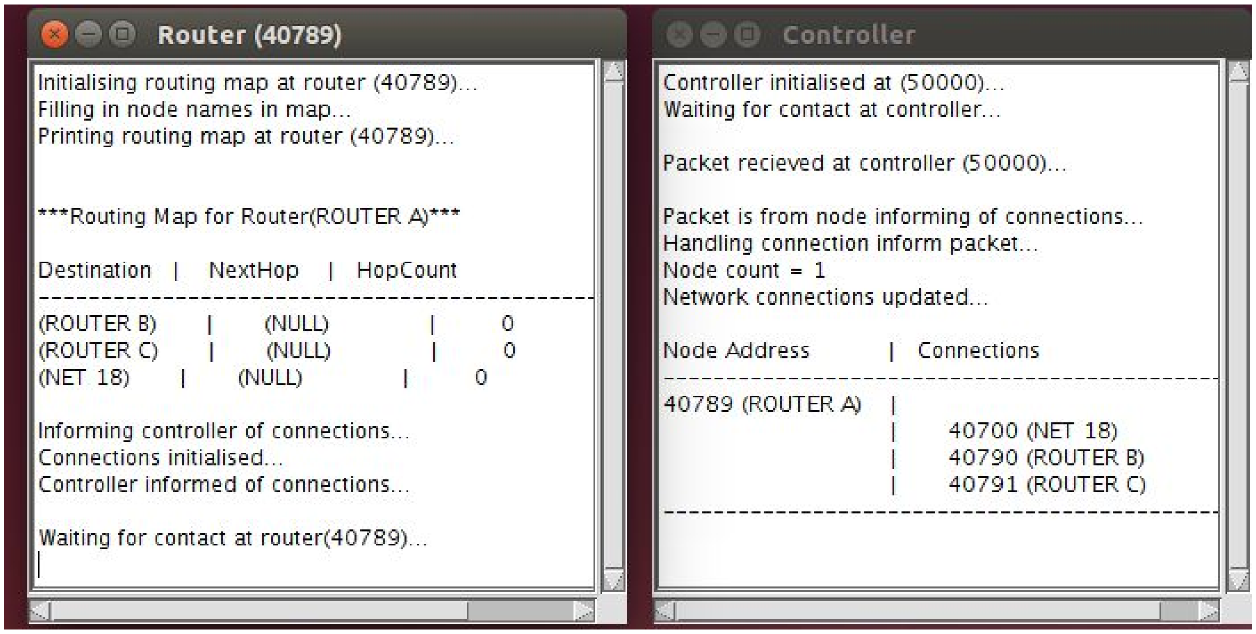
```
Router (40791)

Waiting for contact at router(40791)...

Packet recieved at router (40791)...

Packet came from controller...
Processing controller update...
Previous next hop for dst address(40700) was : 0
New next hop for dst address(40700) is : 40790
Controller update processed successfully...
Updating maps...


***Routing Map for Router(40791)***

Address  |   HopCount   |  NextHop
-------------------------------------------------
40700      |      0             |       40790
40701      |      0             |       40701
```

```
Controller
40791          |      40790

Packet recieved at controller (50000)

Router(40789) informed of new flow...

Router(40790) informed of new flow...

Router(40791) informed of new flow...


Packet recieved at controller (50000)

Router(40789) informed of new flow...

Router(40790) informed of new flow...

Router(40791) informed of new flow...
```

Part Two - Link State Routing

First, the controller is initialized and it then waits to receive communication from nodes within the network. The controller has knowledge of how many nodes are within the network (*NETWORK_NODE_COUNT*).

