TED Replication by Galera Software to compute 3-Shortest Path
Report #11

Maziar Sedghisaray (524923)
Maziar.sedghisaray2@gmail.com
Master in Computer Science and Networking (mcsn)

1. Introduction

This Software computes 3 shortest path from each source to each destination. I wrote this program with new features of C++11. I run some sample tests to check the correctness of results. For computing big topologies such as 10**100 node we can't run it on small machines due to time needed for computation and much more important, amount of Memory needed. I will try to parallelize the program and prepare it to run it on Intel Xeon Phi coprocessor with 60 cores on University of Pisa Informatica department.

2. Folders

- **0_Prepare_Input**
  ✓Includes 2 files written in PHP programming Language to prepare input data for KSP program.
  1) *selectLinks.php :* by running this program it will connect to Database and it will extract all Links in topology data to a file named ***Links***.
  2) *selectNodes.php:* by running this program it will connect to Database and it will extract all Nodes in topology data to file named ***Nodes***.

  Nodes and Links will be our input files to our KSP program.

- **1_KSP**

  ✓Includes KSP program C++ files, header files, executable version of program, and 2 sub folder.
  - ✓ ***Data****:* this is the place we put our input files extracted from database (Links and Nodes)
  - ✓ ***Final_Result:*** this is the place after computation finished, the final result file held in this folder.

- **3_Topology_Data**
  ✓Includes some sample topology data 3*3, 4*4, and 10*10

- **4_Sample_Results**
  ✓Includes results of the test which I got from sample topology data

3. Usage

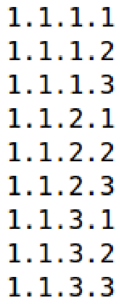
For running program we need to put Links and Nodes file in data folder and run the program. After computation finished we take the final result from Final_Result folder.

4. Sample run for 3*3 topology

A 3*3 topology includes 9 node and 24 Links.

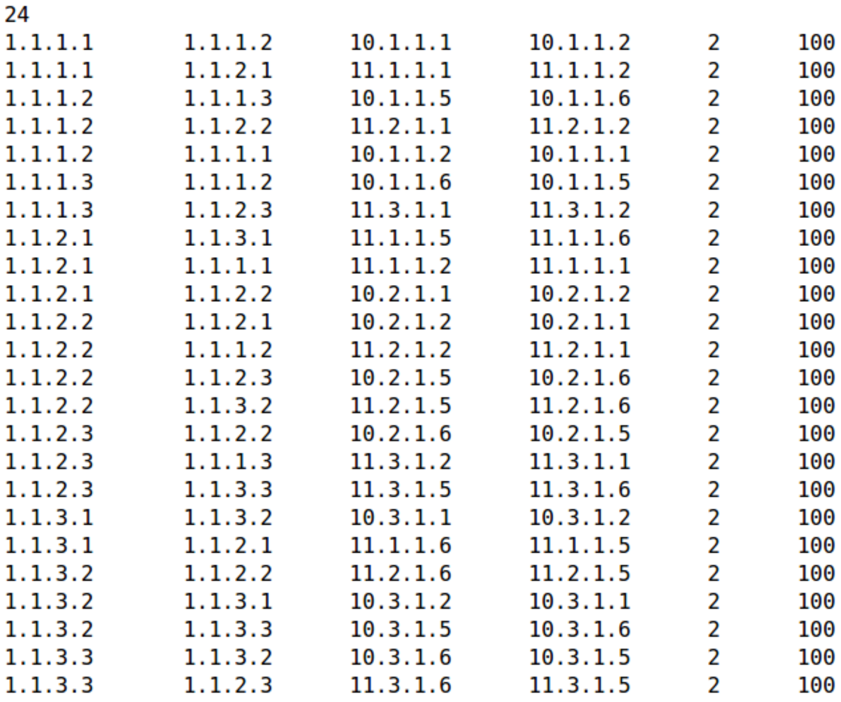
Nodes and Links files includes these information:

