University of Tromsø

INF-3200 Distributed Systems Fundamentals

Assignment 2

November 9, 2015

1 Introduction

This report will describe a peer-to-peer network. The network will have the ability of nodes joining and leaving. One of the nodes in the network will be a leader.

- Least sum of squares (LS)
- Perceptron
- Gaussian Naive Bayes
- Two-layer perceptron (2LP)
- Support vector machine (SVM)
- Basic sequential algorithmic scheme(BSAS)

2 Technical Background

Leader election is a method for deciding which one of the nodes will be the leader. This will change the state of that node to become a leader node. All other nodes will then recognize that node as the leader. During each election, only one node will become the leader.

2.1 Election with unique ID

The process with the highest ID will eventually become the leader of the system. Each node will check its own id against the current favorite. If the node has a bigger ID it will set itself to the favorite and pass the election message to the next node.

3 Design

In this assignment we have used the chord implementation from the previous implementation as the basis. This means that the system will form a ring, where each node knows only about it successor. This assignment will implement leader election, joining and leaving on top of that system.

3.1 Join

During a join, the node will performe a number of operations to set up its state. To be able to join the network the new node needs to know about at least on other node in the network. It will then query this node for its own place in the system. The new node hooks onto the successor, and updates its new predecessor.

3.2 Leader election

The leader election will be initiated during a
getCurrentLeader message. The node receiving this message will also be in charge of initiating the eleciton. This node will pass a election message to all other nodes. The other nodes will then either respond with their own identity or with *None* if they are not a candiate for the leader. When a node has been elected to a leader it will broadcast itself to all other nodes.

3.3 Leaving

A node will leave if it recives the SIGTERM signal. This can be done by killing the process that runs the node. When a node leaves it needs to make sure the structure of the system is intact. It will do this by doing the reverse of the action taken by the join method. It will update its predecessor's succesor with its own succssor. The leading node can be the leader of the system. It will then have make sure that all other nodes are aware that a new election has be done.

4 Discussion

4.1 Leader election

When starting the program, each node will perform the join method. This method can demanding, therefore the leader election will not happen if a leader is not needed at the moment. The leader election will only happen when there is a need for a leader. Such as during the *getCurrentLeader* message. If a leader node leaves the system. A new election will only be held when a leader is required.

4.2 Complexity

The election algorithm in this report is simple. It will have a complexity of $O(n)$, since there will be exactly n messages passed around with a network of size n. The program will therefore scale up pretty well. At the minimum all nodes need to agree on the same leader.

4.3 Evaluation

Performance test for an election with different number of nodes:

nodes	time
2	0.30
4	0.30
8	0.40
16	0.43
32	0.63

From the table we can see that the time it takes to elect a leader increases when we run with more nodes.

5 Implementation

Implemented on the uvrocks cluster using python. A bash script is used to run the startup the nodes.

6 Conclusion

This system has been tested with up to 30 nodes where it works. It supports leaving and joing nodes on demand. The system also support a leader node, where election will happen if they are needed. If a leader leaves the system a new election will be held.