****

**BLG 433E**

**COMPUTER COMMUNICATIONS**

CRN: 12337

**PROJECT #2**

Submission Date: 03.12.2014

**GROUP MEMBERS:**

TUĞRUL YATAĞAN 040100117

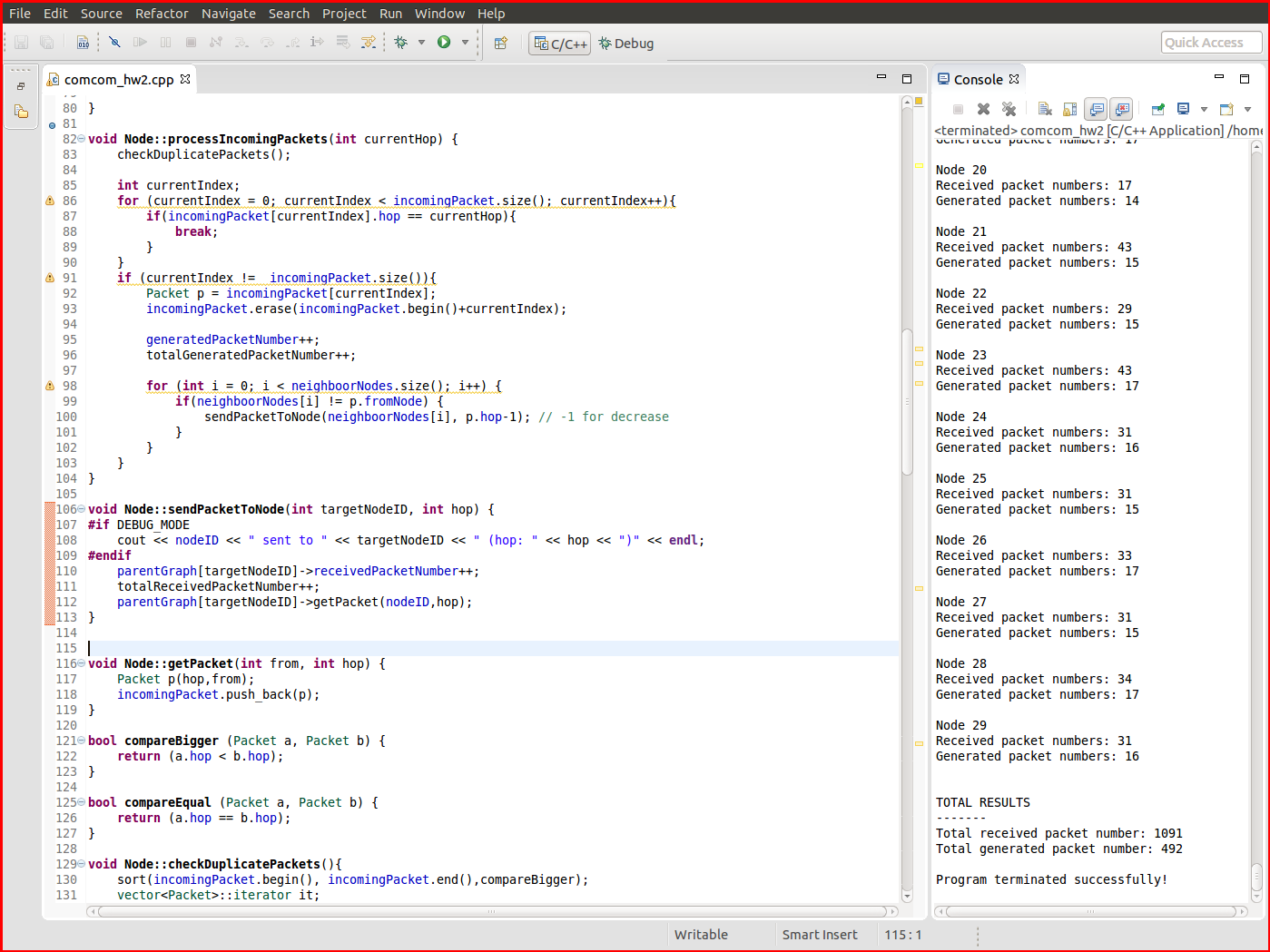
EMRE GÖKREM 040100124

# **Introduction**

In this project a random graph is created and then flooding algorithm runs on this graph.

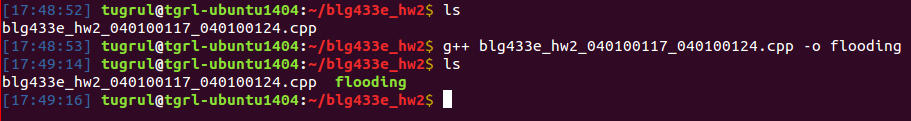
# **Development Environment, Compilation and Running**

All implementation is developed under Ubuntu Linux via Eclipse integrated development environment.



