COE 4DN4 ADVANCED INTERNET COMMUNICATIONS

ASSIGNMENT #1 TRAFFIC ENGINEERING

Jingnan Chen (1073196), Wei Zhang (0951321)
chenj39@mcmaster.ca, zhangw32@mcmaster.ca

Table of Contents

Jingnan Chen
Wei Zhang

PART (A)

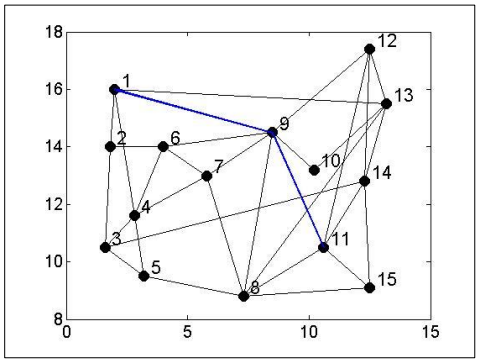

Figure 1: Node 11 to Node 1

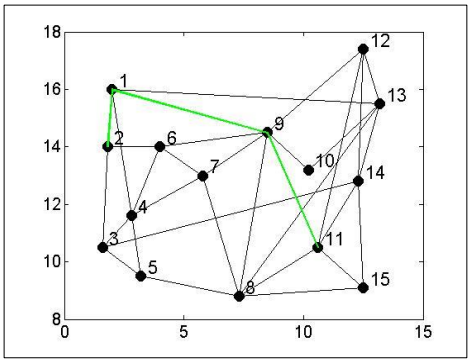

Figure 2: Node 11 to Node 2

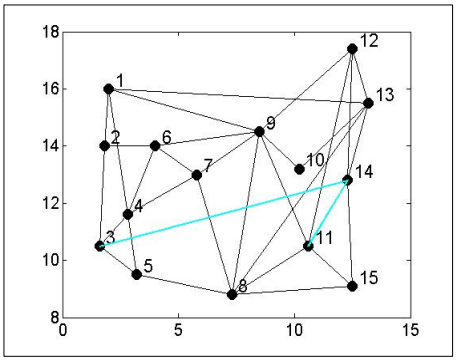

Figure 3: Node 11 to Node 3

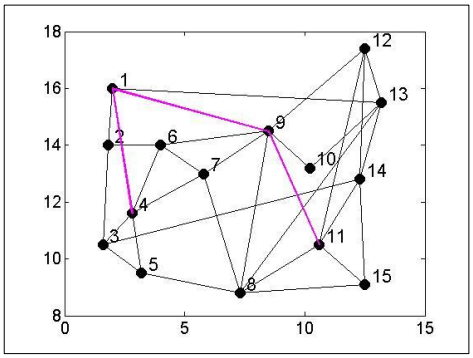

Figure 4: Node 11 to Node 4

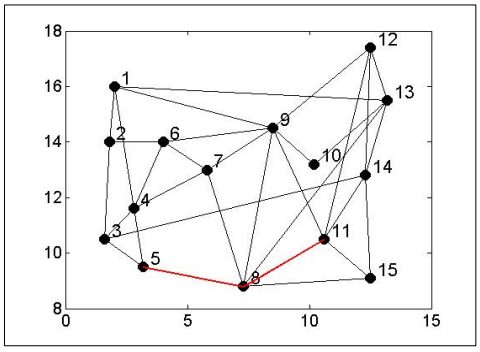

Figure 5: Node 11 to Node 5

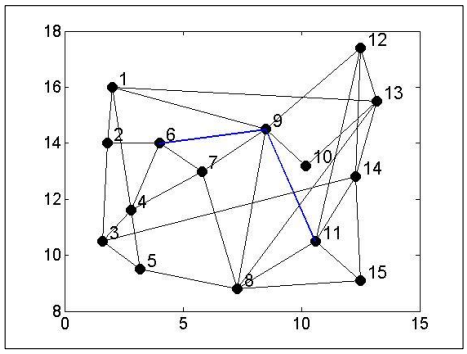

Figure 6: Node 11 to Node 6

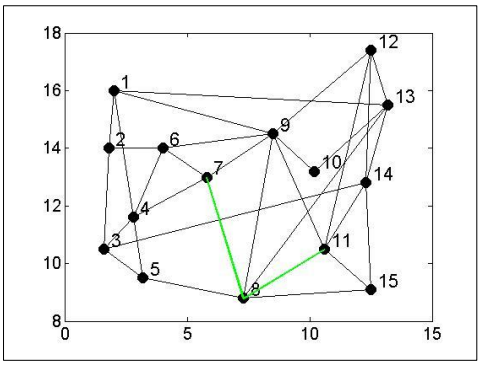

Figure 7: Node 11 to Node 7

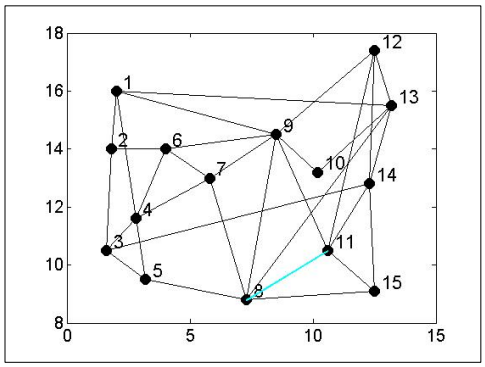

Figure 8: Node 11 to Node 8

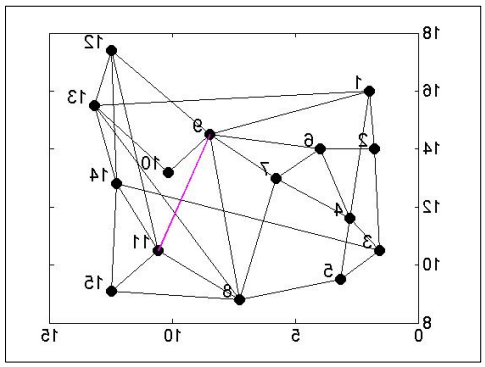

Figure 9: Node 11 to Node 9

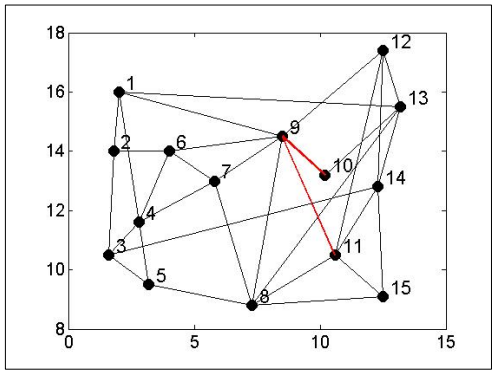

Figure 10: Node 11 to Node 10

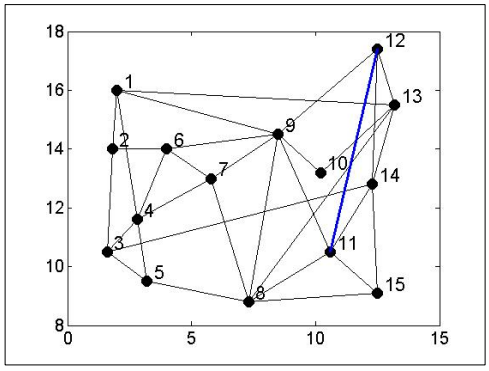

Figure 11: Node 11 to Node 12

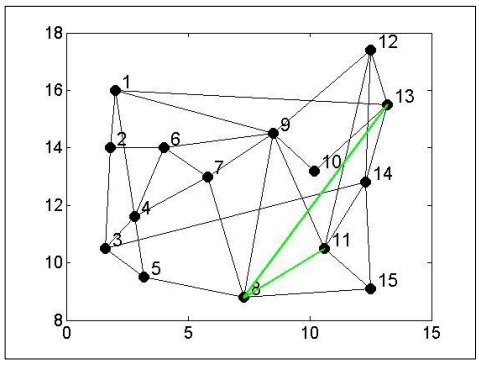

Figure 12: Node 11 to Node 13

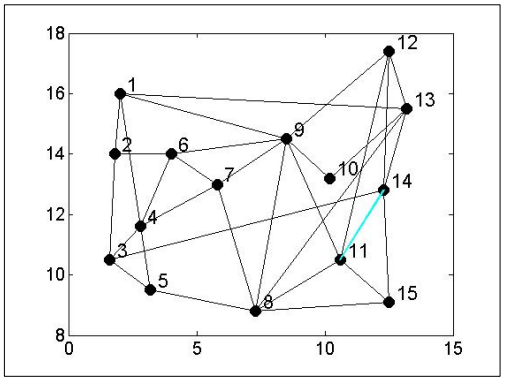

Figure 13: Node 11 to Node 14

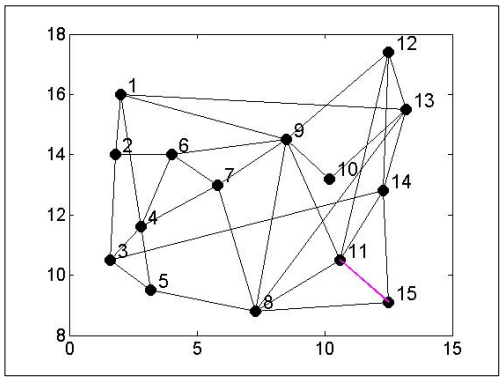

Figure 14: Node 11 to Node 15

The above figures show the shortest paths between Source Node 11 to all other nodes, and the list shortest paths is as follows:

Table 1: Paths from Node 11 to all other nodes

Destination Node:	Paths:
1	11-9-1
2	11-9-1-2
3	11-14-3
4	11-9-1-4
