PROJECT REPORT

CS 6390: Advanced Computer Networks
Fall 2013 - Semester Project
SCALE-FREE TOPOLOGY GENERATION
Instructor: Dr Kamil Sarac
Teaching Assistant: Emrah Cem

Project Members:
Ganesh Salvi - gxs120030
Shreyas Gokhale - sxg122830
Sneha Patil - sxp127030

Project Implementation:

The project is implemented in Java.

Compiling and Executing the Program:

The source code is contained in total 10 files. The function main() is in file Node.java

- **Command To compile the project:** javac *.java
- **Command To run the project:** java Node <node_id>

Example: if m=1, m0=2, n=4

Each node is opened on a new terminal (any machine between net01-net45.utdallas.edu, node_id hostname should match with the one written in the config.txt file)

Enter :
java Node 0 ... on 1st terminal
java Node 1 ... on 2nd terminal
java Node 2 ... on 3rd terminal (this would act as new node)

- The user would be asked for input, if he wants to perform UDP queries or send a Join request.

Enter Query or Join as required.

- Choose the appropriate choices from the displayed menu if UDP queries are to be performed.

- If new node willing to join exceeds n, Reject message will be displayed.

Project Design:

We have maintained a configuration file config.txt which contains information of m, m0,n and node id, hostname, TCP port, UDP port of m0 nodes. Each node has the same value of TCP Port number. Also, each node has the same value of UDP Port number. This does not cause conflicts because they all run on different machines. Hostname can be between net01-net45.utdallas.edu. m<=m0<<n

Assumption: Initial topology is STAR topology

Let m0=Initial number of nodes
n=Maximum desired number of nodes
m=number of links each node can have

Format of config.txt file is as follows:
m m0 n
Node_id Hostname TCP_port UDP_port

Sample config.txt :
1 3 5
0 net01.utdallas.edu 5050 9090
1 net02.utdallas.edu 5050 9090
2 net03.utdallas.edu 5050 9090

- We have implemented Barabasi-Albert model which starts with a connected network of m0-1 nodes with no links and then m0th node is connected(moth will make connections) to all other nodes.

- First nodes will listen on a UDP port for the Queries and Routing table request from the new node/s before they join in the network..

- New node will randomly choose one of the m0 nodes and send a request to join the network from the UDP port mentioned in the config file.

- Upon receiving a request, if network size is less than , the node (one of the m0 nodes who received the request) will reply with routing table information (Destination host name, Next Hop, Cost, Degree)and new node selects m of them based on the Barabasi-Albert model.

- The idea of preferential attachment is implemented such that the node with the highest degree has a high probability of getting chosen , to which the new node will connect to. We have, created an array with size = sum of degrees of all nodes & simultaneously stored the hostnames in the array. The hostnames are stored those many number of times equal to the degree of that node.

Data structures Used:

- **HashMap for tcp sockets.**
  - Key – client's hostname
  - Value - client's Socket
- **ArrayList to store each m0 host details.**
  - Each host has following attribute :- host Id, TCP port, UDP port, Hostname
- **Message Class-** with Sender, Receiver, Message content, Routing Table as elements
- **DistanceVector class has class attributes :-** Destination, Next hop, Cost, Degree
- **HashMap for routing table**
  - Key - hostname
  - Value - DistanceVector class object
  - DistanceVector class has class attributes :- Destination, Next hop, Cost, Degree

Overall Program Flow/ Pseudo Code:

1. Read config.txt file
2. Get values of m, m0, n
3. Initialize routing table. (Put your own entry in the routing Table)
4. Build initial topology- Initial topology is STAR Graph
5. If node belongs to m0 & node != m0th node,
    a. Start node & its threads [TcpSend, TcpReceive, UDPListener, TcpAccept]
6. else (i.e. node belongs to m0 & node == m0th node)
    a. Start node & its threads [TcpSend, TcpReceive, UDPListener, TcpAccept]
    b. and Make connections with m0-1 nodes
7. If the connections are made successfully, trigger the distance vector protocol.
8. // For new node , if JOIN_REQ sent
9. If node does not belong to m0,
    a. Get any node from m0 (randomly) and query its UDP port for routing table information
10. If node queried sends ‘REJECT’ message, which means there are total n nodes in the topology.
    a. Display message 'The Topology has reached the maximum limit for nodes...You are rejected !!' and ask user input if he wants to continue with UDP queries
    b. If not Rejected, proceed ahead:
11. If node queried sends ‘RoutingTable’ .After getting this info, store the information.
12. //Choosing Nodes to connect to
    a. Create array with size = sum of degrees of all nodes & simultaneously start storing the machine names in the array. Store the machine names those many number of times equal to the degree of that node.
13. Let A=Array to store the nodes to be connected

