3D3 Project 2 Report
Communication in the Internet of Things
Simple Distance Vector Routing Protocol in C/C++

Team: Eoin Fitzsimons - 15324823
Jack Hay - 17344163
Siqi Wei - 14330586

Implementation Description:

Files:

- VertexList.h - list of vertices. Two instances created, one for vertices and one for the neighbours of the current router
- VertexList.cpp
- EdgeList.h - list of edges between vertices in the network
- EdgeList.cpp
- RoutingTable.h - distance vector entries with the destination node from the current router, distance to the destination and the predecessor before arriving at the destination
- RoutingTable.cpp
- final_router.cpp - main file to update

Key Features:

```
RoutingTable * compute_table(EdgeList *edges, VertexList *vertices, int src_index);
```

Routing table takes in a list of edges, a list of vertices, and the index of the source vertex and uses the Bellman-Ford algorithm to compute and return a Routing table. The RoutingTable class, EdgeList class, and VertexList class are each implemented as a linked list that includes relevant information. VertexList nodes include assigned character values, address strings, and assigned ports. EdgeList nodes include the two nodes the edge connects; u and v, and the corresponding weight of that edge. Resulting RoutingTable entries include the node's index, the location, and the next hop to that location as characters, and the computed distance.

The compute_table function is used iteratively to update the current routing table by ingesting new topology information as new edges and recomputing. The new edges are created based on routing table distribution from the source node's neighbours.

Message Format:

Messages are sent as a string in the format:

“Source_name, location_from_source[i], distance_to_location[i], location_from_source[i+1], distance_to_location[i+1], etc

Difficulties Faced:

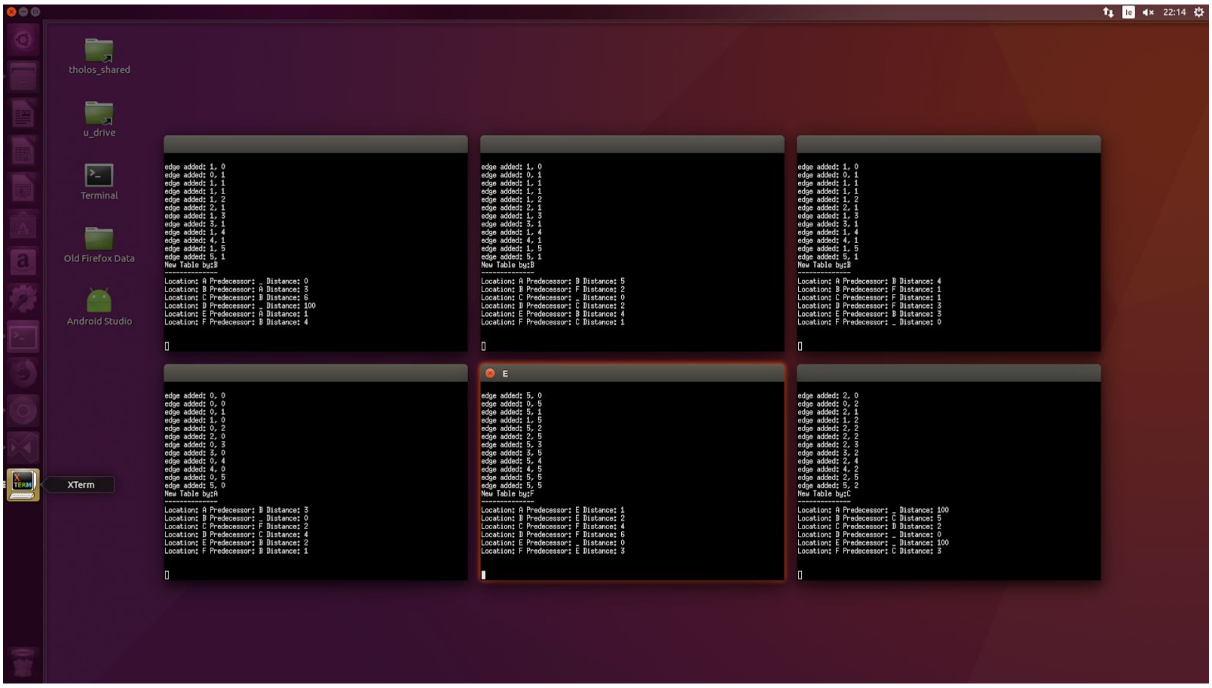
- Integrating the router’s client and server UDP processes.
    - To solve this we used fork() with both a client and a server process.
- Group organization and participation.
    - To solve this we redistributed the workload.
- Debugging the interactions between routers and the forked processes.

