CS 6390: Advanced Computer Networks

Fall 2013 - Semester Project

SCALE-FREE TOPOLOGY GENERATION

Instructor: Dr Kamil Sarac

Due: December 8, 11:59 pm

1 Requirements

Source code must be in the *C/C++ or Java* programming language.

2 Overview

In this project, you will construct a *scale-free* network which is known to have ultra-small diameter ($\sim\ln\ln(n)$), where n is the network size. Diameter of a network is defined as the largest shortest path length between all pairs of nodes in the network. The link between two nodes in the network will correspond to a TCP connection. Your main task is to find a scalable way of creating TCP connections between nodes using *Barabasi-Albert* model. Before going into details of the project, let's first describe scale-free networks and how they are generated using Barabasi-Albert model.

2.1 SCALE-FREE NETWORKS

WHAT IS A SCALE-FREE NETWORK?

Let $G = (V, E)$ be a network with node set V and link set E. The order of G, i.e., the number of its nodes, is denoted by $|V|$, the number of its links by $|E|$. The degree of a node $v \in V$, i.e., number of neighbors of v in G is denoted by $d(v)$. Degree distribution $P(k)$ of G is defined as the probability distribution of degrees of nodes in G. If the degree distribution of G has the form $P(k) = ak^x$, where a and x are constants, then G is called a scale-free network. Some well-known properties of scale-free networks are: i) existence of hubs, nodes with a degree much higher than the average, ii) resilience to random failures, iii) and ultra-small diameter.

WHY ARE THEY IMPORTANT?

Scale-free networks are very common in various domains including social friendship networks, protein-protein interaction networks, power grids, neural networks, citation networks, WWW, AS-Level internet topology network, and many more. Hence, understanding the properties of scale-free networks such as resilience to various types of failures and temporal change may help to understand various phenomena in different disciplines.

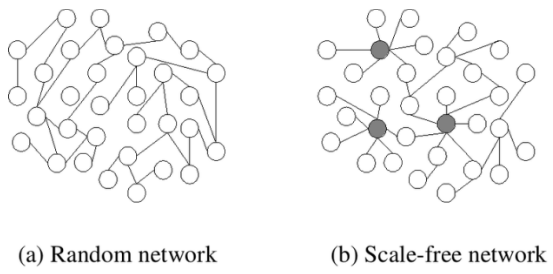


Figure 1 - In the scale-free network, the hubs are highlighted. (http://en.wikipedia.org/wiki/scale-free_network)

HOW CAN THEY BE GENERATED?

There are different models to generate a scale-free network. One of the well-known models that produce scale-free networks is named Barabasi-Albert model. This model uses growth and preferential attachment mechanisms to generate the network. The growth means that the network is formed by adding nodes to the existing network, so it is not the case that all nodes are created first and then links are created between them. Barabasi-Albert model starts with a connected network of m_0 nodes, i.e., $m_0 - 1$ nodes are added with no links and the ${m_0}^{th}$ node is connected to all other nodes. There is also another version which initially creates a complete network of m_0 nodes. You can choose any version as long as it is connected. Then, a new node is added to the network with a fixed number of links m, until the desired number of nodes n is reached. The following link illustrates the animation of a construction of a scale-free network with Barabasi-Albert model with $m = m_0 = 2, n = 20$. http://en.wikipedia.org/wiki/File:Barabasi_Albert_model.gif . When a new node is added to the network, to which nodes it will be connected is where the concept of preferential attachment comes into play. Basic idea of preferential attachment model is that the higher the degree of a node, the *more likely* it is to receive new links. More formally, the probability $P(v)$ that a new node connects to node v is given by

$$P(v) = \frac{d(v)}{\sum_{u \in V} d(u)}$$

Note that using this model, requires a global knowledge, i.e., sum of degrees of all nodes in the network. This might be easy in a simulation environment where you can get the degree of each node by a simple query. In real world, however, this knowledge is either not available or costly to compute.

