**Programming Assignment 3**

# Performance Results

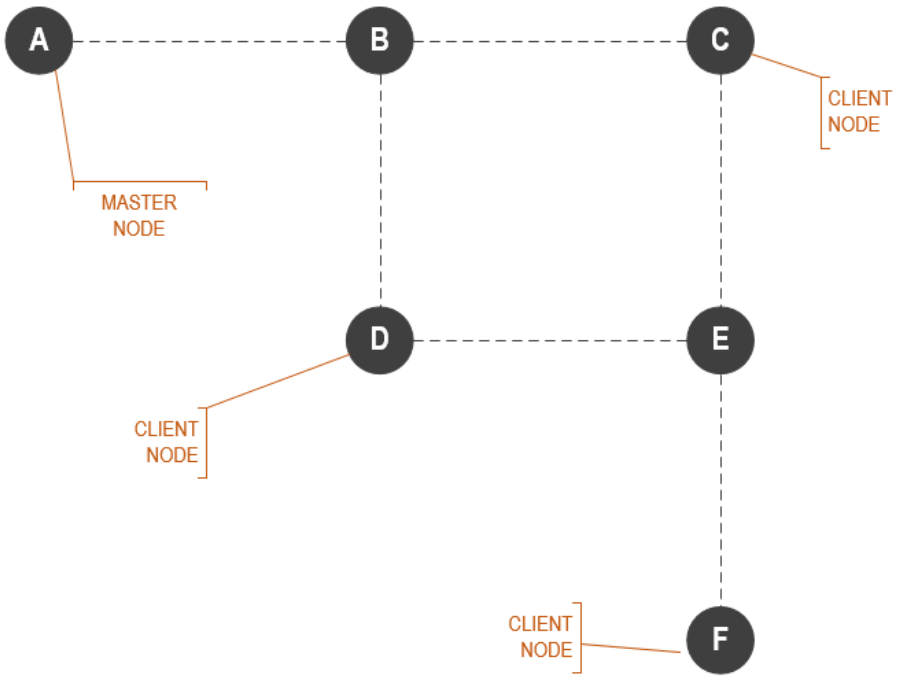
# Maintaining file consistency a Gnutella-styleP2P system


Arun Mathew Iype A20278285
Maheshwara Reddy A20284393


## Contents

## Test Setup

The diagram below shows the test setup for performance analysis. The Node "A" is the Master Node for the file "test_6kb.txt". The Nodes "C", "D" and "F" are programmed to repeatedly search for the file after a particular interval.




This setup is used for analyzing both PUSH and PULL structures.

The file " test_6kb.txt" at "A" is modified at a rate determined by the exponential probability distribution. The algorithm used for modifying the file at "A" is as follows:

A function to modify the file is started every "t" seconds. The probability that a modification will occur in the next t seconds is calculated as:

**p** = **P**(X <= t) | where P(X <= x) = 1 - exp(-lambda * x) *for x >= 0*

Here **P** is the implementation of the cumulative distribution function.

```
if  (r <= p)
  T = 0

  generate modification
else
  T += t
   sleep for t seconds
```

Where "r" be a random number between 0 and 1.

The values of the variables chosen are:

***t*** : 4 seconds

***lambda*** : 0.25

The probability distribution (cumulative) for different values of *lambda* is shown the figure below.

```
mahesh@mahesh-Inspiron-7520:~/Rep/CS550/Project_3/TestBed$ ./etest

Lambda  1.000000        0.750000        0.500000        0.250000
-----------------------------------------------------------------
        0.000000        0.000000        0.000000        0.000000
        0.632121        0.527633        0.393469        0.221199
        0.864665        0.776870        0.632121        0.393469
        0.950213        0.894601        0.776870        0.527633
        0.981684        0.950213        0.864665        0.632121
        0.993262        0.976482        0.917915        0.713495
        0.997521        0.988891        0.950213        0.776870
        0.999088        0.994752        0.969803        0.826226
        0.999665        0.997521        0.981684        0.864665
        0.999877        0.998829        0.988891        0.894601
        0.999955        0.999447        0.993262        0.917915
        0.999983        0.999739        0.995913        0.936072
        0.999994        0.999877        0.997521        0.950213
        0.999998        0.999942        0.998497        0.961226
        0.999999        0.999972        0.999088        0.969803
        1.000000        0.999987        0.999447        0.976482
        1.000000        0.999994        0.999665        0.981684
        1.000000        0.999997        0.999797        0.985736
        1.000000        0.999999        0.999877        0.988891
        1.000000        0.999999        0.999925        0.991348
mahesh@mahesh-Inspiron-7520:~/Rep/CS550/Project_3/TestBed$
```

## Performance

### Results

The percentage of invalid query results for PUSH : **1.5**%

The percentage of invalid query results for PULL : **21.75**%

### Analysis

From the results, it can be inferred that the PUSH is more efficient than the PULL based systems. However that could be due to the test setup and the environment where the testing was carried out.

PUSH: This method may not be very efficient for a large file sharing network with many files at each Node. For a system where the files are frequently changed, this method could clog the network with a large number of de-validate messages.

PULL: The refresh frequency used was 10 seconds. A higher frequency for validate requests could have helped to maintain a lesser percent of invalid query results.