Now for up to ‘m’ number of edges for each incoming new node,

    a. Pick a random element from the created array and connect the new node to it.
    b. Put the selected element in an array of length equal to ‘m’.
    c. For each selected element, check if it already exists in the new array 'A'.
    d. If not, add it in.
    e. If yes, repeat until you get an element not present in the array

14. Connect to all m nodes using Array A.
15. Once the connections are made successfully, trigger the distance vector protocol.
16. //Running the 3 queries over UDP port
    a. The transfer of the current routing table,
    b. Sending the node degree information, and
    c. Sending the IDs of the farthest away nodes in the system.

User is asked for query type and hostname of the node to be queried.
UDPListener thread is run on each machine to listen to incoming UDP queries.

Format of exchanged messages:

Following are the attributes and their values shared in each message:-

Query Type corresponds to query sent by the node to another node.
Message Type corresponds to the reply/response sent by the other node.

Query Type	Message Type	Message Sender	Message Content	Routing Table
QUERY_ROUTING_TABLE or JOIN_REQ (network has less than n nodes)	RoutingTable	Sender hostname	Hello message	updated routing table
JOIN_REQ (network currently has n nodes)	REJECT	Sender hostname	Reject message	null
QUERY_DEGREE	DegreeInfo	Sender hostname	Degree of queried node	null
QUERY_FARTHEST_NODES	FarthestNode	Sender hostname	Farthest node from queried node	null

Following are the attributes associated with each query:-

Query Type	Query to Node	UDP port

Following are the attributes and their values shared in each routing table, values vary depending on the node sending the routing table details :-

Example :
Queried node is - net31.utdallas.edu if its among the m0 nodes, and not the m0th node. Hence it currently has no links.

Destination	Next Hop	Cost	Degree
net31.utdallas.edu	net31.utdallas.edu	0	0

Now consider the case when m0 nodes are connected, new node is trying to join .Network has less than n nodes

Config.txt is :
1 2 3
0 net41.utdallas.edu 5050 9090
1 net43.utdallas.edu 5050 9090

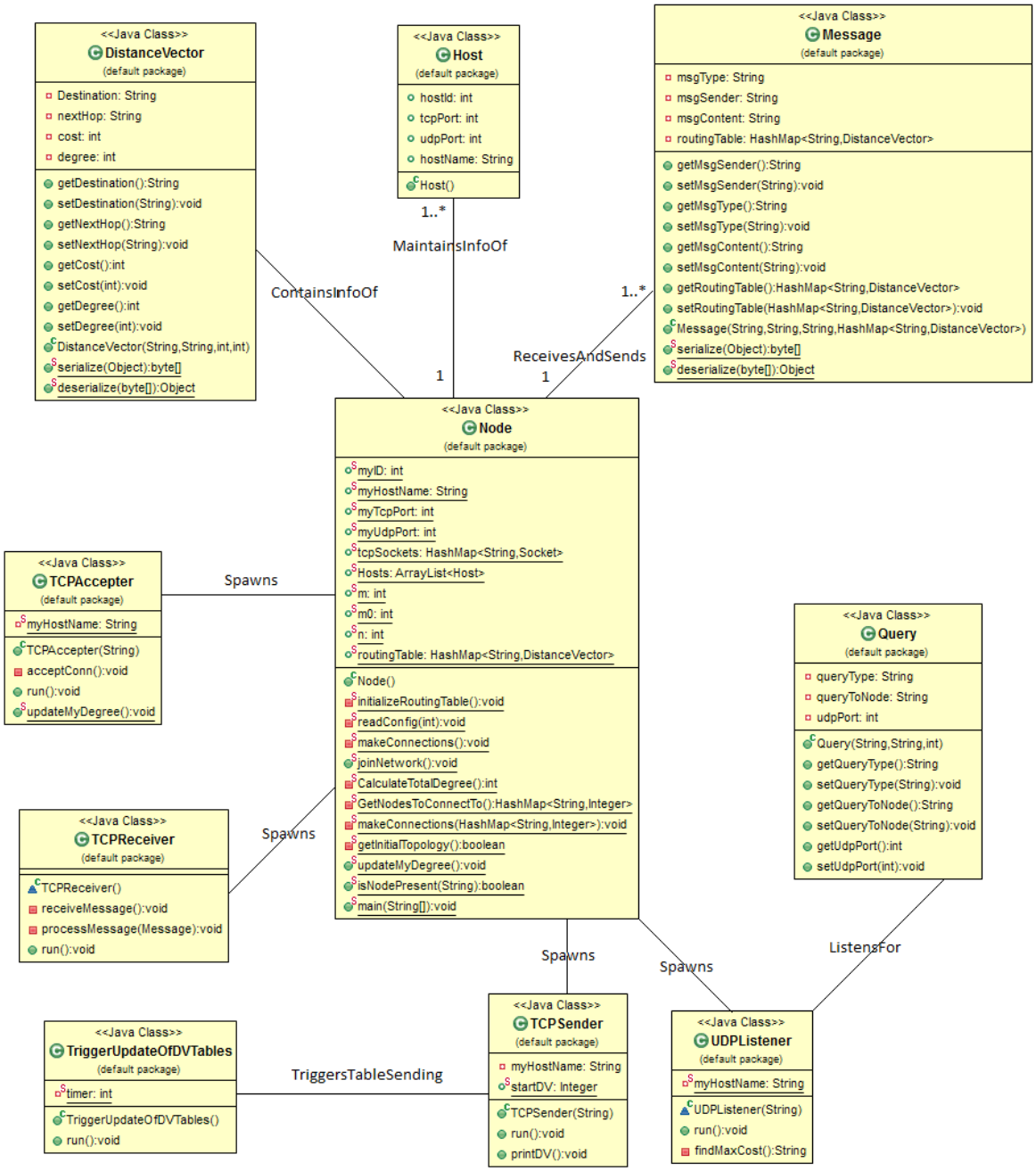
Let Host name of new node:- net26.utdallas.edu

