ITCS – 6166

Computer Communication and Networks

Assignment – 4 | Group Activity

Shubham Gupta | 801081963

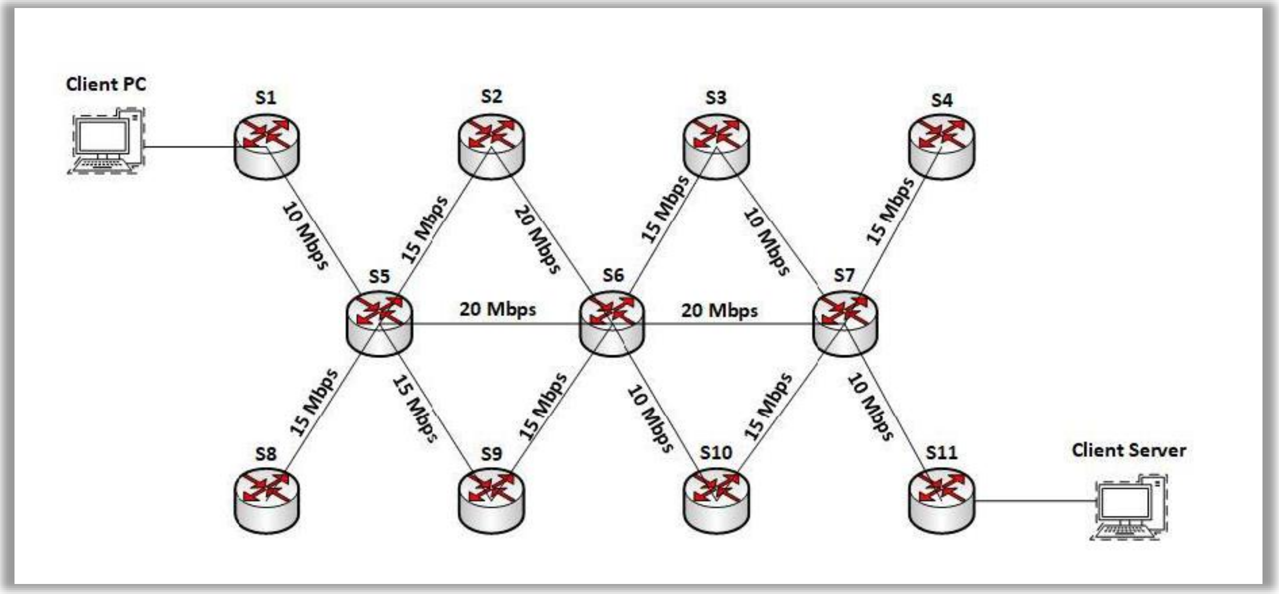
Riddhi Sanjay Panchal | 801076701

Naman Manocha | 801077765

Dhananjay Arora | 801077164

This project report is combined work done by us , which consists of implementing shortest path Dijkstra algorithm and implementation of given topology. There are two python files:

1) spr.py – contains the implementation of Dijkstra's Algorithm
2) topology.py – contains the given topology.



