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
```

These parameters can be in any order and are not case sensitive. Alpha is optional for all topologies except random.

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

```
import sys, random, os

fInName = "generic.conf"
if (( len(sys.argv)-1) == 0):
    raise Exception("Improper format: python3 part1.py <file in> <file out>")
elif (( len(sys.argv)-1) >= 1):
    fInName = sys.argv[1]

fOutName = os.path.splitext(fInName)[0]+".out"
if (( len(sys.argv)-1) >= 2):
    fOutName = sys.argv[2]

fIn =  open(fInName, ’r’)
fOut =  open(fOutName, ’w’)

def connections(node, arr, orig ):
    conns = []
    addConns = [node]
    orig.append(node)
    for i in  range(0,nodes):
        if (arr[node][i] and i != node and i not in orig):
            conns.append(i)
    for conn in conns:
        newConns = connections(conn, arr, orig)
        if newConns:
            addConns.extend(newConns)
    conns.extend(addConns)
    conns.sort()
    return  list( dict.fromkeys(conns))

def randStrInt(minVal,maxVal):
    return  str(random.randint(minVal,maxVal))

def printAdjMatrix(arr):
    [print("\t".join([ str(arr[i][j]) for j in  range(0,nodes)])) for i in range(0,nodes)]

for line in fIn.read().splitlines():
    param = [x.strip() for x in line.upper().split(’:’)]
    if (param[0] == "NODES"):
        nodes =  int(param[1])
    elif (param[0] == "TOPOLOGY"):
        topology = param[1]
    elif (param[0] == "ALPHA"):
        alpha =  float(param[1])
    elif (param[0] == "NODE-MIN"):
        nodeMin =  int(param[1])
    elif (param[0] == "NODE-MAX"):
        nodeMax =  int(param[1])
    elif (param[0] == "LINK-MIN"):
        linkMin =  int(param[1])
    elif (param[0] == "LINK-MAX"):
        linkMax =  int(param[1])

nodeWeights = [randStrInt(nodeMin, nodeMax) for i in  range(0, nodes)]
```

```

if (topology == "LINEAR"):
    for i in range(0, nodes-1):
        line = str(i) + "\t" +
            str(i+1) + "\t" + randStrInt(linkMin, linkMax)+"\n"
        fOut.write(line)
    fOut.write("\t".join(nodeWeights) + "\n")
if (topology == "FULL"):
    for i in range(0, nodes-1):
        for j in range(i+1, nodes):
            line = str(i) + "\t" +
                str(j)+"\t" + randStrInt(linkMin, linkMax)+"\n"
            fOut.write(line)
    fOut.write("\t".join(nodeWeights) + "\n")
if (topology == "STAR"):
    for i in range(1, nodes):
        line = "0\t" + str(i) + "\t" + randStrInt(linkMin, linkMax)+"\n"
        fOut.write(line)
    fOut.write("\t".join(nodeWeights) + "\n")
if (topology == "RANDOM"):
    arr = [[False for i in range(0, nodes)] for i in range(0, nodes)]
    x = 0
    while True:
        for i in range(0, nodes):
            for j in range(i+1, nodes):
                arr[i][j] = arr[j][i] = (random.random() < alpha)
            arr[i][i] = True
        if (connections(0,arr,[]) == [i for i in range(0, nodes)]):
            break
        x+=1
        if (x > 1000):
            print("ERROR: Could not generate a connected graph. Perhaps choose a
                different alpha or number of nodes?\n")
            break
    for i in range(0,nodes-1):
        for j in range(i+1,nodes):
            if (arr[i][j]):
                line = str(i) + "\t" +
                    str(j) + "\t" + randStrInt(linkMin,linkMax) + "\n"
                fOut.write(line)
    fOut.write("\t".join(nodeWeights) + "\n")
```

Program 2: ../part2.py

```
from mininet.topo import Topo
from mininet.net import Mininet
import sys

filename = sys.argv[1]
fIn = open(filename, 'r')

conns = [line.split() for line in fIn.readlines()]
weights = conns[-1]
conns = conns[0:-1]
hosts = [0 for weight in weights]

class MyTopo(Topo):
    def build(self):
        for i in range(0, len(weights)):
            hosts[i] = self.addHost( "h%s" % i )
        for conn in conns:
            self.addLink(hosts[int(conn[0])], hosts[int(conn[1])])

topo = MyTopo()
```