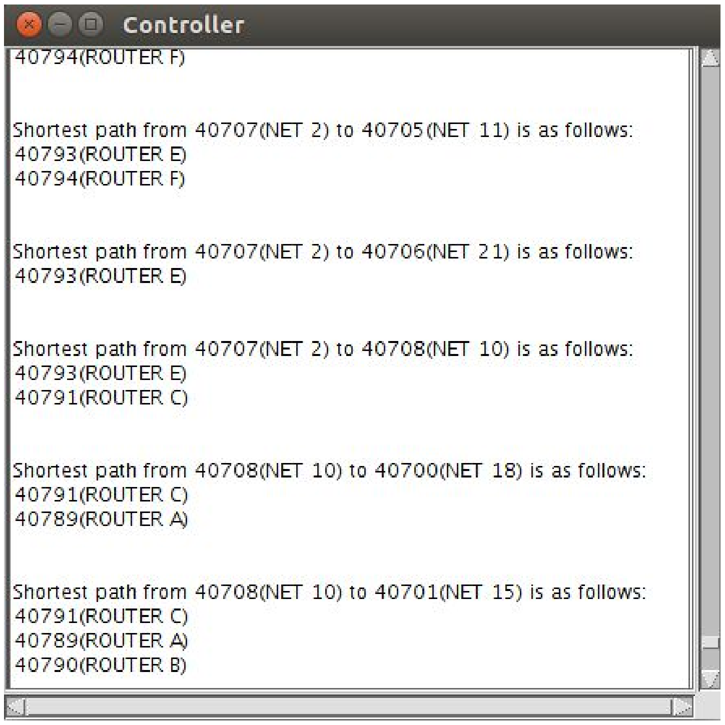
Then, each router and end user node in the network is created at its respective port on the network address (*localhost*). When a node is created it sends a ***Controller Inform Packet*** to the controller which informs it of all the local connections for the node. The controller processes the inform packet and starts to build a network map for all connections.



```
Initialising routing map at router (40789)...
Filling in node names in map...
Printing routing map at router (40789)...


***Routing Map for Router(ROUTER A)***

Destination   |    NextHop    |   HopCount
---------------------------------------------
(ROUTER B)     |      (NULL)          |       0
(ROUTER C)     |      (NULL)          |       0
(NET 18)     |      (NULL)          |       0

Informing controller of connections...
Connections initialised...
Controller informed of connections...

Waiting for contact at router(40789)...
```

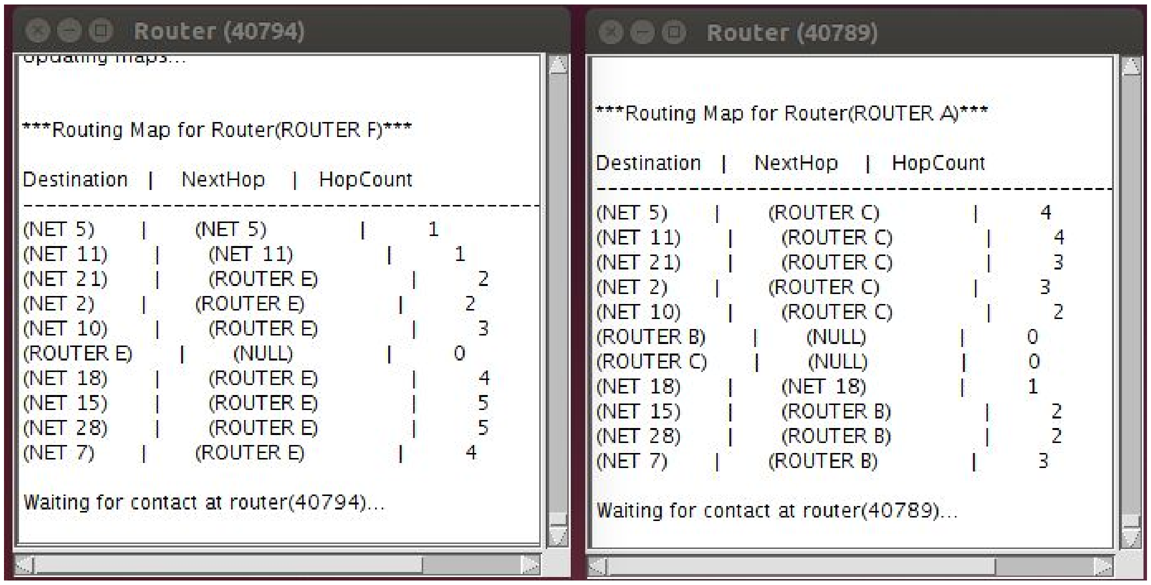
```
Controller initialised at (50000)...
Waiting for contact at controller...

Packet recieved at controller (50000)...

Packet is from node informing of connections...
Handling connection inform packet...
Node count = 1
Network connections updated...

Node Address          |   Connections
---------------------------------------------
40789 (ROUTER A)   |
                   |      40700 (NET 18)
                   |      40790 (ROUTER B)
                   |      40791 (ROUTER C)
---------------------------------------------
```

