NETWORK LAYER SIMULATOR

Computer Networks Project

Team

Abhinav Gupta (120050029)
Anant Gupta (120050086)
Bijoy Singh (120050087)

Objectives Of The Project

- Create a Network Layer Simulator to assist visualization of various Routing and Forwarding algorithms.
- Allow the users to visualise the drawbacks of various algorithms
- Allow non programming users to be able to run the simulator without any previous knowledge of programming or routing
- Create a library for programs to use and run the simulator

The Project

We have created a Python based Network Layer Simulator package. This is a stand alone package that be imported and run to view/create or edit networks, create application layer programs and more, as stated below:

Protocols

- IP Forwarding and IP Forwarding Tables
- IP Routing using Distance Vector Routing
- IP Routing using Link State Routing
- Trace Route over UDP
- ICMP like Error Messages

Features

- Reliable UDP with Port Demultiplexing
- Barebone application class to build any application on Host ports
- See and create issues like Count to infinity
- Fully monitor and edit the network during the simulation
- Create the network completely on the UI and see it work
- Create the network and simulate from code by importing our python package
- Time Driven Simulation to step through the simulation
- View state of every node and connection during the simulation
- Create/Remove hosts/routers/connections during the simulation
- Call TraceRoute during the simulation

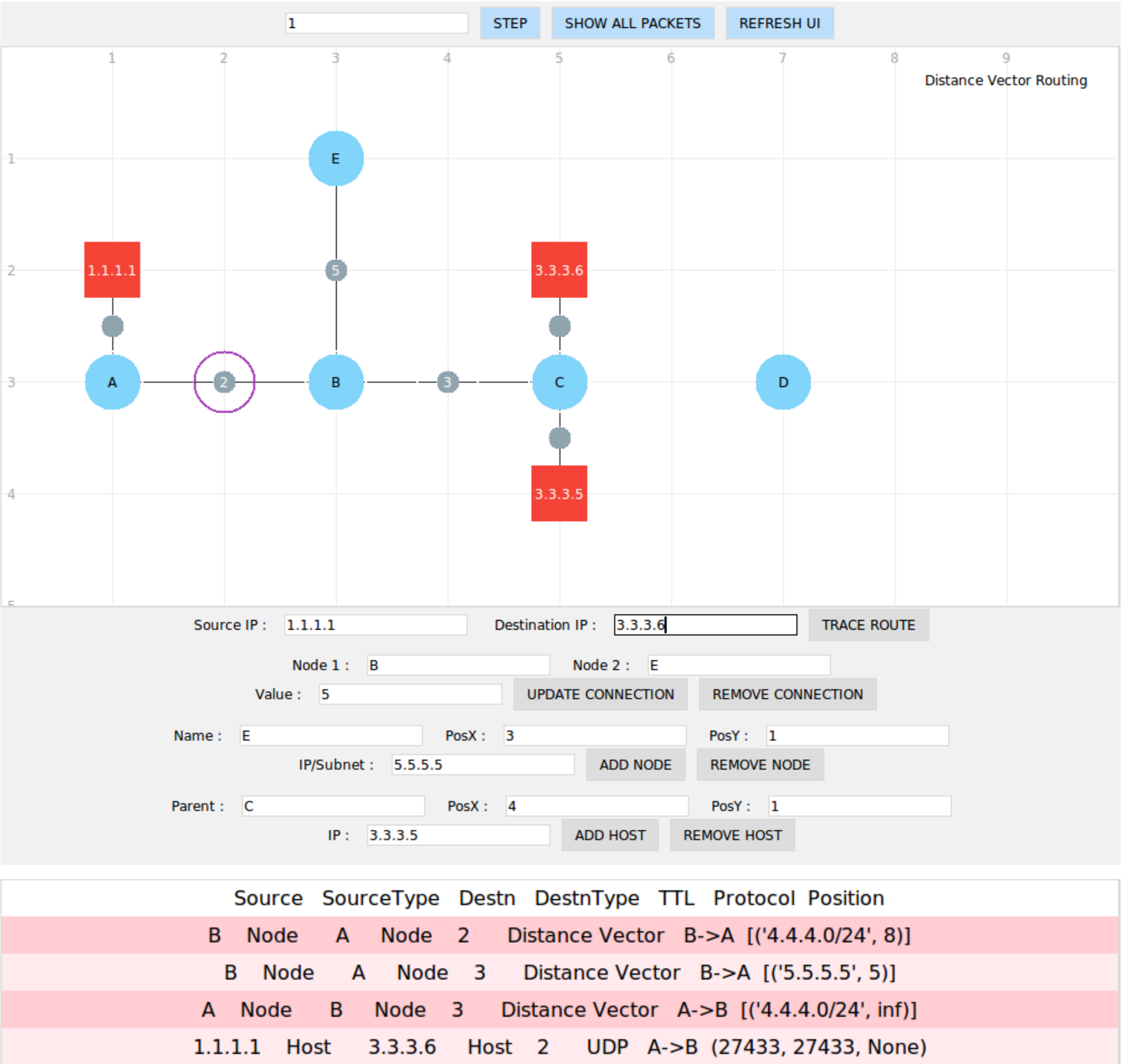
Usage

Dependencies (Compatible with all Operating Systems)

- Python3 - https://www.python.org/downloads/
- Tkinter Library for Python 3 - https://docs.python.org/3/library/tkinter.html
- Our Network Simulator Library - https://github.com/Abhinav2107/cs378/

```
-------------------------------------------------------------------------
from NetworksProject import *  # Import everything

sim = Simulator.Simulator()  # Create a simulator object

sim.set_routing_protocol(("Distance Vector", False, False,10))  #  Set routing
protocol, with options for split horizon and poison reverse and mention periodic update
frequency

sim.add_node("n1", (1, 1), "1.1.1.0/24")  #  Add a router node. Arguments are name of
