MINI-PROJECT 2:

***TRAFFIC ENGINEERING OF TELECOMMUNICATION NETWORKS***

Pedro Caio (105127) and Miguel Ferreira (93419)

Auto-evaluation: 50 % / 50 %

17/12/2024

Contents

[Task 1 (evaluation: 6.0 values) 1](#_Toc184658360)

[1.a. 1](#_Toc184658361)

[1.b 1](#_Toc184658362)

[1.c 2](#_Toc184658363)

[1.d 2](#_Toc184658364)

[1.e 2](#_Toc184658365)

[1.f 2](#_Toc184658366)

[References 3](#_Toc184658367)

[Appendix: 3](#_Toc184658368)

[Exercise 1: 3](#_Toc184658369)

# Task 1 (evaluation: 6.0 values)

In Task 1, consider the solutions where all traffic flows of both unicast services ( and ) are routed through the shortest propagation delay path.

## 1.a.

Compute the worst round-trip delay and the average round trip delay of each of the 3 services (presenting all values in milliseconds) if the anycast nodes of the anycast service () are network nodes and .

**Answer:**

## 1.b

Determine the link loads of all links and the worst link load of the network of the previous solution.

**Answer:**

## 1.c

Consider that you can freely select the anycast nodes of the anycast service. Try all possible combinations of nodes and select the one that minimizes the worst link load. Indicate the two selected nodes, the worst round-trip delay and the average round trip delay of each service.

**Answer:**

## 1.d

Again, consider that you can freely select the anycast nodes of the anycast service. Try all possible combinations of nodes and select the one that minimizes the worst round-trip delay of the anycast service. Indicate the two selected nodes, the worst round-trip delay and the average round trip delay of each service.

**Answer:**

## 1.e

Again, consider that you can freely select the anycast nodes of the anycast service. Try all possible combinations of 2 nodes and select the one that minimizes the average round trip delay of the anycast service. Indicate the two selected nodes, the worst round-trip delay and the average round trip delay of each service.

**Answer:**

## 1.f

Compare the results obtained in all experiments of Task 1 and draw all possible conclusions.

**Answer:**

**Uma imagem com texto, captura de ecrã, diagrama, Tipo de letra

Descrição gerada automaticamente**

# References

**Não existem fontes no documento atual.**

# Appendix:

## Exercise 1:

function [sP,nSP,totalCosts\_unicast, totalCosts\_anycast,T] = createPathFlows(costMatrix,T,k,unicastservices,anycastNodes)

%CREATEPATHFLOWS Computes the shortest paths and their costs for unicast and anycast services.

% [sP, nSP, totalCosts\_unicast, totalCosts\_anycast, T] = createPathFlows(D, T, k, unicastservices, anycastNodes)

% computes the shortest paths for all traffic flows in the network,

% considering both unicast and anycast services. The function returns the

% paths, the number of shortest paths, and the total costs for unicast

% and anycast services.

%

% Inputs:

% costMatrix - Matrix used to determine the shortest path between

% nodes (NxN matrix). This can represent weights, delays, or any

% other metric.

% T - Matrix of traffic flows (nFlowsx5 matrix), with:

% - Column 1: Service type (e.g. 1 for unicast service 1, 2 for unicast service 2, 3 for anycast service).

% - Column 2: Source node of the traffic flow.

% - Column 3: Destination node of the traffic flow (0 for anycast flows).

% - Column 4: Throughput from source to destination (in Gbps).

% - Column 5: Throughput from destination to source (in Gbps).

% k - Number of shortest paths to compute (1 for the shortest path).

% unicastservices - Vector of unicast service types (e.g., [1, 2]).

% anycastNodes - Vector of nodes that can serve as anycast destinations (e.g., [3, 10]).

%

% Outputs:

% sP - Cell array of shortest paths for each traffic flow.

% nSP - Vector of the number of shortest paths for each traffic flow.

% totalCosts\_unicast - Vector of total costs (based on D metric) for unicast flows.

% totalCosts\_anycast - Vector of total costs (based on D metric) for anycast flows.

% T - Updated traffic flow matrix with destination nodes for anycast flows.

%

% Example:

% anycastNodes = [3, 10];

% unicastservices = [1, 2];

% [sP, nSP, totalCosts\_unicast, totalCosts\_anycast, T] = createPathFlows(D, T, 1, unicastservices, anycastNodes);

%

% This function uses the kShortestPath function to compute the shortest paths and their costs.

nFlows= size(T,1);

nFlows\_unicast = nnz(ismember(T(:,1),unicastservices));

n\_anycastNodes = length(anycastNodes);

totalCosts\_unicast = zeros(nFlows\_unicast, 1);

totalCosts\_anycast = zeros(nFlows - nFlows\_unicast, 1);

% Possible anycast paths

possible\_shortestPath = cell(n\_anycastNodes,1);

possible\_totalCost = zeros(n\_anycastNodes,1);

k= 1;

sP= cell(1,nFlows);

nSP= zeros(1,nFlows);

for f = 1:nFlows

if ismember(T(f,1), unicastservices)

% Flow is unicast

[shortestPath, totalCost] = kShortestPath(costMatrix,T(f,2),T(f,3),k);

sP{f}= shortestPath;

nSP(f)= length(totalCost);

totalCosts\_unicast(f) = totalCost;

else

% Flow is anycast

if ismember(T(f,2), anycastNodes)

sP{f} = {T(f,2)};

nSP(f) = 1;

T(f,3) = T(f,2);

% totalCost is already initialize to zero

else

for node = 1:n\_anycastNodes

[possible\_shortestPath(node), possible\_totalCost(node)] = kShortestPath(costMatrix,T(f,2),anycastNodes(node),k);

end

[M, I] = min(possible\_totalCost);

sP{f} = possible\_shortestPath(I);

nSP(f) = length(possible\_totalCost(I));

% index of array must begin in 1

totalCosts\_anycast(f - nFlows\_unicast) = possible\_totalCost(I);

T(f,3) = anycastNodes(I);

end

end

end

end