Once the controller has received an information packet from each node in the network it begins discovering the quickest route between each end user node in the network. This is done by implementing a version of *Link State Routing* using a *Breadth First Search* algorithm to find the shortest route between each end user node. This is pretty much using ***Dijkstra's*** algorithm without having weighted distances between each node.



```
40794(ROUTER F)


Shortest path from 40707(NET 2) to 40705(NET 11) is as follows:
40793(ROUTER E)
40794(ROUTER F)


Shortest path from 40707(NET 2) to 40706(NET 21) is as follows:
40793(ROUTER E)


Shortest path from 40707(NET 2) to 40708(NET 10) is as follows:
40793(ROUTER E)
40791(ROUTER C)


Shortest path from 40708(NET 10) to 40700(NET 18) is as follows:
40791(ROUTER C)
40789(ROUTER A)


Shortest path from 40708(NET 10) to 40701(NET 15) is as follows:
40791(ROUTER C)
40789(ROUTER A)
40790(ROUTER B)
```

When the controller has finished this process it then has full knowledge of the best routes between all end users in the network stored in a *Routing Table*. Once the *Routing Table* is fully populated the controller informs each affected node of their *NextHop* address for when a packet is being attempted to be sent to an end user. This is done by sending a **Node Inform Packet** to each node informing them of the appropriate details.