3 Your task

Your task is to create links (TCP connections) between nodes in such a way that network will mimic the Barabasi-Albert model. The challenge is that new node that attempts to connect to the network does not have the global knowledge of IP address and port number of all existing nodes. Assume that the nodes initially know only IP address and UDP port numbers of first m_0 nodes. You are asked to design a *scalable* mechanism, with this limited knowledge, which will mimic Barabasi-Albert model to create a TCP connection between nodes. After a node is added to the network, i.e., created m TCP connections with other nodes, it will need to compute shortest paths to all other nodes. [Hint: You can implement a distance vector protocol. Can you also make use of this protocol to compute degrees of other nodes?] For testing purposes, each node will need to support some simple querying over a dedicated UDP port. The supported queries should include (1) the transfer of the current routing table, (2) sending the node degree information, and (3) sending the IDs of the farthest away nodes in the system.

4 Implementation details

4.1. BUILDING THE TOPOLOGY

- You are not supposed to handle node exit and node failures during topology construction
- $m \leq m_0 \ll n$
- First m_0 nodes will listen from a UDP port for the new nodes. You can assume that all nodes know the IP addresses and UDP ports at which first m_0 nodes listen. First m_0 nodes also know the parameter n.
- New node will randomly choose one of m_0 nodes and send a request to join the network from the UDP port mentioned in 2. Upon receiving a request, if network size is less than n, the node (one of the m_0 nodes who received the request) will reply with m (IP address, TCP port) tuples from the network based on the Barabasi-Albert model; otherwise it will send a reject message. (Another option is that the node will reply with all (IP address, TCP port) tuples as well as degree information of all the nodes in the network and new node selects m of them based on the Barabasi-Albert model.)
- When a new node sends a join request to one of m_0 nodes, it should be able respond immediately without communicating any other nodes. (This requirement is satisfied if you implement distance vector protocol and include degree information in it.)
- After deciding m nodes to connect to, new node will create links (TCP connections) with those m nodes. Then, it will start listening for TCP connections since new nodes can connect to it.
- Note that link creation is a stochastic process. New node does not necessarily connect to the m highest degree nodes in the network; high degree nodes have higher chances to be connected than lower degree nodes.
- $m \leq m_0 \ll n$

4.3 MAINTAINING TOPOLOGY INFORMATION (DISTANCE VECTOR PROTOCOL)

- Each node should maintain a routing table and send it to its neighbors periodically.
- You don’t need to consider the count-to-infinity problem.

4.2 ANSWERING QUERIES

- After joining the network, each node should be ready to answer queries from a dedicated UDP port 9090. The format of the query messages is up to you, but you are required to explain it in your REPORT file.