Sample Routing Output:

Picture format:

Top row: A, C, F

Bottom row: B, E, D



```
edge added: 1, 0
edge added: 0, 1
edge added: 1, 1
edge added: 1, 1
edge added: 1, 2
edge added: 2, 1
edge added: 1, 3
edge added: 3, 1
edge added: 1, 4
edge added: 4, 1
edge added: 1, 5
edge added: 5, 1
New Table by:B
--------------
Location: A Predecessor: _ Distance: 0
Location: B Predecessor: A Distance: 3
Location: C Predecessor: B Distance: 6
Location: D Predecessor: _ Distance: 100
Location: E Predecessor: A Distance: 1
Location: F Predecessor: B Distance: 4
```

```
edge added: 1, 0
edge added: 0, 1
edge added: 1, 1
edge added: 1, 1
edge added: 1, 2
edge added: 2, 1
edge added: 1, 3
edge added: 3, 1
edge added: 1, 4
edge added: 4, 1
edge added: 1, 5
edge added: 5, 1
New Table by:B
--------------
Location: A Predecessor: B Distance: 5
Location: B Predecessor: F Distance: 2
Location: C Predecessor: _ Distance: 0
Location: D Predecessor: C Distance: 2
Location: E Predecessor: B Distance: 4
Location: F Predecessor: C Distance: 1
```

```
edge added: 1, 0
edge added: 0, 1
edge added: 1, 1
edge added: 1, 1
edge added: 1, 2
edge added: 2, 1
edge added: 1, 3
edge added: 3, 1
edge added: 1, 4
edge added: 4, 1
edge added: 1, 5
edge added: 5, 1
New Table by:B
--------------
Location: A Predecessor: B Distance: 4
Location: B Predecessor: F Distance: 1
Location: C Predecessor: F Distance: 1
Location: D Predecessor: F Distance: 3
Location: E Predecessor: B Distance: 3
Location: F Predecessor: _ Distance: 0
```

```
edge added: 0, 0
edge added: 0, 0
edge added: 0, 1
edge added: 1, 0
edge added: 0, 2
edge added: 2, 0
edge added: 0, 3
edge added: 3, 0
edge added: 0, 4
edge added: 4, 0
edge added: 0, 5
edge added: 5, 0
New Table by:A
--------------
Location: A Predecessor: B Distance: 3
Location: B Predecessor: _ Distance: 0
Location: C Predecessor: F Distance: 2
Location: D Predecessor: C Distance: 4
Location: E Predecessor: B Distance: 2
Location: F Predecessor: B Distance: 1
```

```
edge added: 5, 0
edge added: 0, 5
edge added: 5, 1
edge added: 1, 5
edge added: 5, 2
edge added: 2, 5
edge added: 5, 3
edge added: 3, 5
edge added: 5, 4
edge added: 4, 5
edge added: 5, 5
edge added: 5, 5
New Table by:F
--------------
Location: A Predecessor: E Distance: 1
Location: B Predecessor: E Distance: 2
Location: C Predecessor: F Distance: 4
Location: D Predecessor: F Distance: 6
Location: E Predecessor: _ Distance: 0
Location: F Predecessor: E Distance: 3
```

```
edge added: 2, 0
edge added: 0, 2
edge added: 2, 1
edge added: 1, 2
edge added: 2, 2
edge added: 2, 2
edge added: 2, 3
edge added: 3, 2
edge added: 2, 4
edge added: 4, 2
edge added: 2, 5
edge added: 5, 2
New Table by:C
--------------
Location: A Predecessor: _ Distance: 100
Location: B Predecessor: C Distance: 5
Location: C Predecessor: D Distance: 2
Location: D Predecessor: _ Distance: 0
Location: E Predecessor: _ Distance: 100
Location: F Predecessor: C Distance: 3
```