Nodes



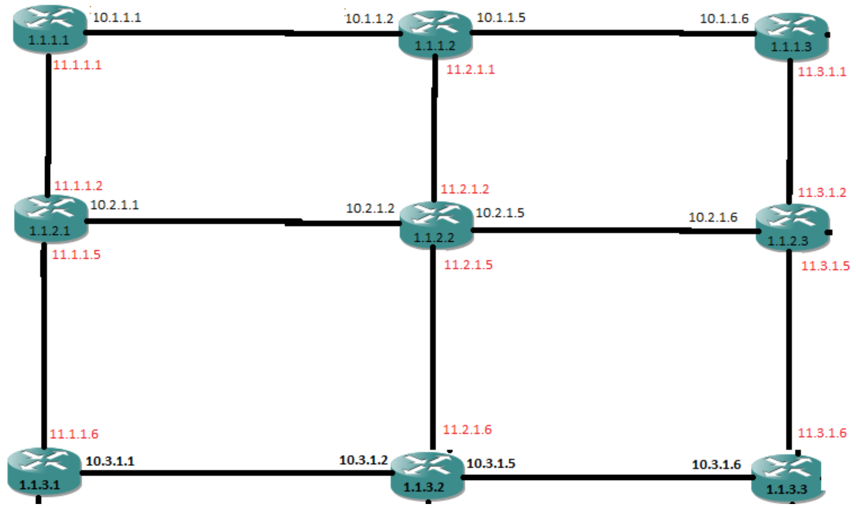
```
1.1.1.1
1.1.1.2
1.1.1.3
1.1.2.1
1.1.2.2
1.1.2.3
1.1.3.1
1.1.3.2
1.1.3.3
```

Links



```
24
1.1.1.1     1.1.1.2     10.1.1.1     10.1.1.2     2     100
1.1.1.1     1.1.2.1     11.1.1.1     11.1.1.2     2     100
1.1.1.2     1.1.1.3     10.1.1.5     10.1.1.6     2     100
1.1.1.2     1.1.2.2     11.2.1.1     11.2.1.2     2     100
1.1.1.2     1.1.1.1     10.1.1.2     10.1.1.1     2     100
1.1.1.3     1.1.1.2     10.1.1.6     10.1.1.5     2     100
1.1.1.3     1.1.2.3     11.3.1.1     11.3.1.2     2     100
1.1.2.1     1.1.3.1     11.1.1.5     11.1.1.6     2     100
1.1.2.1     1.1.1.1     11.1.1.2     11.1.1.1     2     100
1.1.2.1     1.1.2.2     10.2.1.1     10.2.1.2     2     100
1.1.2.2     1.1.2.1     10.2.1.2     10.2.1.1     2     100
1.1.2.2     1.1.1.2     11.2.1.2     11.2.1.1     2     100
1.1.2.2     1.1.2.3     10.2.1.5     10.2.1.6     2     100
1.1.2.2     1.1.3.2     11.2.1.5     11.2.1.6     2     100
1.1.2.3     1.1.2.2     10.2.1.6     10.2.1.5     2     100
1.1.2.3     1.1.1.3     11.3.1.2     11.3.1.1     2     100
1.1.2.3     1.1.3.3     11.3.1.5     11.3.1.6     2     100
1.1.3.1     1.1.3.2     10.3.1.1     10.3.1.2     2     100
1.1.3.1     1.1.2.1     11.1.1.6     11.1.1.5     2     100
1.1.3.2     1.1.2.2     11.2.1.6     11.2.1.5     2     100
1.1.3.2     1.1.3.1     10.3.1.2     10.3.1.1     2     100
1.1.3.2     1.1.3.3     10.3.1.5     10.3.1.6     2     100
1.1.3.3     1.1.3.2     10.3.1.6     10.3.1.5     2     100
1.1.3.3     1.1.2.3     11.3.1.6     11.3.1.5     2     100
```

The Topology for this example looks like:



