DEPARTAMENTO DE ELECTRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

MESTRADO INTEGRADO EM ENG. DE COMPUTADORES E TELEMÁTICA

ANO 2016/2017

DESEMPENHO E DIMENSIONAMENTO DE REDES

ASSIGNMENT GUIDE NO. 4

TRAFFIC ENGINEERING OF PACKET SWITCHED NETWORKS

Assignment Guide No. 4

1. Preamble

The aim of this assignment is to address the traffic engineering of an ISP (Internet Service Provider) core network based on MPLS (Multi-Protocol Label Switching). For a given network and a given set of estimated traffic flows to be supported, the traffic engineering task addressed in this assignment is to select a routing path for the LSP (Label Switched Path) of each traffic flow such that the performance of the network is optimized. The network performance is assessed by the Kleinrock approximation.

RECALL FROM THEORETICAL CLASSES:

The Kleinrock approximation assumes that each network link behaves as a *M*/*M*/1 queuing system. Consider a network composed by a set of unidirectional links (i,j), each one with a capacity μ_{ij} (in packets/second) and a propagation delay d_{ij} (in seconds). The network supports S packet flows s = 1, …, S, each one with a packet arrival rate λ_s (in packets/second) and a routing path composed by the links (i,j) defined in set R_s. The total arrival rate on connection (i,j) is:

$$\lambda_{ij} = \sum_{s:(i,j)\in R_s} \lambda_s$$

and the total traffic supported by the network is:

$$\gamma = \sum_{s=1\ldots S} \lambda_s$$

Then, by the Kleinrock approximation, the network average delay is:

$$W = \frac{1}{\gamma}\sum_{(i,j)}\left(\frac{\lambda_{ij}}{\mu_{ij}-\lambda_{ij}} + \lambda_{ij} d_{ij}\right)$$

and the average delay of each flow s is:

$$W_s = \sum_{(i,j)\in R_s}\left(\frac{1}{\mu_{ij}-\lambda_{ij}} + d_{ij}\right)$$

2. Assignment

Consider an MPLS (Multi-Protocol Label Switching) network of an ISP (Internet Service Provider) with the topology presented in Figure 1 where the gray nodes are transit nodes (they just provide connectivity between the other nodes). The thick connections in Figure 1 have a bandwidth capacity of 10 Gbps and the other connections have a bandwidth capacity of 1 Gbps. In Annex I, the R matrix defined, in MATLAB format, the network topology shown in Figure 1 where element R(i,j) gives the capacity (in Gbps) of the connection from node i to node j.

Consider that, in all network connections, the propagation delay is given by the fiber light speed (2×10^8 meters/second). In Annex I, the L matrix defines, in MATLAB format, the connection lengths (in kilometers) where element L(i,j) indicates the length of the connection from node i to node j.

Consider that the network must support an estimated traffic matrix. In Annex I, the T matrix defines, in MATLAB format, the traffic matrix where element T(i,j) defines the average traffic bandwidth sent from node i to node j.

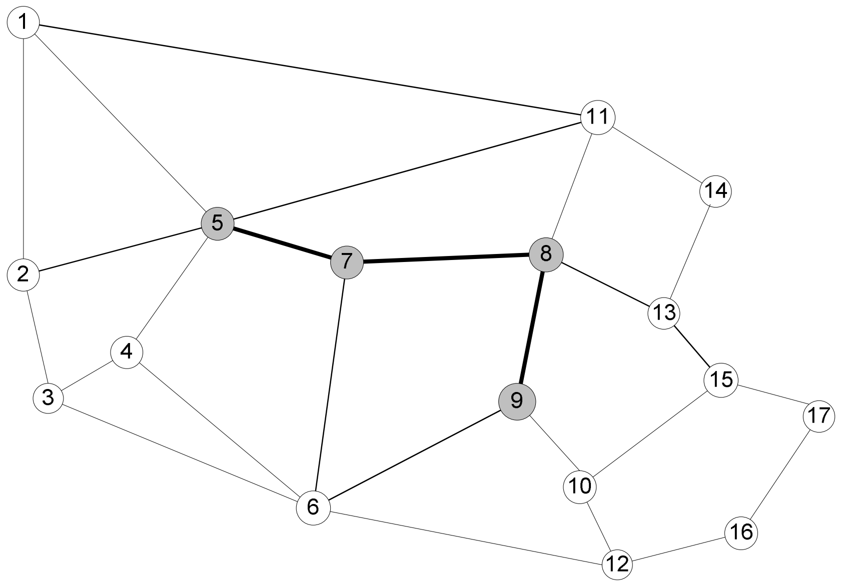

Figure 1: Topology of an ISP Network

Assume that in all traffic flows, the packet arrivals are Poisson processes and the packet sizes are exponentially distributed with an average size of 1000 Bytes.

Consider that the ISP requires each traffic flow to be routed through a single LSP (Label Switched Path). The LSPs between the same nodes are required to be symmetrical (*i.e.*, the network connections of an LSP from node *i* to node *j* must be the same network connections of the LSP from node *j* to node *i*).