the node, position and ip prefix handled
sim.add_node("n2", (3, 1), "2.2.2.0/24")
sim.add_node("n3", (5, 1), "3.3.3.0/24")

sim.add_connection("n1", "n2", 1)  #  Add a connection between two nodes with some cost
sim.add_connection("n2", "n3", 1)

h = sim.add_host("1.1.1.1", "n1", (2, 2))  #  Add a host to a particular node along
with it's position
sim.add_host("3.3.3.3", "n3", (5, 3))

packet = Packet.Packet("Host", "Host", "1.1.1.1", "3.3.3.3", "1.1.1.1", "n1", "UDP",
64, 1, "Package Data")  #  Create a new packet with arguments as follows

#  src_type, dst_type, src, dst, link_src, link_dst, protocol, ttl, cost, data

sim.step()  #  Step the simulation

sim.put_packet(packet)  #  Send a packet

tr = TraceRoute.TraceRoute(h, "TraceRoute")  #  Create a TraceRoute application on host
h. Name the application TraceRoute
tr.trace("3.3.3.3")  #  Trace path to a certain host

sim.update_connection("n2", "n3", float("+inf"))  #  Update the cost of the connection.
A cost higher than routing protocol's infinity value breaks the connection.

gui = SimulatorPlotter.Gui(sim)  #  Create a GUI object
gui.start()  #  Start GUI
-------------------------------------------------------------------------
```

User Interface



1

Source IP : 1.1.1.1 Destination IP : 3.3.3.6

Node 1 : B Node 2 : E

Value : 5

Name : E PosX : 3 PosY : 1

IP/Subnet : 5.5.5.5

Parent : C PosX : 4 PosY : 1

IP : 3.3.3.5

Source	SourceType	Destn	DestnType	TTL	Protocol	Position	
B	Node	A	Node	2	Distance Vector	B->A	[('4.4.4.0/24', 8)]
B	Node	A	Node	3	Distance Vector	B->A	[('5.5.5.5', 5)]
A	Node	B	Node	3	Distance Vector	A->B	[('4.4.4.0/24', inf)]
1.1.1.1	Host	3.3.3.6	Host	2	UDP	A->B	(27433, 27433, None)

Future Developments

- Being open source and on python development on the project is easy and can be collaborative.
- As we have made a class structure, the various application layer protocols, or transport layer protocols like TCP,SSH etc can be developed and simulated
- The same code due to abstract coding will allow for developing different routing algorithms run on the same code without too much coding overhead like OSPF,BGP,etc
- Being a simple program, can be used to teach and demonstrate the various routing algorithms and their failure