Let after computation of preferential attachment ,new node connects to:- net43.utdallas.edu

Routing table at node net26.utdallas.edu(new node) after it has joined the network:-

Destination	**Next Hop**	**Cost**	**Degree**
net41.utdallas.edu	net43.utdallas.edu	2	1
net43.utdallas.edu	net43.utdallas.edu	1	2
net26.utdallas.edu	net26.utdallas.edu	0	1

Class Diagram:



Message/time complexity of building the whole topology:

Message Complexity:

We have considered STAR topology as initial topology.

- **For Initial Configuration: (with m0 initial nodes)**
  m0-1 Messages will be exchanged. As we are using TCP for making connections and TCP uses 3 way handshake so TCP messages exchanged O(3*(m0-1)).
- **For UDP queries:**
  Total two messages for each request: One request and one Response message.
Message complexity=O(2)
- **For Joining in the network:**
  Two Messages (one request and one response message) to get the Routing table.
m messages to connect to the m nodes.
Message complexity=O(3*m0)+O(2) but given that m0 is a constant, overall Message complexity is constant (i.e. O(1)).

Time Complexity:

Let T=Message Transmission time
We have considered STAR topology as initial topology.

- **For Initial Configuration: (with m0 initial nodes)**
  m0-1 Messages will be exchanged. As we are using TCP for making connections and TCP uses 3 way handshake so TCP messages exchanged O(3*(m0-1))
Hence Time Complexity=O(3*(m0-1)*T)
- **For UDP queries:**
  Total two messages for each request: One request and one Response message
Time Complexity=O(2T) of each request.
- **For Joining in the network:**
  Two Messages (one request and one response message) to get the Routing table.
m messages to connect to the m nodes.
Time complexity=O(3*m0*T)+O(2T) where T = message transmission time

Tasks/Role performed by each team Members:

The Project implementation/success is a collaborative efforts of the all the team members. We followed the modularized approach and made the code modules as independent of each other as possible during the design which lead to the smooth integration of the modules in the end.

MODULE	DESIGN	CODING	DEBUGGING/TESTING
Overall Flow and High Level Design	Ganesh Salvi	NA	NA
Initial Topology Formation	Shreyas Gokhale	Sneha Patil	Everyone
TCP Connections	Ganesh Salvi	Sneha Patil	Everyone
Distance Vector Protocol	Sneha Patil	Ganesh Salvi	Everyone
Barabasi-Albert Model	Shreyas Gokhale/Ganesh Salvi	Shreyas Gokhale	Everyone
UDP Queries	Sneha Patil	Sneha Patil	Everyone
Joining of Node	Ganesh Salvi	Shreyas Gokhale	Everyone
Triggering the Distance vector Periodically and on change of topology	Ganesh Salvi	Ganesh Salvi	Everyone
Updating degree on Topology Change	Sneha Patil	Shreyas Gokhale	Everyone

Apart from above tasks the integration, technical documentation and the testing of the project is carried out by everyone together.