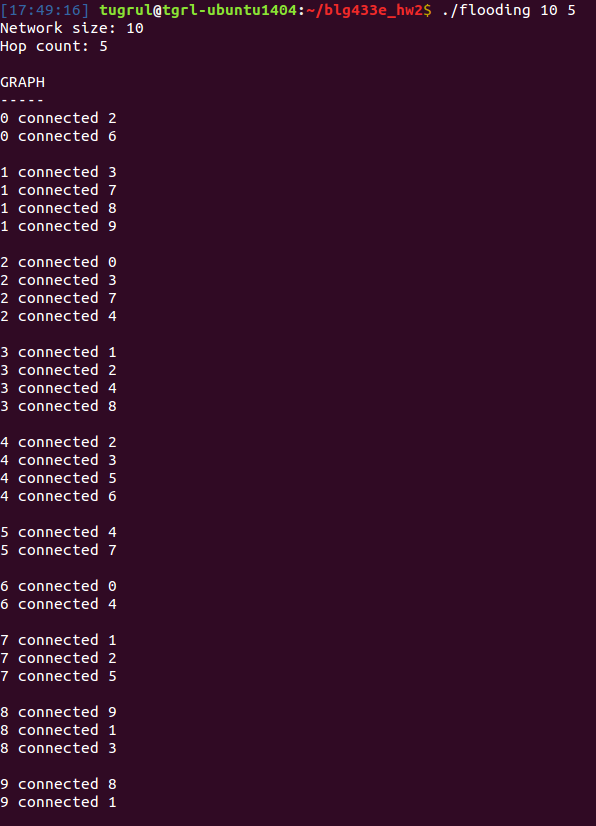
All the code is in one .cpp file. The application can be compiled with:

g++ blg433e\_hw2\_040100117\_040100124.cpp –o flooding

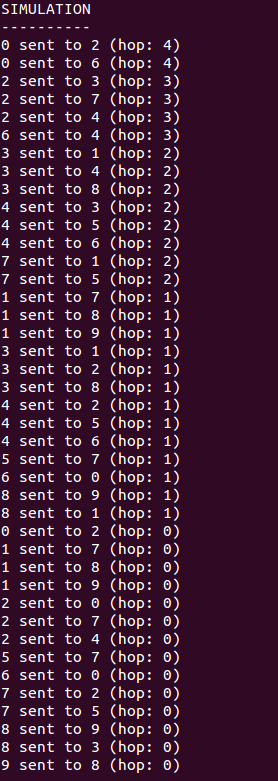


And binary application can be run with:

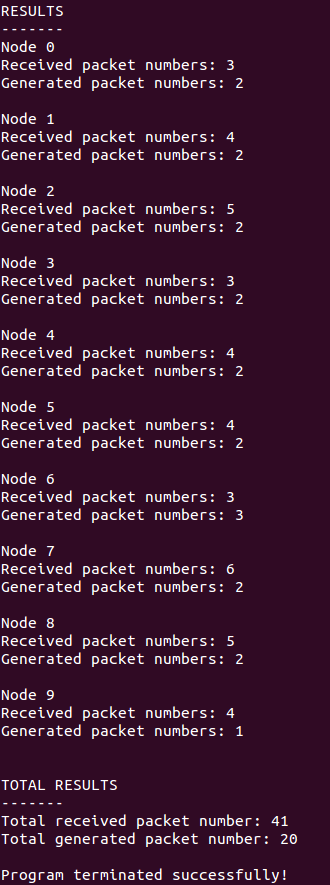
./flooding network\_size hop\_count



Firstly randomly generated graph’s connection map is shown at top and then simulation process steps are shown.



Lastly results per node and results per total are shown at the end of the program.



# **Answers for Report Questions**

# **Creating network graph**

* Graph class is responsible for creating network graph with connecting Node classes together. At initialization, a random degree between 2 and 4 assigns at every node. And then Graph::createGraph()function connects nodes randomly node by node. Every node keeps its connection in vector<int> neighboorNodes vector. If previously assigned node degrees are not sufficient, suitable nodes degree’s incremented by Graph::incrementSuitableDegree()function. If node’s degree is already 4 (maximum), node’s degree cannot incremented. After all nodes connected, Graph::checkConnected() check if all nodes are connected by BFS traversing network with Graph::traverseGraph(int id) recursive function. If graph is not connected Graph::reconnectGraph() function cleared old graph connection map and recreates graph until the graph is confirmed as connected.
* Every instance of Nodeclass has vector<Packet>incomingPacket vector to hold incoming packets. If a node sends packet to another node, an instance of Packet class is pushed target node’s incoming packet vector. Node::processIncomingPackets() function of every node process packet sending steps in Graph::startSimulation() function within a loop, node by node. Simulation runs as hop count number for every node, if node has packet to proceed, node sends its received packet to all its neighbor nodes.
* Every node sends packet to neighboring nodes incoming packet vector so at the same time target node gets it receiving packet. Whole network scanned in a loop for if node has a receiving packet. If node receives packet(s), node eliminates duplicate packets and send a newly generated packet with decreased hop count to all its neighboring nodes. All nodes are scanned for number of hop count in main simulation loop. Received and generated packet counters keeps track of every nodes statistic during simulation and after simulation total received and generated packet counter statistics are created.

# **Simulation with hop count as 5**

Received and generated packet numbers table for **fixed hop count as 5**:

|  |  |  |
| --- | --- | --- |
| Simulation results for fixed hop count as 5 | | |
| Network Size | Received Packet Number | Generated Packet Number |
| 10 | 47 | 19 |
| 20 | 78 | 37 |
| 30 | 84 | 39 |
| 40 | 96 | 43 |

# **Simulation with network size as 30**

Received and generated packet numbers table for **fixed network size as 30**:

|  |  |  |
| --- | --- | --- |
| Simulation results for fixed network size as 30 | | |
| Hop Count | Received Packet Number | Generated Packet Number |
| 5 | 113 | 49 |
| 10 | 373 | 177 |
| 15 | 533 | 292 |
| 20 | 972 | 479 |

# **Explanation**

The number of generated packed increases with both of hop count and network size increase. Generated number depends on connections of nodes so sending packet between nodes. If network size increases, number of nodes increases. Because of this, the number of packet sending increases therefore the number of generated packets increases. Likewise, if the hop count increases, the number of packet sending increases therefore the number of generated packets increases. But effect of the hop count is bigger than effect of the network size because the hop count provides sending more packets at same time. If the hop counter is bigger, more nodes generate packets.