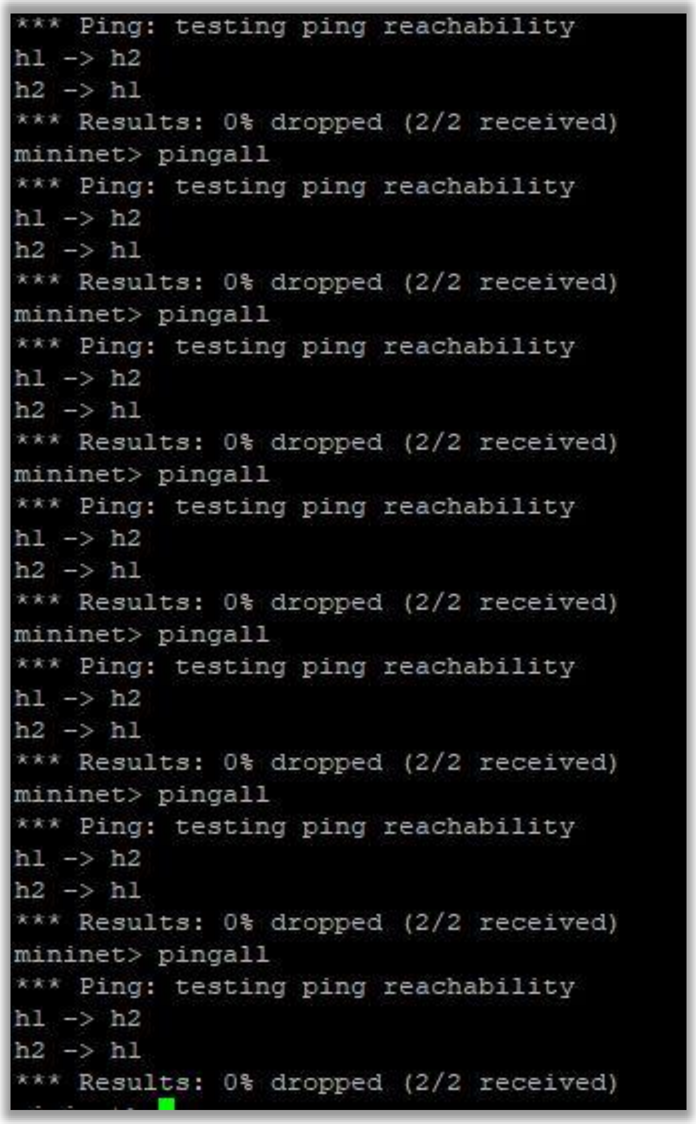
Dijkstra's Algorithm is a greedy algorithm to find single source shortest paths between nodes in a graph. From a given node (i.e. source vertex), the algorithm finds shortest path between source and every other node in the graph.

Dijkstra's Algorithm works on below assumptions:

- Graph should be connected.
