

Documentation for CSCI 3650 Programming Assignment #2

Charlie Coleman

April 4, 2019

1 Documentation

1.1 Configuration to Network Description

Firstly, you need to create a configuration file for your network. Some examples are given in the `./configs/` directory, but if you'd like to make your own the format is as follows.

```
nodes: Int
topology: linear|full|star|random
alpha: Float
node-min: Int
node-max: Int
link-min: Int
link-max: Int
```

Once you have a properly formatted configuration file, you can execute the code with the following command:

```
python3 part1.py <file in> <file out>
```

Only the input file name is required. If `file out` is left out, it will default to the name of the input with `.out` as the extension. This code will output a file of the format:

```
Source-Node-ID Destination Node-ID Link0-weight
Source-Node-ID Destination Node-ID Link1-weight
...
Node0-weight Node1-weight... NodeN-weight
```

1.2 Network Description to Virtual Network

Once we have a network description file from `part1.py`, we can pass it to `part2.py` to get create our virtual network. To run `part2.py`, use the following command.

```
python3 part2.py <file in>
```

This will parse the network description file you specify and create a virtual network that matches that topology.

2 Code

Program 1: ../part1.py

```
1 import sys, random, os
2
3 fInName = "generic.conf"
4 if (( len(sys.argv)-1) == 0):
5     raise Exception("Improper format: python3 part1.py <file in> <file out>")
6 elif (( len(sys.argv)-1) >= 1):
7     fInName = sys.argv[1]
8
9 fOutName = os.path.splitext(fInName)[0]+".out"
10 if (( len(sys.argv)-1) >= 2):
11     fOutName = sys.argv[2]
12
13 fIn = open(fInName, 'r')
14 fOut = open(fOutName, 'w')
15
16 def connections(node, arr, orig ):
17     conns = []
18     addConns = [node]
19     orig.append(node)
20     for i in range(0,nodes):
21         if (arr[node][i] and i != node and i not in orig):
22             conns.append(i)
23     for conn in conns:
24         newConns = connections(conn, arr, orig)
25         if newConns:
26             addConns.extend(newConns)
27     conns.extend(addConns)
28     conns.sort()
29     return list( dict.fromkeys(conns))
30
31 def randStrInt(minVal,maxVal):
32     return str(random.randint(minVal,maxVal))
33
34 def printAdjMatrix(arr):
35     [print("\t".join([ str(arr[i][j]) for j in range(0,nodes)])) for i in
36         range(0,nodes)]
37
38 for line in fIn.read().splitlines():
39     param = [x.strip() for x in line.upper().split(':')]
40     if (param[0] == "NODES"):
41         nodes = int(param[1])
42     elif (param[0] == "TOPOLOGY"):
43         topology = param[1]
44     elif (param[0] == "ALPHA"):
45         alpha = float(param[1])
46     elif (param[0] == "NODE-MIN"):
47         nodeMin = int(param[1])
48     elif (param[0] == "NODE-MAX"):
49         nodeMax = int(param[1])
50     elif (param[0] == "LINK-MIN"):
51         linkMin = int(param[1])
52     elif (param[0] == "LINK-MAX"):
53         linkMax = int(param[1])
54
55 nodeWeights = [randStrInt(nodeMin, nodeMax) for i in range(0, nodes)]
```

```

55
56 if (topology == "LINEAR"):
57     for i in range(0, nodes-1):
58         line = str(i) + "\t" +
59             str(i+1) + "\t" + randStrInt(linkMin, linkMax)+"\n"
60         fOut.write(line)
61     fOut.write("\t".join(nodeWeights) + "\n")
62 if (topology == "FULL"):
63     for i in range(0, nodes-1):
64         for j in range(i+1, nodes):
65             line = str(i) + "\t" +
66                 str(j)+"\t" + randStrInt(linkMin, linkMax)+"\n"
67             fOut.write(line)
68         fOut.write("\t".join(nodeWeights) + "\n")
69 if (topology == "STAR"):
70     for i in range(1, nodes):
71         line = "0\t" + str(i) + "\t" + randStrInt(linkMin, linkMax)+"\n"
72         fOut.write(line)
73     fOut.write("\t".join(nodeWeights) + "\n")
74 if (topology == "RANDOM"):
75     arr = [[False for i in range(0, nodes)] for i in range(0, nodes)]
76     x = 0
77     while True:
78         for i in range(0, nodes):
79             for j in range(i+1, nodes):
80                 arr[i][j] = arr[j][i] = (random.random() < alpha)
81                 arr[i][i] = True
82                 if (connections(0,arr,[])) == [i for i in range(0, nodes)]:
83                     break
84             x+=1
85             if (x > 1000):
86                 print("ERROR: Could not generate a connected graph. Perhaps choose a
87                     different alpha or number of nodes?\n")
88                 break
89     for i in range(0,nodes-1):
90         for j in range(i+1,nodes):
91             if (arr[i][j]):
92                 line = str(i) + "\t" +
93                     str(j) + "\t" + randStrInt(linkMin,linkMax) + "\n"
94                 fOut.write(line)
95     fOut.write("\t".join(nodeWeights) + "\n")

```

Program 2: ../part2.py

```

1 from mininet.topo import Topo
2 from mininet.net import Mininet
3 import sys
4
5 filename = sys.argv[1]
6 fIn = open(filename, 'r')
7
8 conns = [line.split() for line in fIn.readlines()]
9 weights = conns[-1]
10 conns = conns[0:-1]
11 hosts = [0 for weight in weights]
12
13 class MyTopo(Topo):
14     def build(self):
15         for i in range(0, len(weights)):
16             hosts[i] = self.addHost( "h%s" % i )

```

```
17         for conn in conns:
18             self.addLink(hosts[ int(conn[0])], hosts[ int(conn[1])])
19
20 topo = MyTopo()
```