5	11-8-5
6	11-9-6
7	11-8-7
8	11-8
9	11-9
10	11-9-10
12	11-12
13	11-8-13
14	11-14
15	11-15

PART (B)

In this part, we route the flows using the shortest delay paths. Since each the path can receive either 0% or 100% of its requested rate, and all the flows have the same requested rate. In other words, we are asked to route as much traffic as possible, which is the same as to route as many flows as possible. We have tried a lot of combinations from the given path table, and we end up with the following configuration that can route the most flows.

Table 2: One of the good solution of the system

Flow #	Source	Destination	Req. Rate (Pkt/sec)	Path(s) taken	Allocated Rate	Session Delay (ms)
1	1	7	800	1-2-6-7	800	15
2	2	8	800			
3	3	13	800	3-4-1-13	800	15

4	4	11	800			
5	5	1	800	5-4-6-2-1	800	20
6	6	9	800	6-9	800	5
7	7	15	800	7-4-3-14-15	800	20
8	8	2	800			
9	9	6	800	9-6	800	5
10	10	3	800	10-13-14-3	800	15
11	11	12	800	11-14-13-12	800	15
12	12	10	800	12-13-10	800	10
13	13	4	800	13-1-4	800	10
14	14	5	800	14-11-8-5	800	15
15	15	14	800	15-14	800	5
16	1	15	800	1-9-8-15	800	15
17	2	1	800			
18	3	2	800	3-2	800	5
19	4	10	800			
20	5	13	800	5-8-13	800	10
21	6	5	800	6-4-5	800	10
22	7	8	800	7-8	800	5
23	8	4	800			
24	9	14	800	9-12-14	800	10
25	10	7	800	10-9-7	800	10
26	11	12	800	11-12	800	5
27	9	14	800			
28	13	11	800	13-8-11	800	10
29	14	6	800			
30	15	9	800	15-11-9	800	10
31	1	8	800			
32	2	5	800	2-3-5	800	10
33	7	1	800	7-9-1	800	10
