Algorithm:
When a new node wants to join this network, it sets its current status as joining in its membership list. If it is not the introducer to the network, it makes a new connection to the introducer of that network and sends a JOIN message type UDP packet to it. If it is the introducer it just sets itself to active. Since it is the job of the introducer to introduce a new node to the network, when it receives a new JOIN request from a new node from outside the network it updates its membership to add the new node to it and set it to active. Next it makes a new connection to all the nodes in the network including the new one and sends a JOIN packet with just the new node's details. All the other nodes update this new node's data and the new node itself when it receives the JOIN packet, sets itself to active. Every node has a pinger function which runs every 1 second, updates the timestamp of itself in its membership list and then sends a copy of it's entire membership list to its neighbors, which in our case is 2 succeeding nodes and 2 preceding nodes. It only sends these UDP packets to its neighbors if they are also in active status. As soon as it sends the PING packet to its neighbors, the node makes an entry to the timers list with the hostname of the neighbors as well as the timestamp of when it sent it. Once the neighbors receive a PING packet, they compare the membership list with the ones they have, and if any of the nodes in the received membership list have a newer timestamp, it updates the local membership's instance of that particular node to the newer one. Once that is done, it sends back an ACK packet back to the sender of the PING. The Pinging node receives the ACK and deletes the entry to the timer list if it is within specified timeout time, which in our case is 3 seconds. In the ACK packet the sending node sends its own node details, which the receiver updates in its own membership list. For the leave method, the node makes a new LEAVE packet_type message and sends this message via UDP to its neighbors. Once it has sent the leave packet to all of its 4 neighbors, it sets itself as INACTIVE. The neighboring nodes on receiving the LEAVE packet, set the status of that particular node as INACTIVE and update the timestamp. The timercheck method basically keeps checking the timers dictionary to see if there are any entries in the dict and keeps comparing current time to the one in the dictionary and if the current time exceeds 3 secs sets the node with that old time as errored and deletes that entry from the timers list.
Completeness is not violated as a message from a single node can be propagated to the rest of nodes within 3 seconds. A failure of a node is detected within all nodes in at most 6 seconds as every 1 second we are sending a ping and our chosen failure time is 3 seconds.

(i)Bandwidth Usage for 6 Nodes:
Per Node: PING Packet Size = 954, ACK Packet Size = 233
Measured Total Bandwidth Usage for 6 Nodes = 21.366 Kilobytes / s

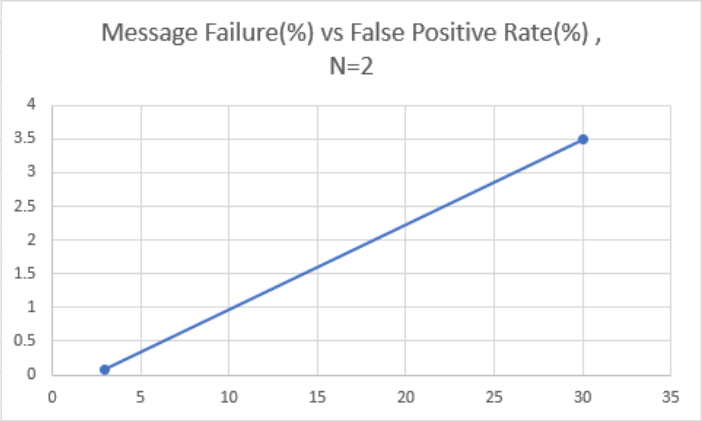
(ii)
Join: The average bandwidth increases when another node joins the 6 node network. The bandwidth usage for the system is now 26.128 Kilobytes / s
Leave: The average bandwidth of the system decreases as there are less nodes communicating via PINGs and ACKs.
Average bandwidth of system after 1 Node Leave = 14.837 KB/s
Fails: Average bandwidth of system decreases since there are less nodes communicating with each other. The average bandwidth is the same being 14.837 KB/s

(iii)

When N = 2:

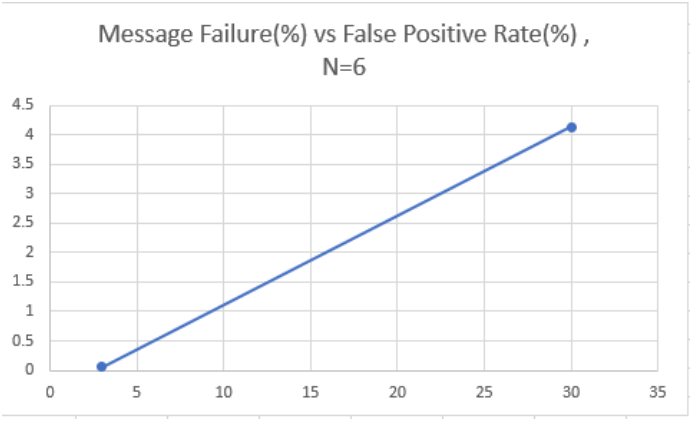
Confidence Interval (90%) = (1.645 * St Dev) / sqrt(Sample Size)

%	Average (%)	St Dev (%)	Confidence
Drop Rate = 3	0.084	.008	0.0058
Drop Rate = 30	3.49	.11	0.0809



When N = 6:

%	Average (%)	St Dev (%)	Confidence
Drop Rate = 3	0.0522	.003	0.0022
Drop Rate = 30	4.14	.10	0.0735



As the message drop rate increases, we expect the false positive rate to increase. When the message drop rate is 3% for both N = 2, N =6, the false positive rate is similar. When we increase the message drop rate to 30%, the false positive rate is higher for N = 6 than N=2 due to the volume of messages being sent in the n=6 system compared to n=2 system. Our findings are consistent with the expected results.