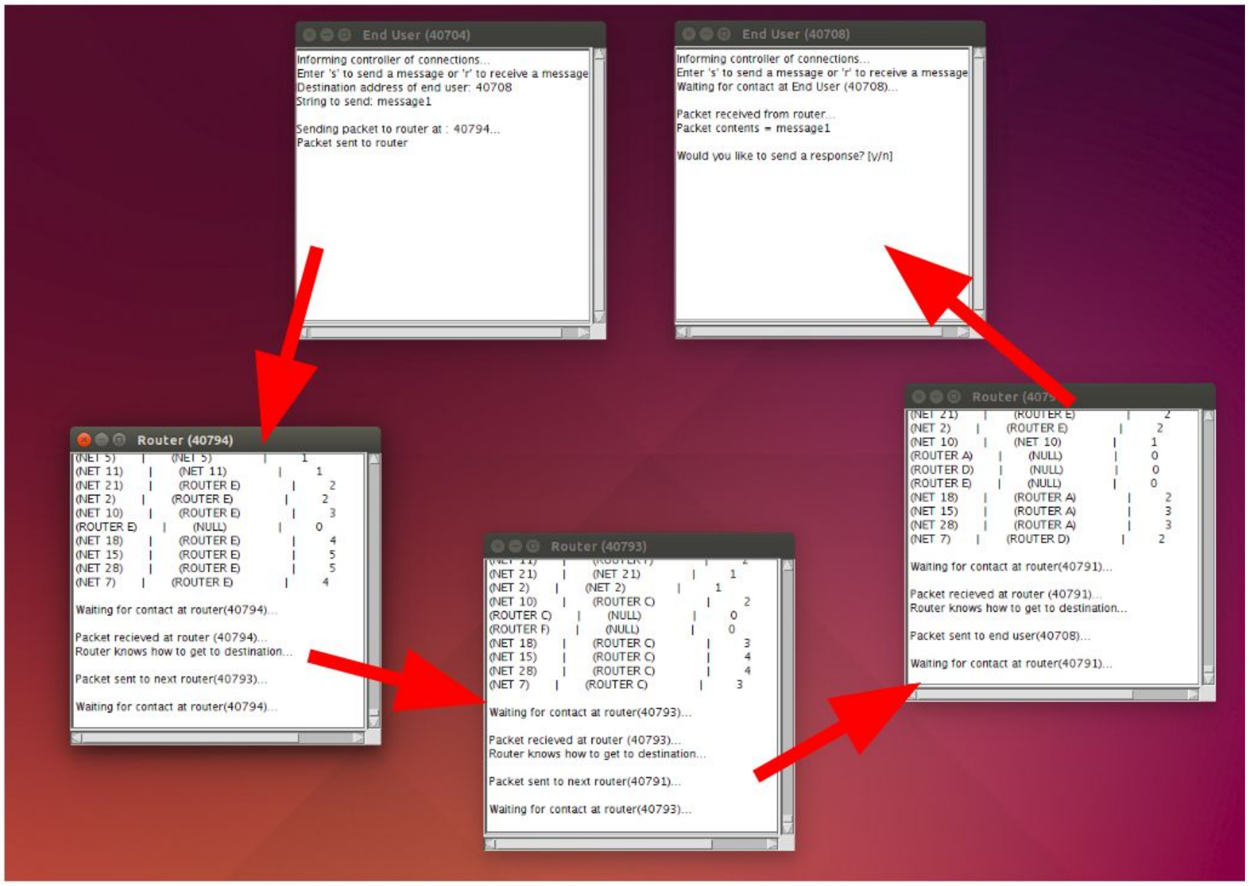


```
Updating maps...


***Routing Map for Router(ROUTER F)***

Destination  |   NextHop    |  HopCount
----------------------------------------------
(NET 5)      |   (NET 5)         |    1
(NET 11)     |   (NET 11)        |    1
(NET 21)     |   (ROUTER E)      |    2
(NET 2)      |   (ROUTER E)      |    2
(NET 10)     |   (ROUTER E)      |    3
(ROUTER E)   |   (NULL)          |    0
(NET 18)     |   (ROUTER E)      |    4
(NET 15)     |   (ROUTER E)      |    5
(NET 28)     |   (ROUTER E)      |    5
(NET 7)      |   (ROUTER E)      |    4

Waiting for contact at router(40794)...
```

```
***Routing Map for Router(ROUTER A)***

Destination  |   NextHop    |  HopCount
----------------------------------------------
(NET 5)      |   (ROUTER C)      |    4
(NET 11)     |   (ROUTER C)      |    4
(NET 21)     |   (ROUTER C)      |    3
(NET 2)      |   (ROUTER C)      |    3
(NET 10)     |   (ROUTER C)      |    2
(ROUTER B)   |   (NULL)          |    0
(ROUTER C)   |   (NULL)          |    0
(NET 18)     |   (NET 18)        |    1
(NET 15)     |   (ROUTER B)      |    2
(NET 28)     |   (ROUTER B)      |    2
(NET 7)      |   (ROUTER B)      |    3

Waiting for contact at router(40789)...
```

When the entire network has been informed of all routing flows by the controller end users are then free to receive/transmit packets between each other around the network. Packets that are sent from an end user will be first sent to the router that the end user is directly connected to. The packet is then directed by that router and all other network routers along the path of minimal cost, i.e the shortest route between ***END_USER_A*** and ***END_USER_B***.

