Universidad Politécnica de Cartagena

etsit escuela técnica superior de ingeniería de telecomunicación

PLANIFICACIÓN Y GESTIÓN DE REDES

Grado en Ingeniería Telemática
Course 2022-2023

Case of study. Fault tolerant topology design problem variant

Author:

Pablo Pavón Mariño

Introduction

In this case of study students should develop a Net2Plan algorithm for solving a variant of the topology design problem, described below.

Let $\mathcal{N}$ be a set of nodes in the network. For each pair of nodes (n_1, n_2), one demand of traffic exists, that requests carrying $traf(n_1, n_2)$ (in Gbps). Additionally, we denote as $dist(n_1, n_2)$ to the geographical distance between the two nodes, in km. The traffic of each demand $n_1 \rightarrow n_2$ is carried via the shortest path in km from n_1 to n_2. If such path does not exist, the demand is blocked.

The problem to solve consists in finding the links that the network must have, with the following constraints:

- No parallel links: Between any two nodes n_1 and n_2, we can have zero or one $n_1 \rightarrow n_2$ link (i.e. no option to have 2, 3, 4,... links $n_1 \rightarrow n_2$).
- Bidirectional links. If one link from node n_1 to node n_2 exists, there must be also one link from n_2 to n_1.
- The topology must be fault tolerant, meaning that it must be **biconnected**, as will be described later.

The solution should minimize the cost of the network given by the sum of the costs of the links created, where the cost of one link between two nodes (n_1, n_2) is given by:

$$cost(n_1, n_2) = dist(n_1, n_2) + K \times cap(n_1, n_2) \tag{1}$$

Note that:

- $cap(n_1, n_2)$ is the capacity of the link $n_1 \rightarrow n_2$ in Gbps, equal to the traffic that the link is carrying.
- K is the cost per Gbps of capacity in the link.

Biconnected topologies

Given a bidirectional topology, we say it is *connected* when for every node pair (n_1, n_2) there is at least a path from n_1 to n_2. Additionally, the number of connected components of the topology is the number of maximal connected subgraphs, i.e. the number of subgraphs that are connected, and are not part of other graphs. Said this, a topology is connected if and only if it has exactly one connected component. Fig. 1 shows a topology with three connected components: $\{n_1, n_2, n_3, n_4\}, \{n_5, n_6, n_8\}$ and $\{n_7\}$.

A topology is said to be biconnected, when it has no articulation nodes or also called *cut nodes*. A cut node is a node in the network that, if removed, would increase its number of connected components. Fig. 2 shows a topology that is not biconnected since it has one articulation node (n_3). Other example is Fig. 3, which has a so-called *bridge link* (n_3, n_5). A bridge link is a link that if removed, would increase the number of connected components of the topology. Note that if (n_3, n_5) is a bridge link, then both n_3 and n_5 are cut nodes, and thus the topology is not biconnected.

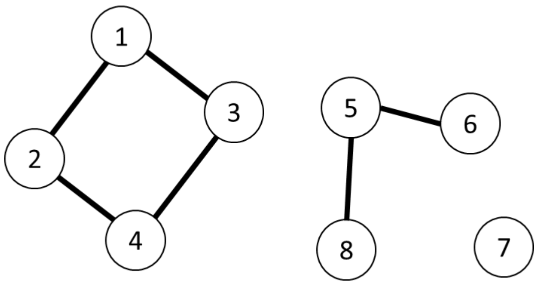

Figure 1: Topology with three connected components.

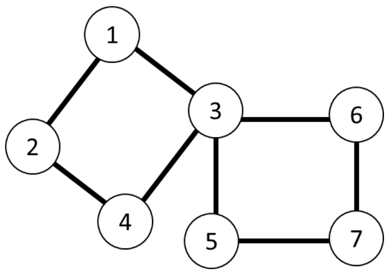

Figure 2: Connected, but not biconnected topology. n_5 is a cut node.

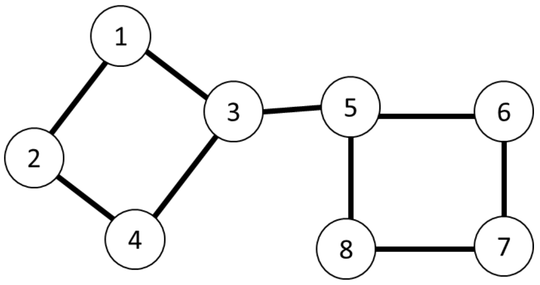

Figure 3: Connected, but not biconnected topology. n_3 and n_5 are cut nodes, the link (n_3, n_5) is a bridge.