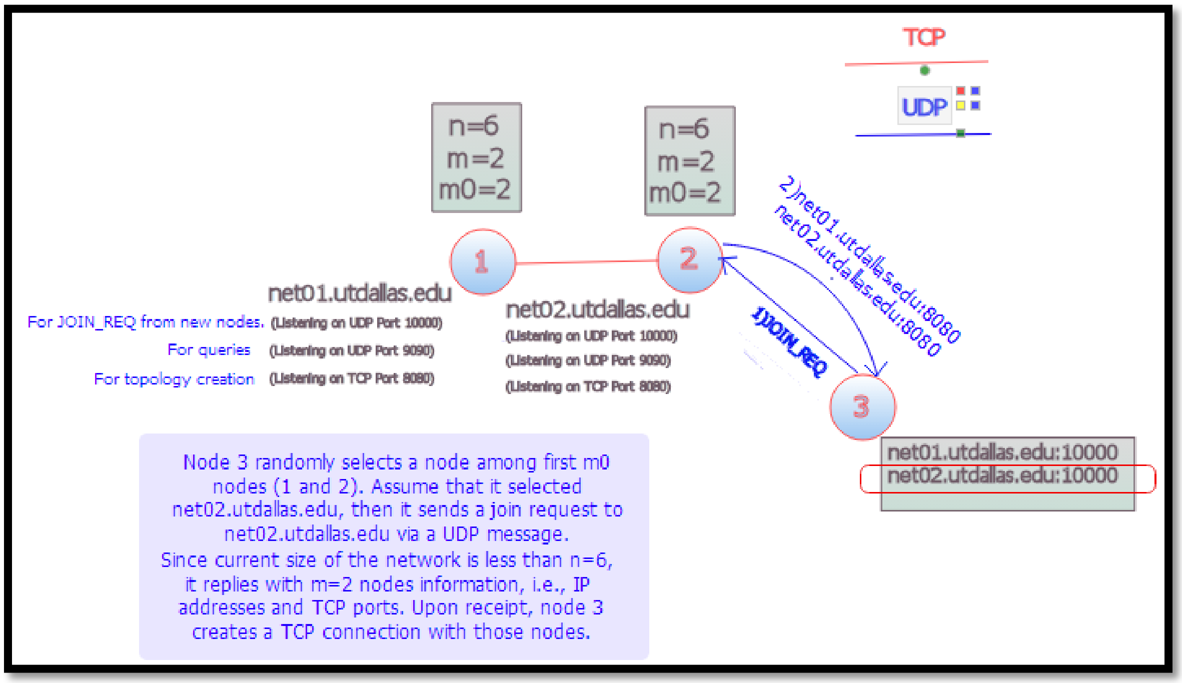
5 What to submit

The submission should be through *eLearning* in the form of an archive consisting of:

1. All files of the source code of your program and a REPORT file as a <netid>.zip file.
2. The REPORT file describes how to compile and run your program, as well as your design of solution including the pseudocode, formats of exchanged messages, and message /time complexity of building the whole topology.

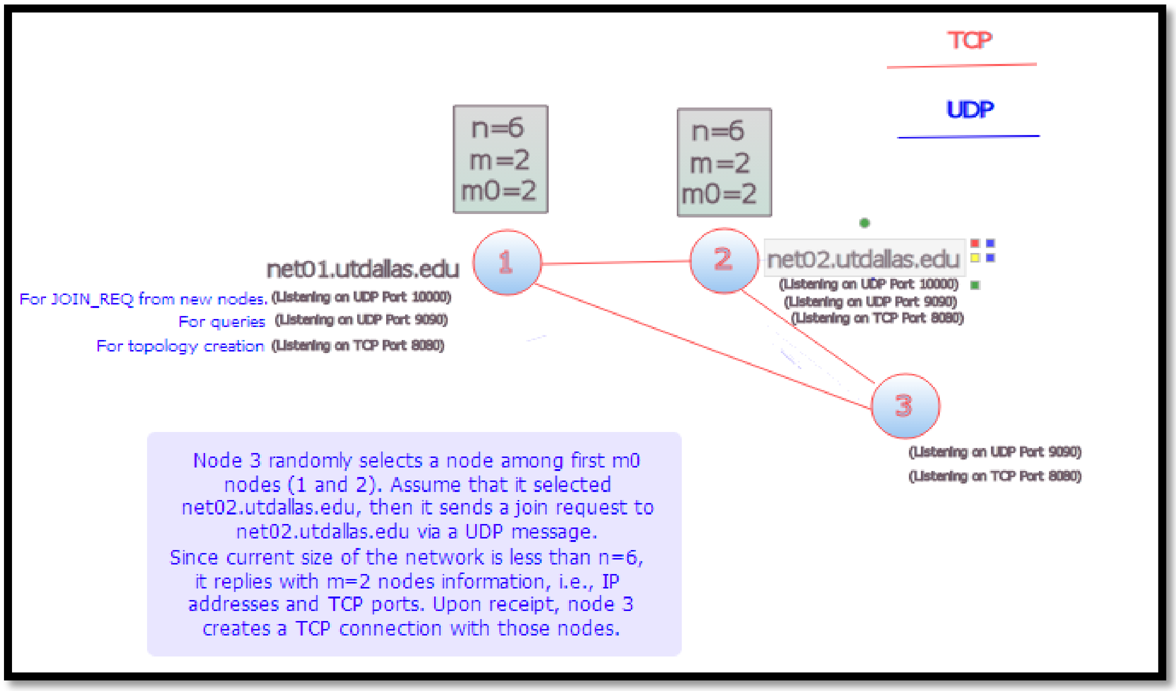
6 Example

The following is an example with $m = m_0 = 2\ and\ n = 6$. Note that in these examples, I assume that m_0 nodes gives (IP, TCP-port) information of nodes to which new node will connect (See the 4th bullet in Section 4.1 for the alternative).