Example Message Flow:



```
Informing controller of connections...
Enter 's' to send a message or 'r' to receive a message
Destination address of end user: 40708
String to send: message1

Sending packet to router at : 40794...
Packet sent to router
```

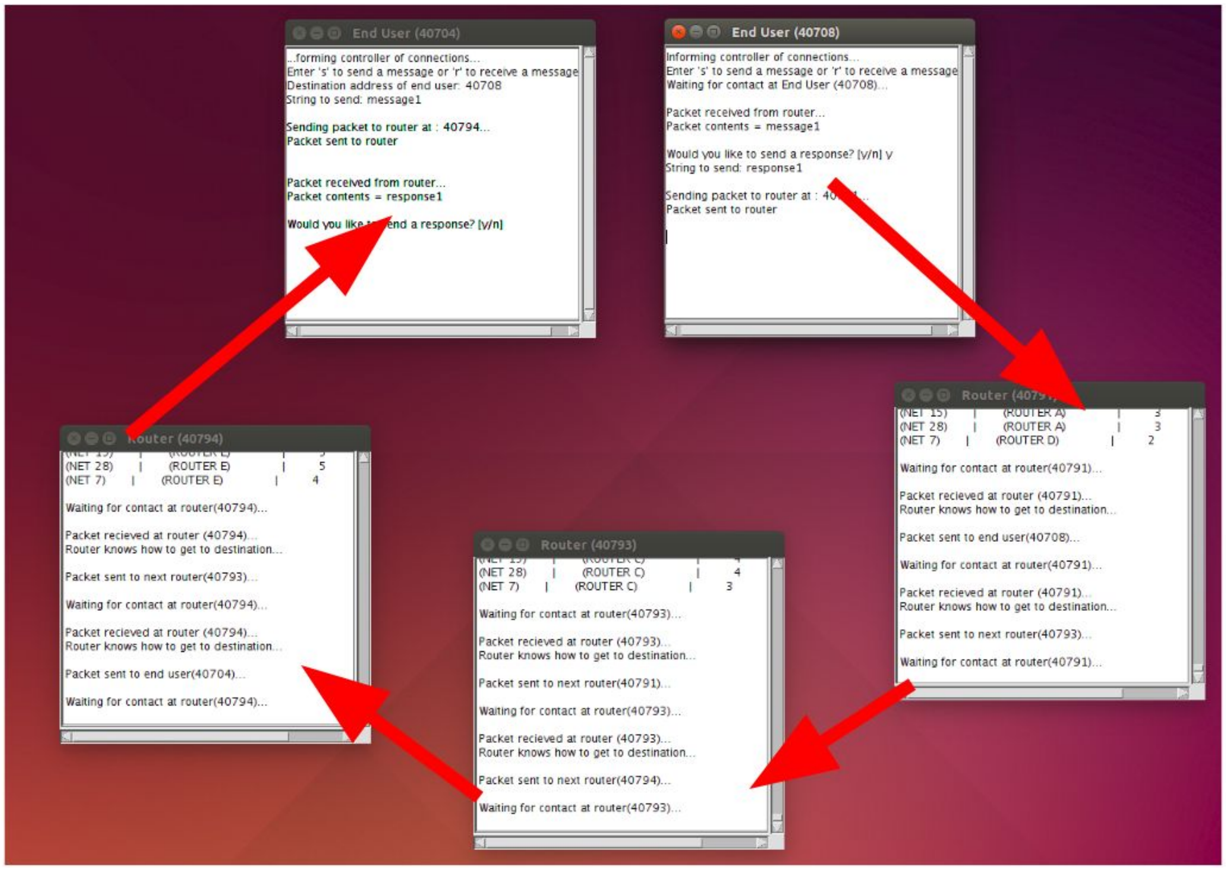
```
Informing controller of connections...
Enter 's' to send a message or 'r' to receive a message
Waiting for contact at End User (40708)...

Packet received from router...
Packet contents = message1

Would you like to send a response? [y/n]
```

```
(NET 5)     |    (NET 5)       |    1
(NET 11)    |    (NET 11)      |    1
(NET 21)    |    (ROUTER E)    |    2
(NET 2)     |    (ROUTER E)    |    2
(NET 10)    |    (ROUTER E)    |    3
(ROUTER E)  |    (NULL)        |    0
(NET 18)    |    (ROUTER E)    |    4
(NET 15)    |    (ROUTER E)    |    5
(NET 28)    |    (ROUTER E)    |    5
(NET 7)     |    (ROUTER E)    |    4

Waiting for contact at router(40794)...

Packet recieved at router (40794)...
Router knows how to get to destination...

Packet sent to next router(40793)...

Waiting for contact at router(40794)...
```

```
(NET 11)    |    (ROUTER F)    |    2
(NET 21)    |    (NET 21)      |    1
(NET 2)     |    (NET 2)       |    1
(NET 10)    |    (ROUTER C)    |    2
(ROUTER C)  |    (NULL)        |    0
(ROUTER F)  |    (NULL)        |    0
(NET 18)    |    (ROUTER C)    |    3
(NET 15)    |    (ROUTER C)    |    4
(NET 28)    |    (ROUTER C)    |    4
(NET 7)     |    (ROUTER C)    |    3

Waiting for contact at router(40793)...

Packet recieved at router (40793)...
Router knows how to get to destination...

Packet sent to next router(40791)...

Waiting for contact at router(40793)...
```

```
(NET 21)    |    (ROUTER E)    |    2
(NET 2)     |    (ROUTER E)    |    2
(NET 10)    |    (NET 10)      |    1
(ROUTER A)  |    (NULL)        |    0
(ROUTER D)  |    (NULL)        |    0
(ROUTER E)  |    (NULL)        |    0
(NET 18)    |    (ROUTER A)    |    2
(NET 15)    |    (ROUTER A)    |    3
(NET 28)    |    (ROUTER A)    |    3
(NET 7)     |    (ROUTER D)    |    2

Waiting for contact at router(40791)...

Packet recieved at router (40791)...
Router knows how to get to destination...

Packet sent to end user(40708)...

Waiting for contact at router(40791)...
```

