Communication Network Design Course	Task Number 1
Full Name	**Person Code**
Dorsa Moadeli	**10926114**
Bahram Hedayati	**10870276**

We have learned from this interesting task to compare splittable and unsplittable types of flow formulation in terms of total carried traffic with respect to the connections with different amounts of link capacity (10, 15, 17, 20, 30, 40, 50 Gbps) for a network topology consisting of 7 nodes using Net2Plan network design tool.

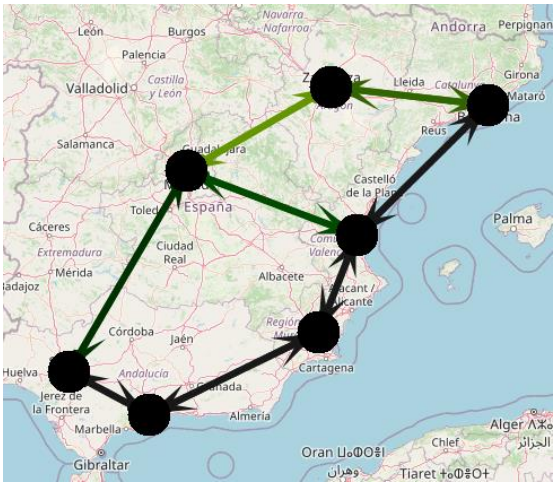

Figure 1: Network Topology

There are 3 kinds of constraints (solenoidality, capacity and integrity) for both splittable and unsplittable flow formulations which should be satisfied during the simulation.

	Splittable FF	Unsplittable FF
Solenoidality	$\sum_{k\in\Lambda_l} x_{k,l,c} - \sum_{k\in\Lambda_l} x_{l,k,c} = \begin{cases} v_c & \text{if } l = d_c \\ -v_c & \text{if } l = s_c \\ 0 & \text{otherwise} \end{cases} \quad \forall\, l,c$	$\sum_{k\in Al} x_{klc} - \sum_{k\in Al} x_{lkc} = \begin{cases} 1 & l = d_c \\ -1 & l = s_c \\ 0 & Otherwise \end{cases} \quad \forall\, l, c$
Capacity	$\sum_{c} x_{l,k,c} \le W \cdot F_{l,k} \quad \forall (l,k)$	$\sum_{c\in C} V_c\, x_{lkc} \le W F_{lk} \quad \forall\, l,k$
Integrity	$x_{l,k,c}$ integer $\quad \forall c,(l,k)$ $F_{l,k}$ integer $\quad \forall (l,k)$	x_{lkc} is binary $\quad \forall\, l,k,c$

Table 1: The constraints of Splittable and Unsplittable Flow Formulations

We exploited from the Java code template to apply the algorithm to the Net2Plan and corresponding network topology in 2 ways:

1. We obtained the time and value of objective function for different amounts of capacity in case of splittable FF.
2. We modified some lines of code in order to make it compatible with unsplittable FF.

The modifications are listed below:

- The type of *x_lkc* decision variable is changed to binary by this line of code:
  `op.addDecisionVariable("x_lkc" , true , new int [] {D , E } , 0 , 1);`
- The definition of *v_c* input parameter is removed in the *for* loop due to it was useless in the solenoidality constraint. Also, we added the definition of it before the objective function to use it as the coefficient of *x_lkc* decision variable:
  `op.setInputParameter("v_c", netPlan.getVectorDemandOfferedTraffic(),"row");`
- The *v_c* parameter which was defined in the previous line is applied in the objective function:
  `op.setObjectiveFunction("minimize" , "sum (v_c*x_lkc)");`
- Modifications of the solenoidality constraint were done as well:

```
if (n == d.getIngressNode())
   op.addConstraint("sum(x_lkc(c,OutgoingLinks))-sum(x_lkc(c,IncomingLinks))== 1");
else if (n == d.getEgressNode())
   op.addConstraint("sum(x_lkc(c,OutgoingLinks))-sum(x_lkc(c,IncomingLinks))== -1");
else
   op.addConstraint("sum(x_lkc(c,OutgoingLinks))-sum(x_lkc(c,IncomingLinks))== 0");
```

- The capacity constraint related to the unsplittable FF is modified as follows (*v_c* is applied):
  `op.addConstraint ("sum(v_c*x_lkc(all, lk)) <= linkCapacity");`
- Line 64 was commented while lines 65 and 66 were uncommented. Due to exploiting the lines 65 and 66 we imported these two libraries:

```
import java.util.HashSet;
import java.util.Set;
```

It is worth noting that all the modifications are specified with // *** symbol as comment to find and read easily.

After running the algorithm on each capacity, we got the following result for splittable and unsplittable FFs.

Link Capacity [Gbps]	***Splittable***		***Unsplittable***	
	Objective Function Value	***Time***	***Objective Function Value***	***Time***
10	**No optimal solution**	**-**	**No optimal solution**	**-**
15	**174.140**	**0.123**	**No optimal solution**	**-**
17	**157.904**	**0.113**	**No optimal solution**	**-**
20	**155.514**	**0.104**	**189.403**	**0.108**
30	**155.514**	**0.103**	**155.514**	**0.104**
40	**155.514**	**0.098**	**155.514**	**0.102**
50	**155.514**	**0.097**	**155.514**	**0.098**

Table 2: Results of applying Splittable and Unsplittable FFs on the network topology

It is clearly seen that we got an optimal total traffic carried via the links for the Unsplittable FF when we have a high amount of capacity (For instance, 30 and 50). However, as we decreased the amount of link capacity, the congestion appeared (for 20 Gbps capacity) and even there is no optimal solution found for the capacities less than 20 Gbps. This was expected because in an unsplittable flow formulation, traffic demands cannot be divided or split; they must follow a single path from source to destination.

On the other hand, Splittable FF behaved more resiliently in correspondence with the decreasing of link capacity. It acted flexible with 17 to 50 Gbps link capacity and only with the least amount of capacity we checked in this test (10 Gbps) it did not find an optimal solution. This flexibility is resulted from the point that in a splittable FF, traffic demands can be split and routed across multiple paths or links simultaneously. This means that a single demand can utilize multiple paths in the network to reach its destination.

Finally, there is no difference between response time pattern of running the algorithm and obtaining the solution in Splittable and Unsplittable FFs. It is obvious that the algorithm finds the optimal solution more quickly as the amount of capacity increases.