TCP
UDP
n=6
m=2
m0=2
n=6
m=2
m0=2
1
2
3
net01.utdallas.edu
For JOIN_REQ from new nodes. (Listening on UDP Port 10000)
For queries (Listening on UDP Port 9090)
For topology creation (Listening on TCP Port 8080)
net02.utdallas.edu
(Listening on UDP Port 10000)
(Listening on UDP Port 9090)
(Listening on TCP Port 8080)
1)JOIN_REQ
2)net01.utdallas.edu:8080
net02.utdallas.edu:8080
net01.utdallas.edu:10000
net02.utdallas.edu:10000

Node 3 randomly selects a node among first m0 nodes (1 and 2). Assume that it selected net02.utdallas.edu, then it sends a join request to net02.utdallas.edu via a UDP message. Since current size of the network is less than n=6, it replies with m=2 nodes information, i.e., IP addresses and TCP ports. Upon receipt, node 3 creates a TCP connection with those nodes.

FIGURE 2- Node 3 joins the network by randomly choosing one of the m_0 nodes and sending a join request. node 2 replies with m=2 IP-address:port information. When node 3 receives this information, it creates a TCP connection with net01.utdallas.edu and net02.utdallas.edu.


TCP
UDP
n=6
m=2
m0=2
n=6
m=2
m0=2
net01.utdallas.edu
For JOIN_REQ from new nodes. (Listening on UDP Port 10000)
For queries (Listening on UDP Port 9090)
For topology creation (Listening on TCP Port 8080)
1
2
net02.utdallas.edu
(Listening on UDP Port 10000)
(Listening on UDP Port 9090)
(Listening on TCP Port 8080)
3
(Listening on UDP Port 9090)
(Listening on TCP Port 8080)

Node 3 randomly selects a node among first m0 nodes (1 and 2). Assume that it selected net02.utdallas.edu, then it sends a join request to net02.utdallas.edu via a UDP message. Since current size of the network is less than n=6, it replies with m=2 nodes information, i.e., IP addresses and TCP ports. Upon receipt, node 3 creates a TCP connection with those nodes.

FIGURE 3- Node 3 joined the network by creating links with net01.utdallas.edu and net02.utdallas.edu.