Example Response Flow:



```
...forming controller of connections...
Enter 's' to send a message or 'r' to receive a message
Destination address of end user: 40708
String to send: message1

Sending packet to router at : 40794...
Packet sent to router


Packet received from router...
Packet contents = response1

Would you like to send a response? [y/n]
```

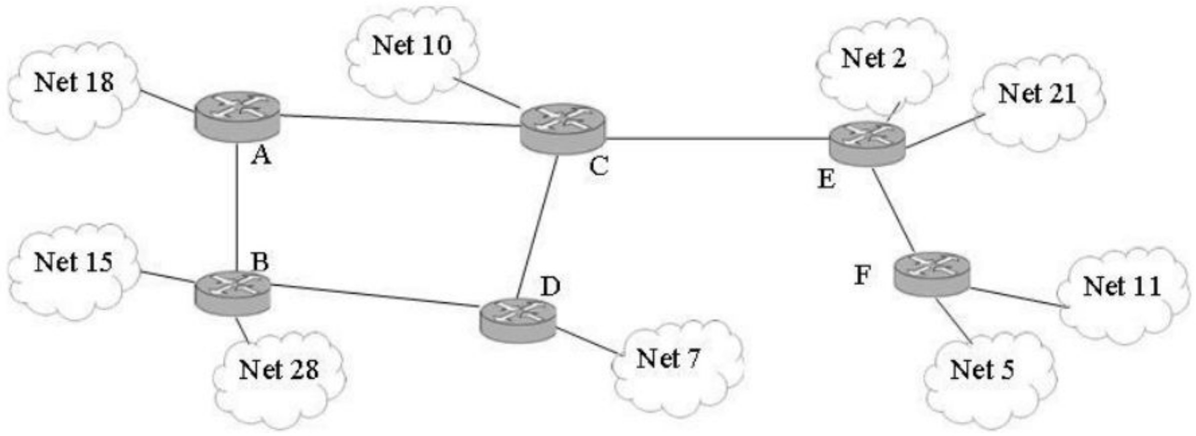
```
Informing controller of connections...
Enter 's' to send a message or 'r' to receive a message
Waiting for contact at End User (40708)...

Packet received from router...
Packet contents = message1

Would you like to send a response? [y/n] y
String to send: response1

Sending packet to router at : 40[illegible]...
Packet sent to router
```

```
(NET 15)      |      (ROUTER A)          |        3
(NET 28)      |      (ROUTER A)          |        3
(NET 7)     |      (ROUTER D)          |       2

Waiting for contact at router(40791)...

Packet recieved at router (40791)...
Router knows how to get to destination...

Packet sent to end user(40708)...

Waiting for contact at router(40791)...

Packet recieved at router (40791)...
Router knows how to get to destination...

Packet sent to next router(40793)...

Waiting for contact at router(40791)...
```

```
(NET 28)      |      (ROUTER C)          |        4
(NET 7)     |      (ROUTER C)          |       3

Waiting for contact at router(40793)...

Packet recieved at router (40793)...
Router knows how to get to destination...

Packet sent to next router(40791)...

Waiting for contact at router(40793)...

Packet recieved at router (40793)...
Router knows how to get to destination...

Packet sent to next router(40794)...

Waiting for contact at router(40793)...
```

