Open Optical Networks
Lab 6

November 22, 2022

These exercise sets cover some aspects you will find useful for the final exam software development. This exercises can be part of the material for the final exam questions. You are strongly encouraged to find yourself a solution to the presented problems.

State of the Art

Until now, you should be able to emulate an optical network controller that operates in following way given a certain request:

1. the controller checks the availability and assigns a light-path (specific path + dedicated channel) for the specific request, accepting or rejecting it;
2. the controller performs the streaming along the designed connection, updating the status of each line during the propagation for the deployed channel (snr, latency and state);
3. the controller updates the routing-space considering each sub-path of the selected channel within the assigned path.

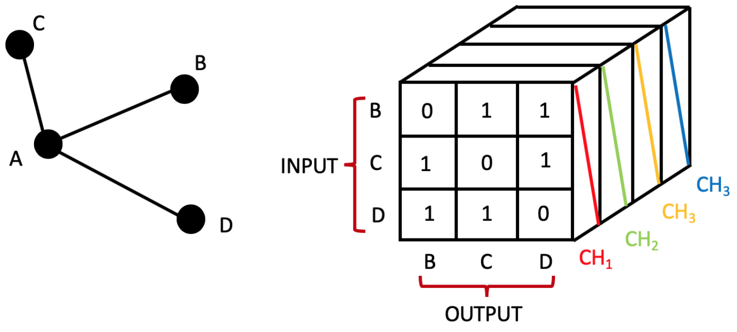

Figure 1: Node A switching matrix.

Exercises

Assume the following convention for the entire framework:

OCCUPIED = 0, FREE = 1

1. In order to introduce different verification methods for path availability, let's introduce a new attribute of the class **Node** called **switching_matrix**. When an object **Node** is instantiated, this attribute is set to none. Modify the Network class method **connect()** to obtain an initial switching matrix of the following type (referring to Fig. 1):

   ```
   switching_matrix = dict{'B': {'B': [0, 0, 0, 0],
   'C': [1, 1, 1, 1], 'D': [1, 1, 1, 1]}, 'C': {...},
   'D': {...}}
   ```

   The switching matrix dictionary has a key for each origin node. The content related to each key is another dictionary that contains for each arrival node a numpy array with the spectral occupation (one bit for each channel). You should be able to access the content of the switching matrix as follows:

   ```
   block = switching_matrix['in_node']['out_node']
   ```

   In the method **connect()**, initialize to 0 the numpy arrays for couples of nodes with the same names (e.g. B-B, C-C, D-D) and to 1 the others. Let's assume that ones the switching matrix has been initialized, it remains constant during the run.

2. Modify the routing space update method using the node switching matrices. In particular, when this method is called, the route space is rewritten performing for each path the multiplication of all the state line arrays and all switching matrix arrays that compose the path (excluding the switching matrix of the initial and last nodes).

does the route_space created in the previous lab changed a lot