After running the program, we will have a final result file which look like:

```
-1- 1.1.1.1-1.1.1.2   1.1.1.1   1.1.1.2   10.1.1.1   10.1.1.2   2   100
-2- 1.1.1.1-1.1.1.2   1.1.1.1   1.1.2.1   11.1.1.1   11.1.1.2   2   100
-2- 1.1.1.1-1.1.1.2   1.1.2.1   1.1.2.2   10.2.1.1   10.2.1.2   2   100
-2- 1.1.1.1-1.1.1.2   1.1.2.2   1.1.1.2   11.2.1.2   11.2.1.1   2   100
-3- 1.1.1.1-1.1.1.2   1.1.1.1   1.1.2.1   11.1.1.1   11.1.1.2   2   100
-3- 1.1.1.1-1.1.1.2   1.1.2.1   1.1.2.2   10.2.1.1   10.2.1.2   2   100
-3- 1.1.1.1-1.1.1.2   1.1.2.2   1.1.2.3   10.2.1.5   10.2.1.6   2   100
-3- 1.1.1.1-1.1.1.2   1.1.2.3   1.1.1.3   11.3.1.2   11.3.1.1   2   100
-3- 1.1.1.1-1.1.1.2   1.1.1.3   1.1.1.2   10.1.1.6   10.1.1.5   2   100
-1- 1.1.1.1-1.1.1.3   1.1.1.1   1.1.1.2   10.1.1.1   10.1.1.2   2   100
-1- 1.1.1.1-1.1.1.3   1.1.1.2   1.1.1.3   10.1.1.5   10.1.1.6   2   100
-2- 1.1.1.1-1.1.1.3   1.1.1.1   1.1.1.2   10.1.1.1   10.1.1.2   2   100
-2- 1.1.1.1-1.1.1.3   1.1.1.2   1.1.2.2   11.2.1.1   11.2.1.2   2   100
-2- 1.1.1.1-1.1.1.3   1.1.2.2   1.1.2.3   10.2.1.5   10.2.1.6   2   100
-2- 1.1.1.1-1.1.1.3   1.1.2.3   1.1.1.3   11.3.1.2   11.3.1.1   2   100
-3- 1.1.1.1-1.1.1.3   1.1.1.1   1.1.2.1   11.1.1.1   11.1.1.2   2   100
-3- 1.1.1.1-1.1.1.3   1.1.2.1   1.1.2.2   10.2.1.1   10.2.1.2   2   100
-3- 1.1.1.1-1.1.1.3   1.1.2.2   1.1.2.3   10.2.1.5   10.2.1.6   2   100
-3- 1.1.1.1-1.1.1.3   1.1.2.3   1.1.1.3   11.3.1.2   11.3.1.1   2   100
-1- 1.1.1.1-1.1.2.1   1.1.1.1   1.1.2.1   11.1.1.1   11.1.1.2   2   100
-2- 1.1.1.1-1.1.2.1   1.1.1.1   1.1.1.2   10.1.1.1   10.1.1.2   2   100
-2- 1.1.1.1-1.1.2.1   1.1.1.2   1.1.2.2   11.2.1.1   11.2.1.2   2   100
-2- 1.1.1.1-1.1.2.1   1.1.2.2   1.1.2.1   10.2.1.2   10.2.1.1   2   100
-3- 1.1.1.1-1.1.2.1   1.1.1.1   1.1.1.2   10.1.1.1   10.1.1.2   2   100
-3- 1.1.1.1-1.1.2.1   1.1.1.2   1.1.2.2   11.2.1.1   11.2.1.2   2   100
-3- 1.1.1.1-1.1.2.1   1.1.2.2   1.1.3.2   11.2.1.5   11.2.1.6   2   100
-3- 1.1.1.1-1.1.2.1   1.1.3.2   1.1.3.1   10.3.1.2   10.3.1.1   2   100
-3- 1.1.1.1-1.1.2.1   1.1.3.1   1.1.2.1   11.1.1.6   11.1.1.5   2   100
```

The first column indicates the path number, in this case is 1, 2, 3

-1- Indicates the first best path

-2- Indicates the second best path

-3- Indicates the third best path

The second column shows the path

1.1.1.1-1.1.1.2   Indicates path from Node 1.1.1.1 to node 1.1.1.2

The third and fourth column shows the intermediate nodes

1.1.1.1   1.1.1.2   Indicates Link used FROM node 1.1.1.1, TO node 1.1.1.2

The fifth and sixth column shows the interfaces between intermediate nodes

10.1.1.1   10.1.1.2   Indicates Link used FROM node 1.1.1.1, TO node 1.1.1.2 used by Interface 10.1.1.1 and 10.1.1.2.

The seventh and last column shows the TEmetric and Reservable Bandwidth between intermediate Links.

2 Indicates Traffic Engineering metric on this link is 2.

100 Indicates Reservable bandwidth on this link is 100.