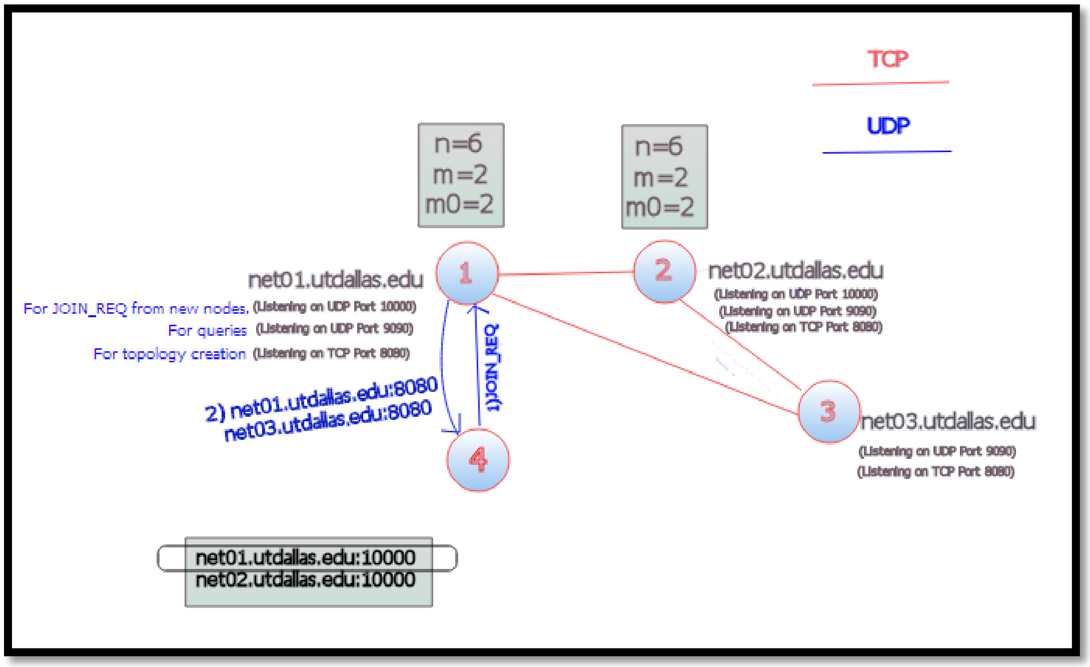

TCP
UDP
n=6
m=2
m0=2
n=6
m=2
m0=2
net01.utdallas.edu
For JOIN_REQ from new nodes. (Listening on UDP Port 10000)
For queries (Listening on UDP Port 9090)
For topology creation (Listening on TCP Port 8080)
1
2
net02.utdallas.edu
(Listening on UDP Port 10000)
(Listening on UDP Port 9090)
(Listening on TCP Port 8080)
1)JOIN_REQ
2) net01.utdallas.edu:8080
net03.utdallas.edu:8080
3
net03.utdallas.edu
(Listening on UDP Port 9090)
(Listening on TCP Port 8080)
4
net01.utdallas.edu:10000
net02.utdallas.edu:10000

FIGURE 4- Node 4 tries to join the network by sending a join request to one of the m_0 nodes, node 1 in this case. Since current network size is less than 6, node 1 replies with m=2 IP-address:port information to node 4 and node 4 creates TCP connections with those nodes.