- Edges can be directed or undirected.
- Edge weights should not be negative, i.e. w(e)>0

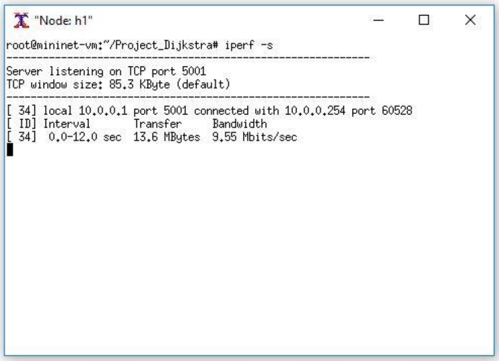
Here are the snapshots of the program:

pingall :

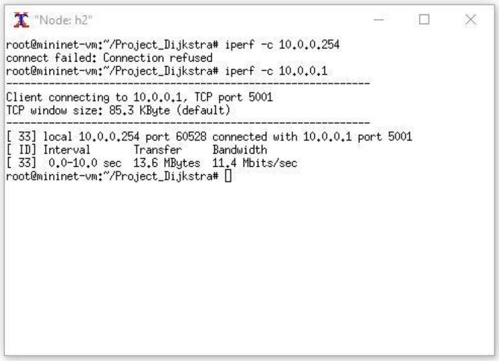


```
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

iperf :

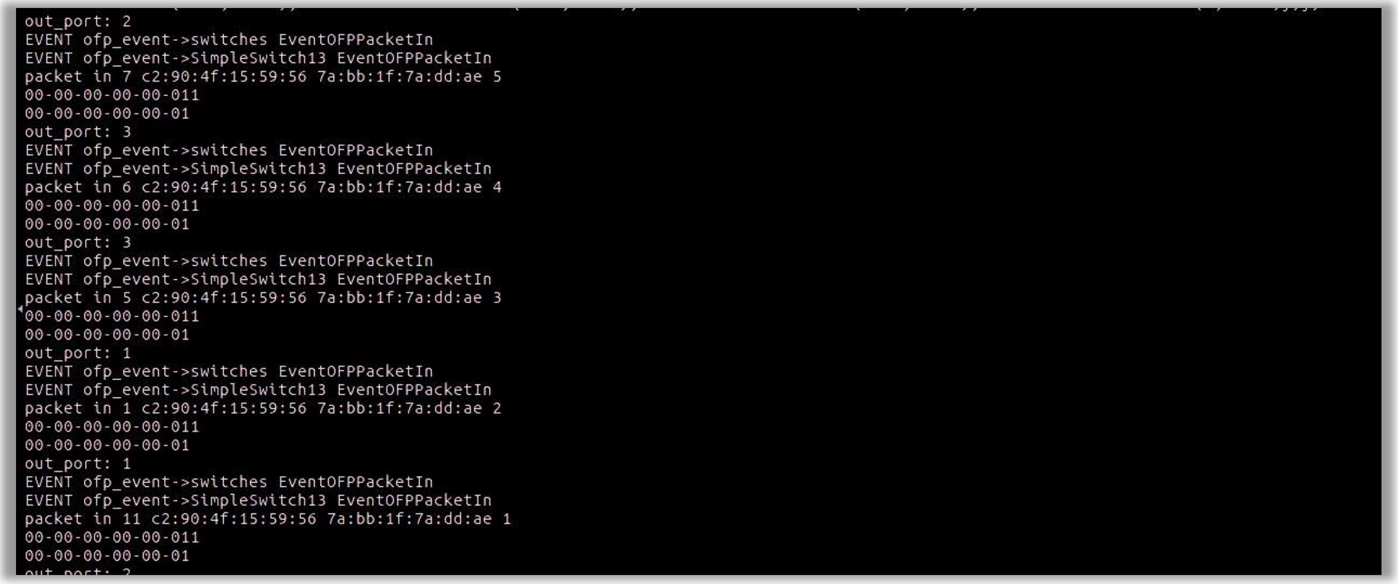