34	8	10	800	8-9-10	800	10
35	14	9	800	14-12-9	800	10
36	15	7	800	15-8-7	800	10

B.1) The largest aggregate flow that we can route is 800 Pkt/sec/flow x 27 Flows = 21600 Pkt/sec, as shown above.

B.2) The algorithm flow-chart

Box 1: In the MAIN_assignment_1 script:

```
script_SPRINT_TOPOLOGY
define_flows
Loop_number = 20000;
BEST_SEQ = Find_Best_Solution(FLOW,TOP,Loop_number);
```

Comment: Set up the topology matrix TOP and flow matrix FLOW and use the function Find_Best_Solution to find the best solution from 20000 times of random permutations of 36.

Box 2: In the Find_Best_Solution function:

```
BEST_SEQ = Find_Best_Solution(FLOW,TOP,Loop_number);
prev_num_path_routed = 0;
num_path_routed = 0;
Mu = TOP * 1000;
for perm_loop = 1:Loop_number
    num_path_routed = 0;
    sequence = randperm(36);
    Lambda  = zeros(N,N);
    Delay       = zeros(N,N);
```

Comment 1: Generate the random permutations then try to route all the flows in the inner For loop and record the number of flows that are routed and set up the lambda and delay matrixes.

```
for  i = sequence
    flow_data = FLOW(i,1:3); % get the flow informatoin
    [Delay]  = find_network_delay(Lambda, Mu);
    [Dist,Pred]= Dijkstra_Shortest_Path_Tree(flow_data(1), Delay,TOP);
```

Comment 2: Call Dijkstra's Shortest Delay Path function and pass the source of each flow as a vector flow_data, Dealy and TOP matrix. The delay matrix is generated from the given find_network_delay function.