For the implementation of this assignment, a MATLAB function is provided (file `ShortestPathSym.m`) that, for a given pair of nodes, determines the routing path that minimizes the sum of the connection costs in both directions between them. The function sintax is:

```
function [route]= ShortestPathSym(costs,node1,node2)
```

Input parameters:

`costs` – a square matrix of 17×17 elements where element `costs(i,j)` defines the cost of the connection from node *i* to node *j*;

`node1` – the first end node;

`node2` – the second end node.

NOTES:
- the cost values must be non-negative;
- the function ignores the values `costs(i,j)` for pairs of nodes such that there is no connection in the ISP network (as defined in Figure 1).

Output parameter:

`route` – a row array of 17 elements with the sequence of nodes in the routing path from `node1` up to `node2` (the remaining elements are zero).

Error codes:

When the output parameter `route` is a single element, its content indicates the following error in the input parameters:

`-1` : the matrix `costs` is not a square matrix of 17×17 elements

`-2` : the `node1` and/or `node2` are either equal or invalid
`-3` : at least one cost value is negative

a) Assume that the routing of each flow is done on the shortest path given by the sum of the propagation delays of the connections. Consider this solution as solution A. In Annex II, a MATLAB script is provided that computes solution A. The provided MATLAB script:
i) determines the average connection load (variable `AverageLoad`) and the maximum connection load (variable `MaximumLoad`); note that the load of a connection is the total bandwidth supported by the connection divided by its bandwidth capacity);
ii) determines the network average round-trip time (variable `AverageDelay`) and the maximum average round-trip time among all flows (variable `MaxAvDelay`);
iii) presents graphically in decreasing order: (i) the average round-trip delay of each flow and (ii) the load of each direction of each connection.
Analyze and run the provided MATLAB script and register the obtained performance values.

b) Compute a different solution in the following way. Start by considering the network empty. Then, for each flow, compute as its routing path the one that minimizes the sum of the connection loads that resulted from the previous selected routing paths. Consider this solution as solution B. Change the provided MATLAB script to compute this solution B and its performance values.

c) Compute another solution in the following way. Start by considering the network empty. Then, for each flow, select as its routing path the one that has the minimum average round-trip delay resulted from the previous selected routing paths (use the M/M/1 assumption for the delay of each connection). Consider this solution as solution C. Change the provided MATLAB script to compute this solution C and its performance values.

d) Compare solutions A, B and C (both the performance values and the graphical representations) and draw conclusions on which solution is better and in what cases.

e) Develop a multi-start local search optimization algorithm for *n* cycles, where *n* is an input parameter, to find a solution with the lowest network average round-trip delay. At the end, the algorithm outputs the highest average round-trip delay, the network average round-trip delay and the maximum connection load of the best found solution. Run your algorithm for *n* = 3, 10 and 30 cycles. Analyze the values of the solutions founds in the 3 cases. What do you conclude?

f) Adapt the previous developed algorithm to find a solution with the lowest maximum connection load. Run your algorithm for *n* = 3, 10 and 30 cycles. Analyze the values of the solutions founds in the 3 cases. What do you conclude?

g) Between the best solution found in e) and the best solution found in f), which one is, in your opinion, the solution that the ISP should adopt on its network? Explain your opinion.