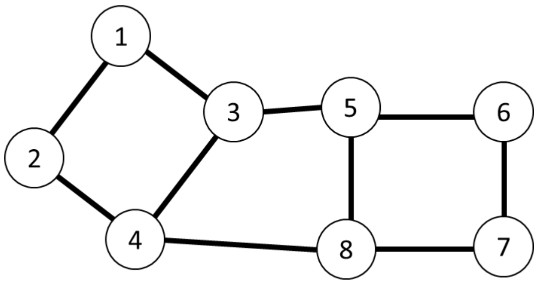

Figure 4: Biconnected topology (no cut nodes)

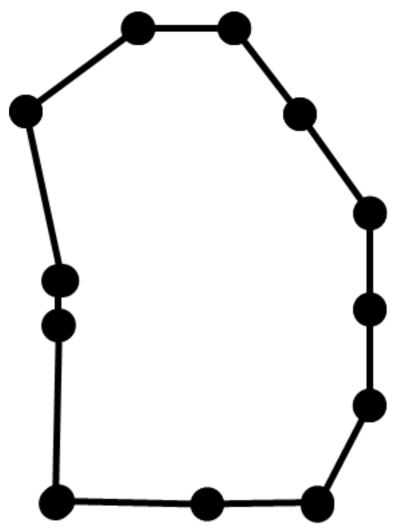

Figure 5: Topology where the minimum length biconnected topology is a ring

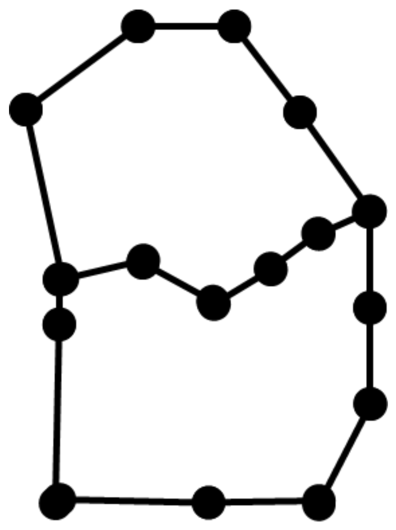

Figure 6: Topology where the minimum length biconnected topology is *not* a ring

Biconnected topologies like Fig. 4 are important in communication networks, since they produce fault tolerant networks, in the sense that can still be connected under the presence of a single node failure, or a single link failure.

The conceptually simplest biconnected topologies are the rings. However:

- Large rings involve typically a high latency, since in the worst case half of the ring have to be traversed to reach the destination node, and this can result in too many hops.
- For the same reason as above, the rings have a higher average number of hops, and more capacity is needed in the links.
- Finally, if the cost of the link is highly dependent of the link length, the minimum cost topology may again not be a ring. For instance, Fig. 5 shows an example where the minimum length biconnected topology is a ring, and Fig. 6 an example when it is not.

Implementation in Net2Plan

The students must individually develop a Net2Plan algorithm that solves the described problem.

- Net2Plan 0.6.6.0 version should be used, please download it from `https://github.com/girtel/Net2Plan/releases`. Note that this version requires Java 8 (i.e. and will not work with Java 9 or above versions).
- The algorithm will be applied on input Net2Plan designs, with no links, and just nodes, and one demand for each node pair.
- The distance between two nodes ($dist(n_1, n_2)$) is given as the **Euclidean distance between them**[1].
- The algorithm must accept the following input parameters:
  - `maxExecTimeSecs` (default value 60): Maximum running time (in seconds) of the algorithm. The algorithm must implement internally the procedure to stop the algorithm after, at most, this amount of time[2]. When stopped, the algorithm should return the best solution found so far.
  - `costPerGbps` (default value `5.0`): The cost per Gbps of capacity in the links (factor K in (1)).

The output of the algorithm is returned by just adding the link to the input design. There is **no need to add forwarding rules or routes to the traffic**. Also there is no need to set the capacity of the links (e.g. just set a capacity of 1000.0 for all), since the evaluator will assume that the capacity will be the one required for carrying the link traffic, and that will be automatically computed.

Net2Plan template

To assist the development of the algorithm, the student is provided with a `.java` template of the algorithm[3]. The template will include:

- A general skeleton of the algorithm.
- The code for reading the input attributes.
- A class called `DesignEvaluation`, that in its constructor receives a NetPlan object as an input, the value of the `costPerGbps` input parameter, and another parameter containing the maximum length of any design, which is the sum of all the potential links a design can have. This latter value is computed in the template in `const_maxSumOfLinkLengths`. See in the template an example of how to instantiate this object.

  The `DesignEvaluation` object created contains information that is relevant for knowing the cost of a design etc. We sketch here the key public methods of interest:
  - `isOk`. Returns true if the design satisfies all the constraints, which means it will have no blocked demands and will be biconnected.