```
root@mininet-vm:~/Project_Dijkstra# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 34] local 10.0.0.1 port 5001 connected with 10.0.0.254 port 60528
[ ID] Interval       Transfer     Bandwidth
[ 34]  0.0-12.0 sec  13.6 MBytes  9.55 Mbits/sec
```



```
root@mininet-vm:~/Project_Dijkstra# iperf -c 10.0.0.254
connect failed: Connection refused
root@mininet-vm:~/Project_Dijkstra# iperf -c 10.0.0.1
------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 33] local 10.0.0.254 port 60528 connected with 10.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 33]  0.0-10.0 sec  13.6 MBytes  11.4 Mbits/sec
root@mininet-vm:~/Project_Dijkstra# 
```

OpenFlow Table:



```
out_port: 2
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 7 c2:90:4f:15:59:56 7a:bb:1f:7a:dd:ae 5
00-00-00-00-00-011
00-00-00-00-00-01
out_port: 3
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 6 c2:90:4f:15:59:56 7a:bb:1f:7a:dd:ae 4
00-00-00-00-00-011
00-00-00-00-00-01
out_port: 3
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 5 c2:90:4f:15:59:56 7a:bb:1f:7a:dd:ae 3
00-00-00-00-00-011
00-00-00-00-00-01
out_port: 1
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 1 c2:90:4f:15:59:56 7a:bb:1f:7a:dd:ae 2
00-00-00-00-00-011
00-00-00-00-00-01
out_port: 1
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 11 c2:90:4f:15:59:56 7a:bb:1f:7a:dd:ae 1
00-00-00-00-00-011
00-00-00-00-00-01
out_port: 2
```

Description of Implementation of code:

We build a software defined networking (SDN) network using simulated switches with an SDN controller. Dijkstra's algorithm run between two pairs of end hosts. We first built the algorithm that takes in a nested dictionary list that holds all nodes and every connected node to that given node.

Dijkstra's algorithm takes a graph or topology as an input with a source node and finds shortest paths from source to all vertices in the given graph.

- We create a set that keeps track of vertices included in the shortest path tree, that is, whose minimum distance from source is calculated and finalized.
- Assign a distance value to all vertices in the input graph. Initialize all as infinite. Assign 0 for distance value for the source vertex.
- While set doesn't include all vertices:
  a) Pick a vertex u which is not there in the set and has minimum distance value.
  b) Include u to the set.
  c) Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

Time Complexity of the implementation is O(V^2). If the input graph is represented using adjacency list, it can be reduced to O(E log V) with the help of binary heap.

Adjacency and path_map are the 2 default dictionaries that store shortest path routes.

- Get_raw_path() method takes source and destination as input parameters and get a raw path with list of nodes to be explored from current node.
- Get_path() method takes source, destination, first port and final port as input parameters and gets a cooked path - a list of (node,in_port,out_port)
- dijkstra_paths() is the method that finds the shortest path.
- SimpleSwitch13 class has several methods:
- switch_features_handler() takes ev as input and calls add_flow() method.
- add_flow() method takes datapath, priority, match, actions as input and further calls send_msg() method.
- Packet_in_handler() method takes ev as an input and calls send_msg() method.
- State_change_handler() method also takes ev as an input and deals with datapath.id
- Install_path() method takes src_sw, dst_sw, in_port, last_port, ev as an input and attempts to install a path between this switch and some destination
- Install_path() method takes p and ev as an input and is polymorphism.
- get_topology() method takes ev as an input and is used to get topology.

Topology python file is made based on the topology provided in the assignment taking 2 hosts and 11 switches.

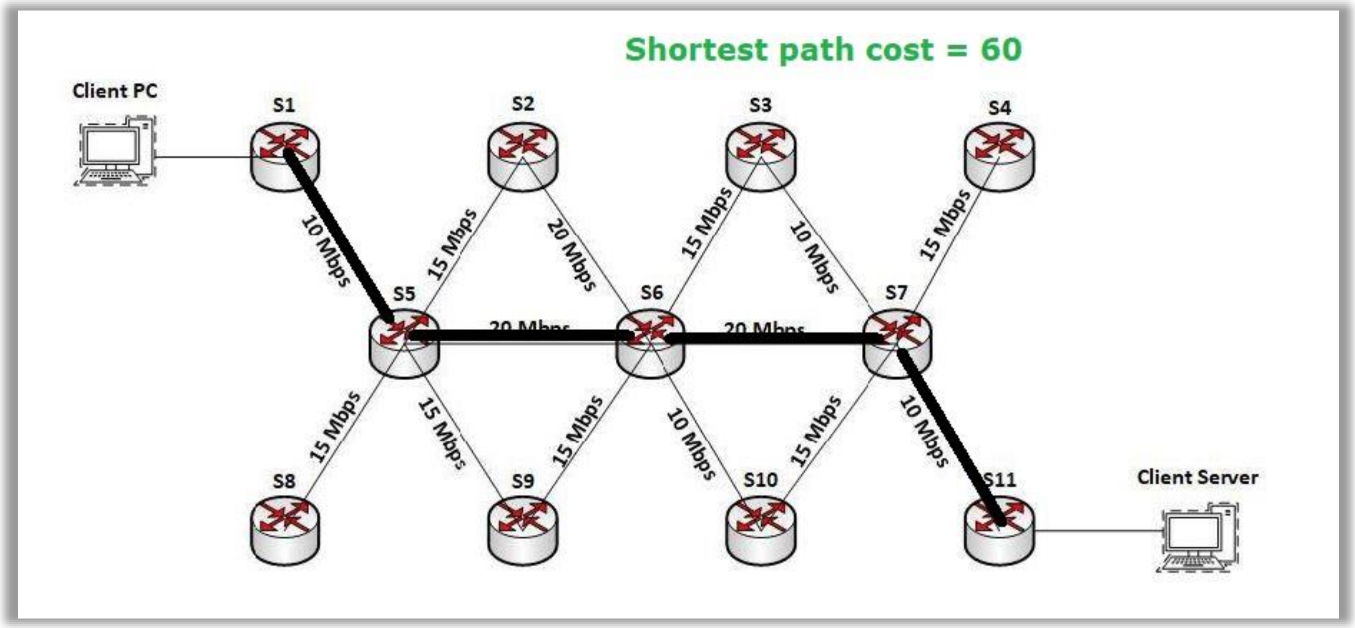
Efficiency:

We have implemented the code considering all the possible negative test cases such as:

- Installed a flow to avoid a packet in next time.
- Learn a mac address to avoid Flooding.

Path Chosen by our algorithm:

Shortest Path is coming to be 60 according to the code written.



References:

[1] https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/