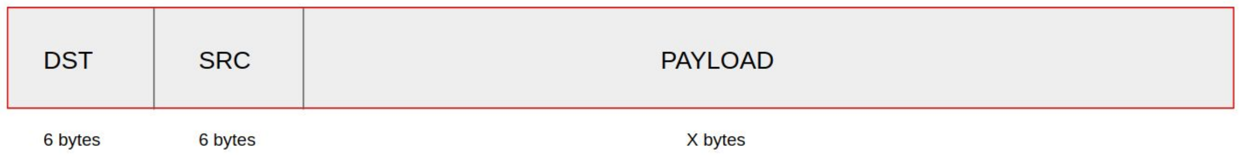
```
(NET 28)      |      (ROUTER E)          |        5
(NET 7)     |      (ROUTER E)          |       4

Waiting for contact at router(40794)...

Packet recieved at router (40794)...
Router knows how to get to destination...

Packet sent to next router(40793)...

Waiting for contact at router(40794)...

Packet recieved at router (40794)...
Router knows how to get to destination...

Packet sent to end user(40704)...

Waiting for contact at router(40794)...
```

Network Implemented:



Packet Protocol

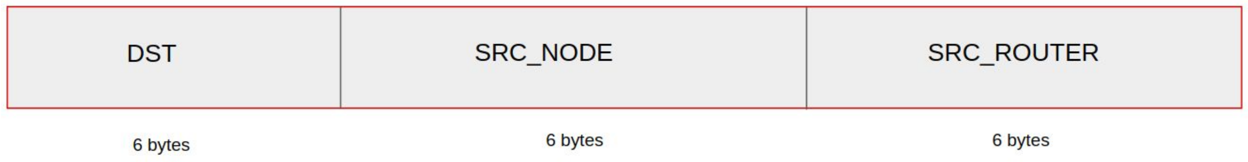
The **Packet** follows the following protocol:



- **DST -** Destination address of the packet (e.g END_USER_2)
- **SRC -** Source address of the packet (e.g END_USER_1)
- **PAYLOAD -** Data to be transmitted from SRC to DST

Update Request Packet Protocol

The **Update Request Packet** follows the following protocol:



DST - Destination address of the packet (e.g END_USER_2)

SRC_NODE - Source address of the packet (e.g END_USER_1)

SRC_ROUTER - Address of router seeking knowledge

Update Response Packet Protocol

The ***Update Response Packet*** follows the following protocol:

DST	NODE_ADDR	NEXT_HOP
6 bytes	6 bytes	6 bytes

DST - End destination address of the initial packet (e.g END_USER_2)

NODE_ADDR - Address of the node to be updated (e.g ROUTER_4)

NEXT_HOP - Next hop for NODE_ADDR (e.g ROUTER_5)