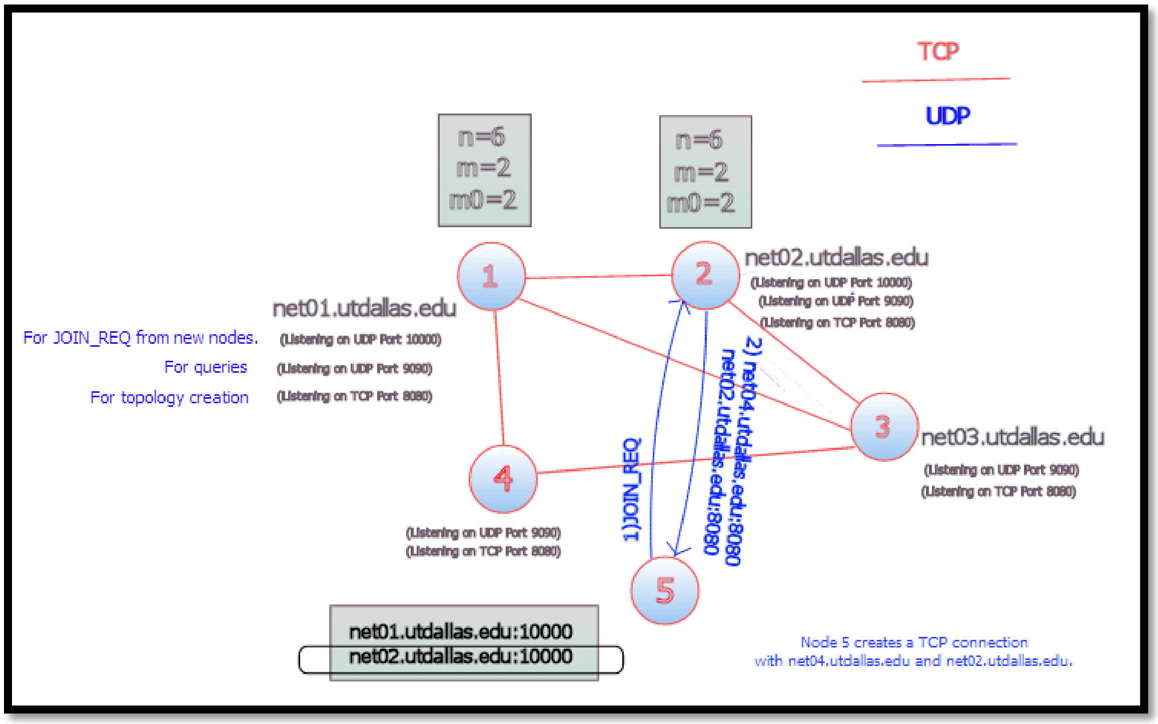

TCP
UDP
n=6
m=2
m0=2
n=6
m=2
m0=2
1
2
net02.utdallas.edu
(Listening on UDP Port 10000)
(Listening on UDP Port 9090)
(Listening on TCP Port 8080)
net01.utdallas.edu
For JOIN_REQ from new nodes.
(Listening on UDP Port 10000)
For queries
(Listening on UDP Port 9090)
For topology creation
(Listening on TCP Port 8080)
3
net03.utdallas.edu
(Listening on UDP Port 9090)
(Listening on TCP Port 8080)
4
(Listening on UDP Port 9090)
(Listening on TCP Port 8080)
1)JOIN_REQ
2) net04.utdallas.edu:8080
net02.utdallas.edu:8080
5
net01.utdallas.edu:10000
net02.utdallas.edu:10000
Node 5 creates a TCP connection
with net04.utdallas.edu and net02.utdallas.edu.

FIGURE 5- Node 5 tries to join the network in the same way.