Annex I – Problem Data Matrices

```
R= [0  1  0  0  1    0  0    0    0    0  1  0  0  0  0  0  0
    1  0  1  0  1    0  0    0    0    0  0  0  0  0  0  0  0
    0  1  0  1  0    1  0    0    0    0  0  0  0  0  0  0  0
    0  0  1  0  1    1  0    0    0    0  0  0  0  0  0  0  0
    1  1  0  1  0    0  10   0    0    0  1  0  0  0  0  0  0
    0  0  1  1  0    0  1    0    1    0  0  1  0  0  0  0  0
    0  0  0  0  10   1  0    10   0    0  0  0  0  0  0  0  0
    0  0  0  0  0    0  10   0    10   0  1  0  1  0  0  0  0
    0  0  0  0  0    1  0    10   0    1  0  0  0  0  0  0  0
    0  0  0  0  0    0  0    0    1    0  0  1  0  0  1  0  0
    1  0  0  0  1    0  0    1    0    0  0  0  0  1  0  0  0
    0  0  0  0  0    1  0    0    0    1  0  0  0  0  0  1  0
    0  0  0  0  0    0  0    1    0    0  0  0  0  1  1  0  0
    0  0  0  0  0    0  0    0    0    0  1  0  1  0  0  0  0
    0  0  0  0  0    0  0    0    0    1  0  0  1  0  0  0  1
    0  0  0  0  0    0  0    0    0    0  0  1  0  0  0  0  1
    0  0  0  0  0    0  0    0    0    0  0  0  0  0  1  1  0];
```

```
L= [0   119 0   0   147 0   0   0   0   0   292 0   0   0   0   0   0
    119 0   61  0   105 0   0   0   0   0   0   0   0   0   0   0   0
    0   61  0   40  0   136 0   0   0   0   0   0   0   0   0   0   0
    0   0   40  0   83  116 0   0   0   0   0   0   0   0   0   0   0
    147 105 0   83  0   0   65  0   0   0   181 0   0   0   0   0   0
    0   0   136 116 0   0   120 0   136 0   0   157 0   0   0   0   0
    0   0   0   0   65  120 0   95  0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   95  0   71  0   80  0   67  0   0   0   0
    0   0   0   0   0   136 0   71  0   51  0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   51  0   0   32  0   0   87  0   0
    292 0   0   0   181 0   0   80  0   0   0   0   0   73  0   0   0
    0   0   0   0   0   157 0   0   0   32  0   0   0   0   0   61  0
    0   0   0   0   0   0   0   67  0   0   0   0   0   68  40  0   0
    0   0   0   0   0   0   0   0   0   0   73  0   68  0   0   0   0
    0   0   0   0   0   0   0   0   0   87  0   0   40  0   0   0   45
    0   0   0   0   0   0   0   0   0   0   0   61  0   0   0   0   81
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   45  81  0];
```

```
T=  [0   32  78  23  0   60  0   0   0   13  14  15  25  55  17  14  12
    60  0   80  38  0   290 0   0   0   51  78  70  51  280 42  64  54
    82  84  0   11  0   260 0   0   0   19  15  48  17  84  18  19  15
    18  51  21  0   0   41  0   0   0   26  41  16  12  42  15  14  19
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    72  350 270 73  0   0   0   0   0   54  71  59  56  410 54  94  83
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    17  70  14  13  0   61  0   0   0   0   12  13  15  56  14  15  12
    12  48  13  41  0   54  0   0   0   19  0   13  14  63  12  15  14
    14  81  47  13  0   79  0   0   0   14  16  0   13  44  13  37  13
    14  43  12  16  0   47  0   0   0   19  13  18  0   50  15  23  12
    54  240 65  76  0   450 0   0   0   42  42  44  61  0   54  76  53
    16  49  14  13  0   49  0   0   0   14  13  16  15  63  0   15  14
    14  63  13  15  0   59  0   0   0   18  24  17  12  75  19  0   15
    21  45  16  22  0   73  0   0   0   12  14  17  14  59  14  16  0];
```

ANNEX II –PROPOSED CODE FOR SOLUTION A

```
Matrizes;
miu= R*1e9/(8*1000);
NumberLinks= sum(sum(R>0));
lambda_s= T*1e6/(8*1000);
gama= sum(sum(lambda_s));
d= L*1e3/2e8;

pairs= [];
for origin=1:16
    for destination=(origin+1):17
        if T(origin,destination)+T(destination,origin)>0
            pairs= [pairs; origin destination];
        end
    end
end
npairs= size(pairs,1);
lambda= zeros(17);
routes= zeros(npairs,17);
for i=1:npairs
    origin= pairs(i,1);
    destination= pairs(i,2);
    r= ShortestPathSym(d,origin,destination);
    routes(i,:)= r;
    j= 1;
    while r(j)~= destination
        lambda(r(j),r(j+1))= lambda(r(j),r(j+1)) + ...
            lambda_s(origin,destination);
        lambda(r(j+1),r(j))= lambda(r(j+1),r(j)) + ...
            lambda_s(destination,origin);
        j= j+1;
    end
end
Load= lambda./miu;
Load(isnan(Load))= 0;
MaximumLoad= max(max(Load))
AverageLoad= sum(sum(Load))/NumberLinks
AverageDelay= (lambda./(miu-lambda)+lambda.*d);
AverageDelay(isnan(AverageDelay))= 0;
AverageDelay= 2*sum(sum(AverageDelay))/gama
Delay_s= zeros(npairs,1);
for i=1:npairs
    origin= pairs(i,1);
    destination= pairs(i,2);
    r= routes(i,:);
    j= 1;
    while r(j)~= destination
        Delay_s(i)= Delay_s(i)+ 1/(miu(r(j),r(j+1))-...
            lambda(r(j),r(j+1))) + d(r(j),r(j+1));
        Delay_s(i)= Delay_s(i)+ 1/(miu(r(j+1),r(j))-...
            lambda(r(j+1),r(j))) + d(r(j+1),r(j));
        j= j+1;
    end
end
MaxAvDelay= max(Delay_s)
```

```
subplot(1,2,1)
printDelay_s= sortrows(Delay_s,-1);
plot(printDelay_s)
axis([1 npairs 0 1.1*MaxAvDelay])
title('Flow Delays')
subplot(1,2,2)
printLoad= sortrows(Load(:),-1);
printLoad= printLoad(1:NumberLinks);
plot(printLoad)
axis([1 NumberLinks 0 1])
title('Link Loads')
```