Controller Inform Packet Protocol

The ***Controller Inform Packet*** follows the following protocol:

FLAG	SRC	N	CONNECTION_I	...	...	...	CONNECTION_N
4 bytes	6 bytes	4 bytes	6 bytes	6 bytes	6 bytes	6 bytes	6 bytes

FLAG - Flag (2 = Controller Inform Packet)

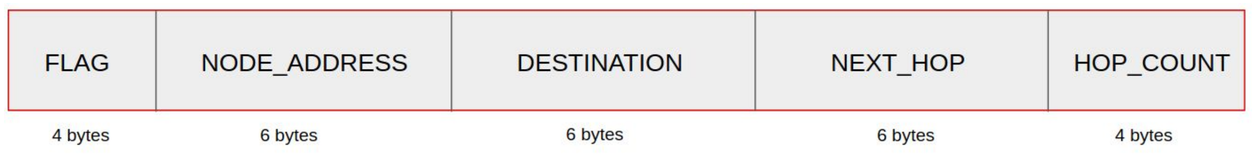
SRC - Source Address (Node)

N - Connection Count

CONNECTION_I - Port of CONNECTION_I connected to SRC

Node Inform Packet Protocol

The **Node Inform Packet** follows the following protocol:



FLAG	NODE_ADDRESS	DESTINATION	NEXT_HOP	HOP_COUNT
4 bytes	6 bytes	6 bytes	6 bytes	4 bytes

FLAG - Flag (3 = Node Inform Packet)

NODE_ADDRESS - Address of node being informed

DESTINATION - Destination Address (End User)

NEXT_HOP - Address of next hop for packet addressed to DESTINATION

HOP_COUNT - Distance in hops from NODE_ADDRESS to DESTINATION

Conclusion and Reflection

Overall, I found this assignment to be extremely beneficial towards developing my understanding how how networks actually work on a hardware/physical level. The assignment greatly helped me grasp an understanding of the routing protocols that are in place in the real world i.e Link State Routing and Distance Vector Routing.

At first the assignment was a bit daunting as both networks and their underlying technologies were fairly new to me. However, using both college resources and external resources found online I began to quickly understand just how networks worked and also began to greatly appreciate the brilliance behind their design.

Getting the underlying framework established was probably the most difficult and time consuming part of the assignment i.e building the routers, end users and controller and getting packets sent between them. However, once they were built and fully operational it was actually quite enjoyable and rewarding extending the network and adding multiple nodes and watching transmissions working successfully between them.

Another aspect of the assignment I found extremely rewarding was implementing the modification of DIjkstra's algorithm. At first it was very confusing trying to get my head around the algorithm. However, in our Algorithms & Data Structures module we recently studied *Breadth First Search* algorithms which I thought would be perfect for this use case. After researching around on the web a bit I discovered that this *BFS* algorithm is actually in fact the same as *Dijkstra's* however just without a weighted distance between each node. Once it was implemented it was amazing to see it in practice. I extended the network to a total of 9 end users and 6 routers which, in practice would be extremely difficult and time consuming to hardcode the shortest routes between each node. However, using the *BFS/Dijkstra's* algorithm, the shortest routes were calculated and implemented at the touch of a button almost immediately. It was amazing to see this in practice and it allowed me to appreciate the power of this method on a global scale of millions of end users and routers.

If I had more time and was to do some things differently I would attempt to fully implement *Dijkstra's* algorithm by hard coding weighted distances between each of the nodes in the network. I would also extend the packet protocols used in the development of this assignment to use more real-life applications e.g OSPF packets.

What I would have also liked to have done was allow each end user to decide whether to send a response, send to a new user or remain idle. However, I only had time to implement the response/idle options and was unable to implement the option to send to a new end user within the given time frame.

Overall I thoroughly enjoyed working on this assignment over the past month. I have gained a greater understanding of how connected networks and how the entire whole world is now connected.

Brandon Dooley - *16327446*