Box 3: In the Dijkstra_Shortest_Path_Tree script:

```
for u = U
     if ((u ~=0)&&(TOP(v,u) ~=0))
          % check the lowest delay
          if(Dist(v) + W(v,u) < Dist(u))
               Dist(u) = Dist(v)+W(v,u);
               Pred(u) = v;
          end
     end
 end
```

Comment: in the modified Dijkstra's Algorithm, use the TOP matrix to find if there is an edge between two nodes and use Delay matrix to determine the lowest delay path. And return the Pred and Dist vector. Other logic is similar to the Dijkstra's Algorithm shown in the lecture notes.

Box 4: In the Find_Best_Solution function:

```
path = [];
prev_node = Pred(flow_data(2));
while (prev_node ~= flow_data(1))
     path = [prev_node path];
     prev_node = Pred(prev_node);
end
path = [flow_data(1) path];
path = [path flow_data(2)];
```

Comment 1: Trace the shortest delay path to the source, starting from the destination flow_data(2). Then generate the path with source at the beginning and destination at the end to as a vector "path".

```
path_delay = test_path(path, 800, Lambda, Mu);
if (path_delay <= 0.05)
        [path_delay, Lambda] = route_path(path, 800, Lambda, Mu);
         num_path_routed = num_path_routed + 1;
end
```

Comment 2: Record the number of flows that have been successfully routed.

```
if (num_path_routed > prev_num_path_routed)
    % print the number of flows that have been routed so far
    fprintf('best num of flows routed so far: %g \n', num_path_routed);
    prev_num_path_routed = num_path_routed;
    % record the bast solution
    BEST_SEQ = sequence;
end
```

Comment 3: if the number of path routed is greater than the previous recorded number of path routed, store the new sequence. After certain number of loops of random permutation is done, return the best sequence in all of these trials to the Main script.

Box 5: In the MAIN_assignment_1 script:

```
Lambda      = zeros(N,N);
Delay       = zeros(N,N);
```

Comment 1: After getting the best sequence, re-initialize Lambda and Delay matrix.

```
for  i = BEST_SEQ
     flow_data = FLOW(i,1:3);
    [Delay]  = find_network_delay(Lambda, Mu);
    [Dist,Pred]= Dijkstra_Shortest_Path_Tree(flow_data(1), Delay,TOP);

    path = [];
    prev_node = Pred(flow_data(2));
    while (prev_node ~= flow_data(1))
        path = [prev_node path];
        prev_node = Pred(prev_node);
    end
    path = [flow_data(1) path];
    path = [path flow_data(2)];

    path_delay = test_path(path, 800, Lambda, Mu);
    if (path_delay <= 0.05)
        [path_delay, Lambda] = route_path(path, 800, Lambda, Mu);
        fprintf('\nRouted flow #%g along path: ',i);
        fprintf('%g ', path);
        fprintf('. The Path Delay is %g \n', path_delay);
        fprintf('\n');
        num_path_routed = num_path_routed + 1;
    end
```

Comment 2: Apply the similar idea as Box 4 and try the route the flows from the BEST_SEQ. Then print the successfully routed paths.

Box 6: In the MAIN_assignment_1 script:

```
Delay  = find_network_delay(Lambda, Mu);
Nq = Compute_Nq(Lambda, Mu);
```

Comment: After finishing trying all the flows, find the latest Delay matrix and Nq matrix.