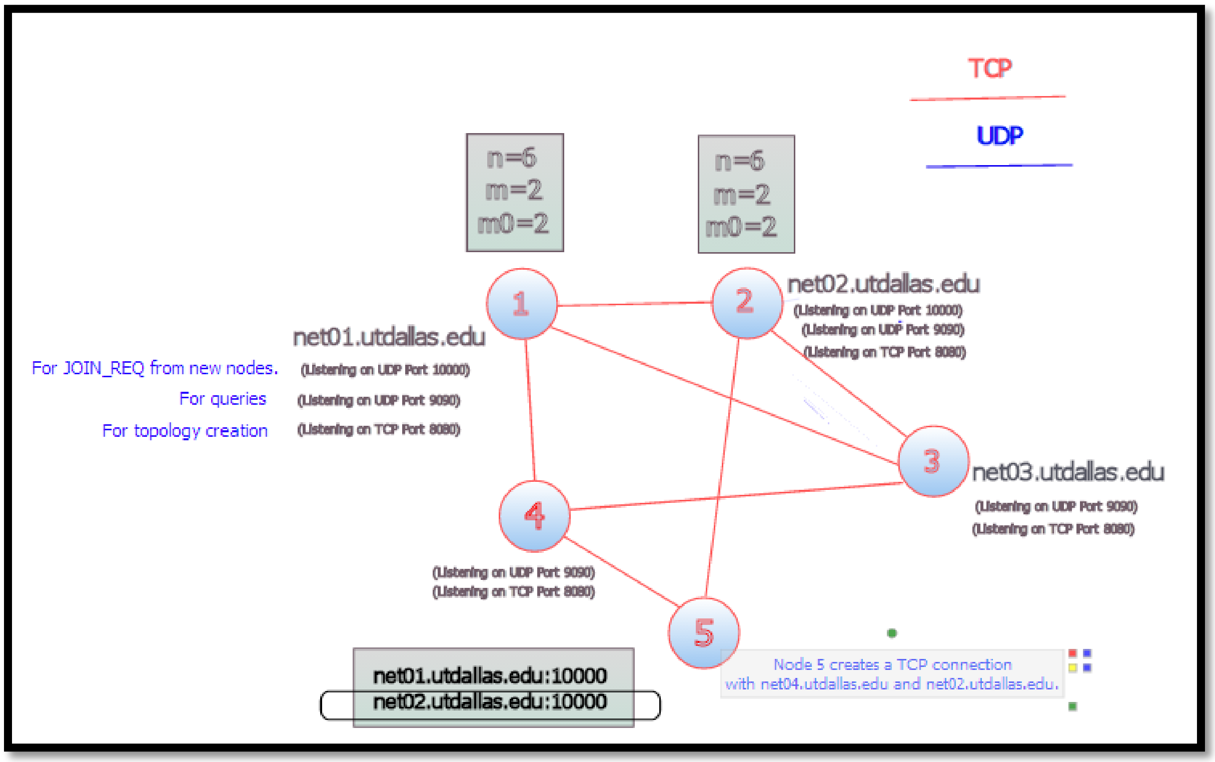

TCP
UDP
n=6
m=2
m0=2
n=6
m=2
m0=2
1
2
net02.utdallas.edu
(Listening on UDP Port 10000)
(Listening on UDP Port 9090)
(Listening on TCP Port 8080)
net01.utdallas.edu
For JOIN_REQ from new nodes.
(Listening on UDP Port 10000)
For queries
(Listening on UDP Port 9090)
For topology creation
(Listening on TCP Port 8080)
3
net03.utdallas.edu
(Listening on UDP Port 9090)
(Listening on TCP Port 8080)
4
(Listening on UDP Port 9090)
(Listening on TCP Port 8080)
5
Node 5 creates a TCP connection
with net04.utdallas.edu and net02.utdallas.edu.
net01.utdallas.edu:10000
net02.utdallas.edu:10000

FIGURE 6- Node 5 joined the network.