[1]As provided by the method `getNodePairEuclideanDistance` in class `NetPlan`

[2]To measure the running time, use the Java method `System.nanoTime`

[3]The file is placed in Aula Virtual with name `TopologyDesign_TEMPLATE.java`

  - `getCost`. Returns the cost of the solution, given by the expression (1).
  - `isStrictlyBetterThan`. Returns true if the current evaluation corresponds to a solution that is better than the one passed as parameter. Being a better solution means: (i) have a lower number of blocked demands, (ii) if the same, have a lower number of connected components, (iii) if the same, have a lower number of cut nodes, (iv) if the same, have a lower cost according to (1).
  - `getAugmentedCost`. Returns the cost of the solution, with some penalizations added. These penalizations mean that when a solution has lower cost, but has more blocked demands, or more cut nodes, it will suffer more penalizations, and have a worse augmented cost. As stated in theory, algorithms should use this method, and not the `getCost` method to compare solutions, when the solutions may no satisfy the constraints.

Students are encouraged to reuse the provided template.

Additionally, the students are encouraged to follow the videos of the case of study, where the teacher will show how to setup the environment to program and test Net2Plan algorithms, and comment on best practices for a clean and well structured coding.

Delivery

The student should develop, reusing the provided template, a Net2Plan algorithm that solves the problem described. The student is free to choose any combination of the mathematical techniques and heuristics described in the course (e.g. tabu search, GRASP, ant-colony, evolutionary algorithm, ...).

This lab work should be solved **individually**. The students must produce **one** single `.java` file with name `TopologyDesign_XXXX.java`, where XXXX is the first and second surnames of the student[4].

The students should send by email to `pablo.pavon@upct.es` with the Java file and the student name in the email text:

- Those students sending the material before **Sunday May 14th at 23:59**, will have a *continuous evaluation* grade for the case study. Students will have a 5-10 minutes online interview via MS Teams on **May 17th**, in a slot that will be indicated in advance via Aula Virtual, **within the time frame 15:30-17:30** (i.e. during the time of the theory class, which will be dedicated to this purpose).
- Those students not entering the continuous evaluation grade, that send the material before Monday **May 28th at 23:59**, will be evaluated in the June call. These students will have a 5-10 minutes online interview via MS Teams on **May 31st**, in a slot that will be indicated in advance via Aula Virtual.

In all the cases, the work reception time will be taken as the time of reception of the email.

Students do not have to send any other material than the Java file. It is not allowed to spread the code in several Java files. If the student needs to define more than one class, they should be integrated in the single Java file to run.

[4]No special characters like 'ñ' or accents can be used, since this violates Java file naming conventions.

Evaluation

Evaluation - continuous evaluation

The evaluation of the case of study is performed as follows:

1. 60%: Average cost obtained in a the sample topology that will be available in Aula Virtual (`TopologyDesignSampleTopology.n2p`), tested with the values of the parameter `costPerGbps = { 5, 10, 100 }`, and a running time of 60 seconds in all the cases. The costs obtained by the different students will be ranked, and the grade received will be dependent on the average ranking among the tests. Algorithms failing to produce a valid solution (that does not return `true` in the `isOk` evaluation method), will not be accepted. Algorithms for which the running time exceeds the `maxExecTimeSecs` limit in more than three seconds, will be penalized. Algorithms that just repeat without variations or simply rephrase the algorithm example provided by the teacher in the lab session, will be rejected.
2. 40%: Creativity, technical depth of the algorithm, as well as clarity, adherence to recommended practices in Java programming and cleanness in the code. A more structured code, with an adequate amount of comments (not too much and not too few!) is preferred. Please follow the indications of the teacher in the lab videos.

The complete evaluation will be modulated by the individual online interview that each student will have with the teacher.

Just as an indication, the table below shows close to optimal cost values for the parameters of interest. For those students obtaining better values, congratulations!! ;-)

Orientative close-to-optimal costs

`costPerGbps`	**Cost**
5	33.29
10	43.29
100	188.23

Evaluation - final examen June

The case study in the final exam will be the same as the case study in the continuous evaluation:

- Students that did *not* present the case study in the continuous evaluation, will be evaluated as described above.
- Students that already presented the case study and do *not* send a new version of the algorithm again, will have its algorithm re-evaluated for the 60% corresponding to the ranks (but no interview is done again). His/her grade in the case study will be the maximum between the one obtained in the continuous evaluation and this one.
- Students that already presented the case study and send a new version of the algorithm again, will have its algorithm fully re-evaluated, including also the personal interview. His/her grade in the case study will be the maximum between the one obtained in the continuous evaluation and this one.

Evaluation - final examen July

Students that did not pass this course in the continuous evaluation, nor June, can make the final exam in July.

Students that already presented the case study, just inherit the grade of this part. For the rest, the case study in the final exam in July will be **a variation** respect to the case study described here. New instructions on this will be provided with enough time in advance.