**UMBC, Fall 2013**

**Advanced Operating Systems**

**CMSC 621 Project 1**

**Gossip based Distributed application System Using Erlang**

**David Lachut Mahbubur Rahman**

[**dlachut1@umbc.edu**](mailto:dlachut1@umbc.edu)[**mahbub1@umbc.edu**](mailto:mahbub1@umbc.edu)

**Kaavya Srinivasan Shrikant Bhangdiya**

[**kaavya1@umbc.edu**](mailto:kaavya1@umbc.edu)[**shrik1@umbc.edu**](mailto:shrik1@umbc.edu)

**Introduction**

Gossip Protocol is a type of communication protocol used mostly in large scale distributed systems. It is based upon gossip conversations which is common in many social circles. Gossip Protocols are popular in distributed application due to its simplicity, scalability and high reliability even in constantly changing environments. Gossip Protocol involves periodic message exchanges between node pairs, which eventually results in information being spread throughout the system which is similar to human gossiping. The term epidemic protocol is sometimes used as a synonym for a gossip protocol, because gossip spreads information in a manner similar to the spread of a virus in a biological community. In the case of large scale distributed systems, instead of diseases, they spread information.

In this project, we are trying to split the file F into M fragments F\_1, F\_2, … , F\_M and then place them at various nodes of the system. While placing the fragments, we place each fragment at one or more nodes i.e. we replicate some of the fragments. All the fragments are not of the same size. We use gossip algorithm to perform the following computational tasks:

1. Finding the longest word in F and storing it at all the nodes.

2. For specific word(s), we find in which nodes this/those word(s) is/are stored.

We start this task at node 1, with the user specifying the word(s).

3. Finding the most frequent word in the text file F and storing it at node 1.

4. Updating the contents of fragment i at each node that may have a copy of it.

The update originates at node 1, with the user specifying the fragment number.

**Primary Working**

The first operation is file fragmentation. The file is fragmented in to m parts and distributed among the n nodes in the system. After the distribution of the fragments every node has at least 1 fragment. The distribution is such that two nodes can have the same fragments. This essentially means that the fragments are replicated and distributed.

When gossip starts, each node communicates with it neighbouring nodes, and it exchanges the node data, if required. ……………..

……………………

**Problem Statement**

Consider a single large text file F and a system that consists of N computer nodes. Split the file F into M fragments F\_1, F\_2, … , F\_M and place them at various nodes of the system. Place each fragment at one or more nodes (that means that you should replicate some of the fragments). The fragments may not all have the same size. Perform the following computational tasks:

* Find the longest word in F and store it at all the nodes
* For specific word(s) find in which nodes this/those word(s) is/are stored. Start this task at node 1, with the user specifying the word(s).
* Find the most frequent word in the text file F and store it at node 1.
* Update the contents of fragment i at each node that may have a copy of it. The update originates at node 1, with the user specifying the fragment number.

In doing these computations, our users are willing to trade-off accuracy for latency and communication costs. The highest accuracy for given latency and communication costs is highly desired.

Design and develop a distributed application that performs these computational tasks. Your solution must utilize gossip and be robust, efficient, and effective at solving the problem(s) at hand. Your application should be implemented using the Erlang distributed computing system.

1. **Find the longest word in F and store it at all the nodes**

**Initial Setup:**

A file is fragmented into m number of fragments and all the fragments are stored at various nodes, at least one fragment at each node, and some nodes have copy of same fragment, i.e. replication of fragments.

**Working**

1. **For specific word(s) find in which nodes this/those word(s) is/are stored.**

**Initial Setup:**

A file is fragmented into m number of fragments and all the fragments are stored at various nodes, at least one fragment at each node, and some nodes have copy of same fragment, i.e. replication of fragments.

**Working**

1. **Find the most frequent word in the text file F and store it at node 1.**

**Initial Setup:**

A file is fragmented into m number of fragments and all the fragments are stored at various nodes, at least one fragment at each node, and some nodes have copy of same fragment, i.e. replication of fragments.

**Working**

1. **Update the contents of fragment i at each node that may have a copy of it.**

**Initial Setup:**

A file is fragmented into m number of fragments and all the fragments are stored at various nodes, at least one fragment at each node, and some nodes have copy of same fragment, i.e. replication of fragments.

**Working**