Box 7: In the Compute_Nq function:

```
if (Lambda(u,v) == 0)
    Nq(u,v)  = 0;
else
    Nq(u,v)  = (Lambda(u,v) /  Mu(u,v)) /(1 - Lambda(u,v) /  Mu(u,v));
end;
```

Comment: compute the Nq as Nq = rho/(1-rho)

B.3)

Here is the Lambda matrix (rates in 1000 pps)

0.00	0.80	0.00	0.80	0.00	0.00	0.00	0.00	0.80	0.00	0.00	0.00	0.80	0.00	0.00
0.80	0.00	0.80	0.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.80	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.00
0.80	0.00	0.80	0.00	0.80	0.80	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.80	0.80	0.00	0.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.80	0.00	0.80	0.00	0.00	0.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.80	0.00	0.80	0.80	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.80	0.00	0.80	0.00	0.80	0.00	0.80	0.00	0.80	0.00	0.80
0.80	0.00	0.00	0.00	0.00	0.80	0.80	0.80	0.00	0.80	0.80	0.80	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.00	0.00	0.00	0.80	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.80	0.80
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.00	0.80	0.00	0.80	0.80	0.00
0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.00	0.80	0.00	0.80	0.00	0.80	0.00
0.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.80	0.80	0.00	0.80
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.80	0.00

Here is the Delay matrix (in millisec)

0.00	5.00	0.00	5.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	5.00	0.00	0.00
5.00	0.00	5.00	0.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	5.00	0.00	5.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00
5.00	0.00	5.00	0.00	5.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	5.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	5.00	0.00	5.00	0.00	0.00	5.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	5.00	0.00	0.00	0.00	5.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	5.00	0.00	5.00	0.00	5.00	0.00	5.00	0.00	5.00	0.00	5.00
5.00	0.00	0.00	0.00	0.00	5.00	5.00	5.00	0.00	5.00	0.00	5.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	5.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	5.00	0.00	0.00	5.00	0.00	5.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	5.00	5.00	0.00
5.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	5.00	0.00	5.00	0.00	5.00	0.00
0.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	5.00	5.00	0.00	5.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00	5.00	0.00	0.00	5.00	0.00

Here is the Nq matrix (in millisec)

0.00	4.00	0.00	4.00	0.00	0.00	0.00	0.00	4.00	0.00	0.00	0.00	4.00	0.00	0.00
4.00	0.00	4.00	0.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	4.00	0.00	4.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	0.00
4.00	0.00	4.00	0.00	4.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	4.00	0.00	0.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	4.00	0.00	4.00	0.00	0.00	4.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	4.00	0.00	0.00	0.00	4.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00
4.00	0.00	0.00	0.00	0.00	4.00	4.00	4.00	0.00	4.00	0.00	4.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	0.00	0.00	0.00	4.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	4.00	0.00	0.00	4.00	0.00	4.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	0.00	0.00	0.00	4.00	4.00	0.00
4.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00
0.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	4.00	4.00	0.00	4.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	0.00	0.00	4.00	0.00	0.00	4.00	0.00

PART (C)

From Part (B) and the Delay matrix, we have the path delay as follows:

Table 3: Delay for each session

Flow #	Paths	Session Delay (ms)
1	1-2-6-7	5+5+5 = 15
3	3-4-1-13	5+5+5 = 15
5	5-4-6-2-1	5+5+5+5 = 20
6	6-9	5
7	7-4-3-14-15	5+5+5+5 = 20
9	9-6	5
10	10-13-14-3	5+5+5 = 15
11	11-14-13-12	5+5+5 = 15
12	12-13-10	5+5 = 10
13	13-1-4	5+5 = 10
14	14-11-8-5	5+5+5 = 15
15	15-14	5
16	1-9-8-15	5+5+5 = 15
18	3-2	5
20	5-8-13	5+5 = 10
21	6-4-5	5+5 = 10
22	7-8	5
24	9-12-14	5+5 = 10
25	10-9-7	5+5 = 10
26	11-12	5
28	13-8-11	5+5 = 10
30	15-11-9	5+5 = 10
32	2-3-5	5+5 = 10
33	7-9-1	5+5 = 10
34	8-9-10	5+5 = 10
35	14-12-9	5+5 = 10
36	15-8-7	5+5 = 10

Then we can compute the average packet delay in the system using “Session-by-Session” as follows:

$E[T_{system}]$ =
1/27*(15+15+20+5+20+5+15+15+10+10+15+5+15+5+10+10+5+10+10+5+10+10+10+10+10+10+10) = 10.74074 milli sec

PART (D)

Using the Nq matrix from part B, we can compute the average packet delay in the system using “Littles-Law-applied-to-entire-network” as follows:

$E[T_{system}] = E[Nq] / Lambda_{total} = 232/(27*800) *1000 = 10.74074$ milli sec