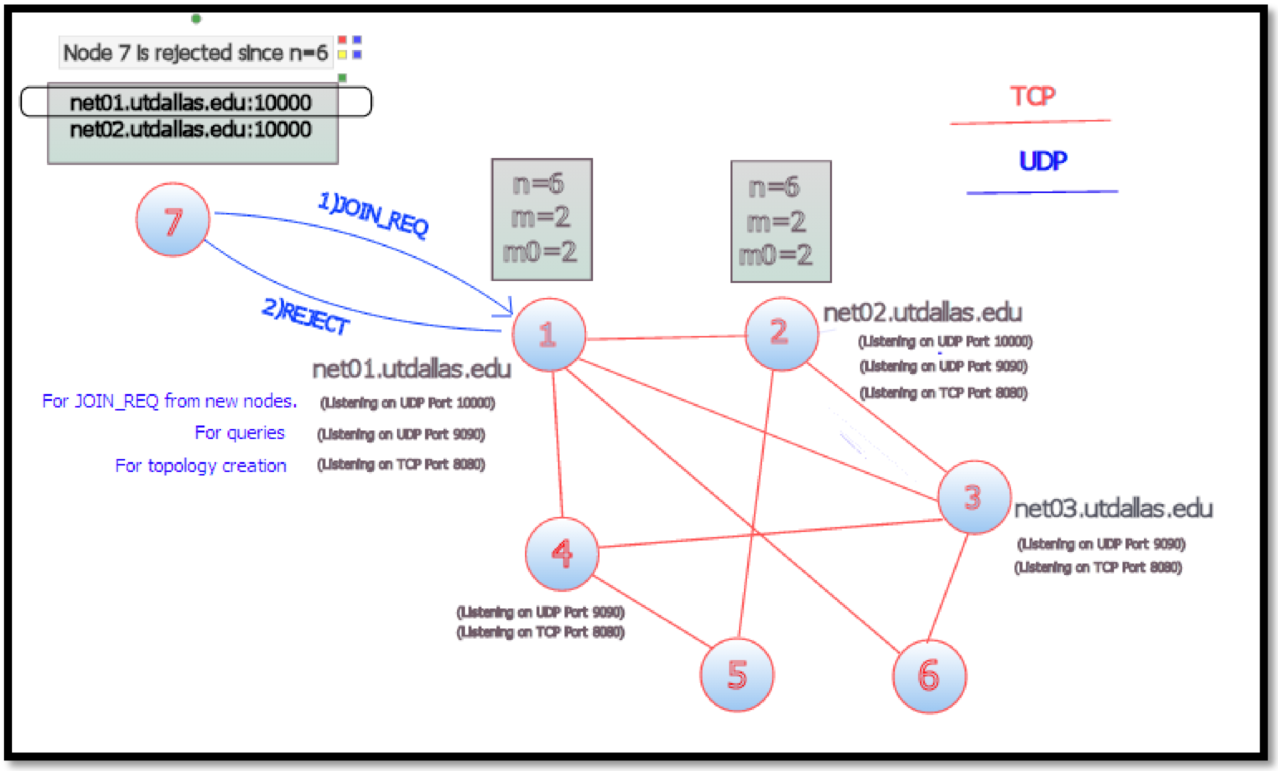


FIGURE 7- Whenever n+1=7th node tries to join the network, it receives a reject message.

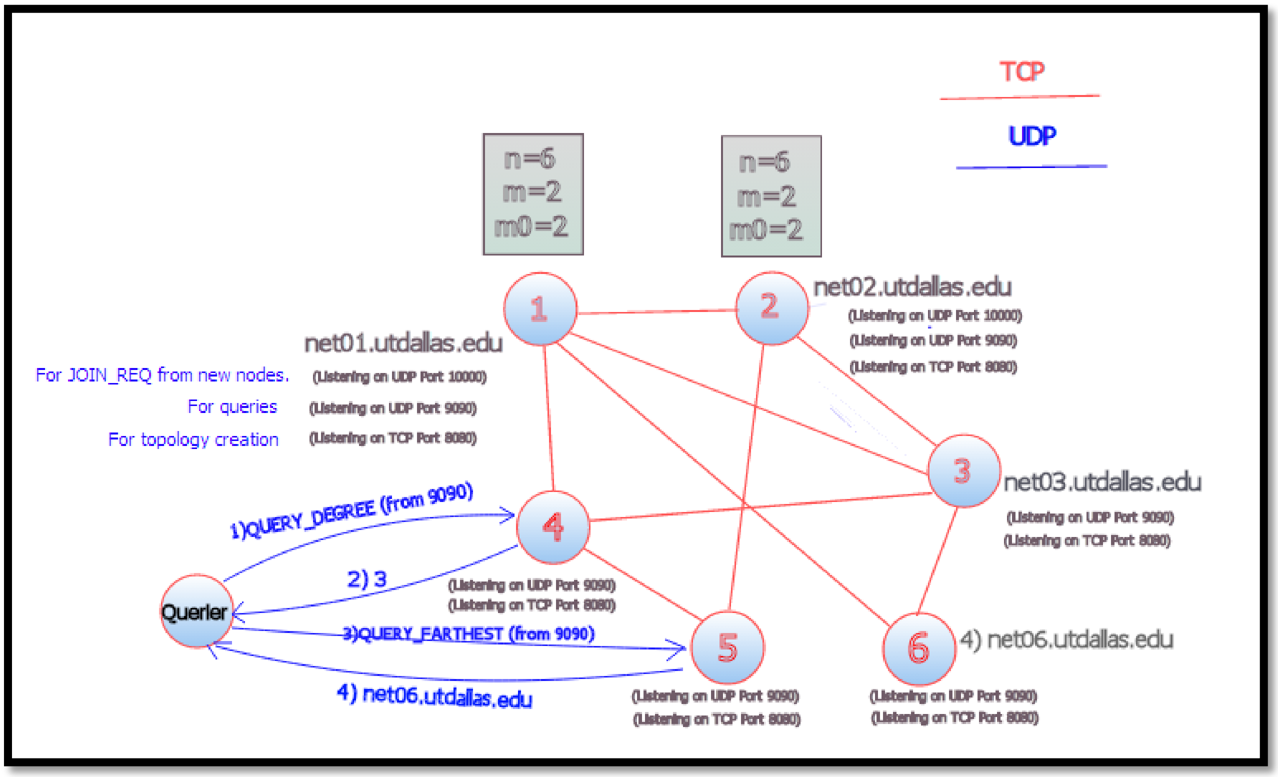


FIGURE 8- A node sends queries to nodes in the network from a dedicated udp port 9090,