

## Project 1: Network Size Estimation of P2P Systems

Due date: Midnight Feb. 27, 2009

### 1 Introduction

In telecommunication networks the exact number of active users (the users who pay bills) is quite accurately maintained for network administration purposes. In computer networks (the Internet, for example), due to the openness of the system, the number of hosts often remains unknown. In terms of network size, this problem gets more significant in any Peer-to-Peer (P2P) systems because of the well known phenomenon *churn*, i.e., a fast and unpredictable membership change. In order to manage network systems, however, the network size should be known or at least be estimated closely to the actual value.

The problem of network size estimation in fact has been well recognized due to its fundamental nature of any dynamic open system. In this project, an interesting methodology of *shuffle* and one of the well known network size estimation algorithms, *Jack-Knife Algorithm* will be used to estimate the size of a P2P system. See resources of the course web page for further information on shuffle algorithms and Jackknife algorithm.

### 2 Objectives

The goal of this project is to understand the dynamic nature of P2P systems and explore the effectiveness of the two algorithms (shuffle and Jack-Knife) in estimating the size of P2P systems.

### 3 Network Size Estimation

In general, network size estimation is done with some number of samplings. By sampling we mean that a tiny subset of the entire membership can be easily obtained by the node which needs to estimate the network size. The accuracy of estimation is dependent on the quality of network samplings, therefore. In this project we use shuffle algorithms as a means to sample the given network, and use Jackknife algorithm to estimate the network size.

1. *Syntax*: EST par1 par2 par3 par4
2. *par1*: network size (static)
3. *par2*: local view size
4. *par3*: Order of  $K$  in Jack-Knife algorithm
5. *par3*: number of samples in Jack-Knife algorithm

## 4 Analysis Report

1. Create a two-dimensional graph with x-axis for number of shuffle rounds and y-axis for estimates. Use the fixed network size of 32,000 nodes.
2. Create two more graphs in the same way as above with different network sizes, 16K and 64K nodes
3. Create a two-dimensional graph with x-axis for number of shuffle rounds and y-axis for estimates. Remove the half of the network nodes at a shuffle round, and show the estimates around that round. The result will show the resiliency of the estimation algorithm to an unpredictable network disaster, or how fast the algorithm converges to a new radically different value.
4. Create a graph which shows the estimation error in standard deviation for the three cases of 24K, 32K, and 64K nodes. Does the error grow as the network size grow? Better if not. Explain why it does or does not. You can provide an optimal value of samplings for each different network size.
5. Explain the difference between different values of  $K$ .
6. Provide a good explanation to each graph.

## 5 Shuffle

The main idea of shuffle algorithms is to *distribute* the entire membership across all the member nodes so that each member node has to maintain only a tiny subset of the entire membership (a local view), and *exchange* part of the local view with that of a randomly chosen another node. Every node runs the idea of shuffle at each *cycle*. The time period of the shuffle is arbitrary. However, every node initiates one shuffle at every cycle. Statistically, each node does shuffle twice: once initiated by itself, and once initiated by another random node. The algorithm of the shuffle is below.

1. Node  $P$  Increase the age of all nodes in the local view by one.
2. Node  $P$  selects a random subset of  $l$  neighbors from  $P$ 's own local view. The subset has a node  $Q$  with the highest age in the local view, and  $l-1$  other random nodes.
3. Node  $P$  replaces  $Q$ 's entry in the subset with a new entry of age 0 and with  $P$ 's address.
4. Node  $P$  sends this subset to peer  $Q$ .
5. Node  $P$  receives from node  $Q$  a subset of no more than  $l$  of  $Q$ 's neighbors.
6. Node  $P$  discards entries pointing to  $P$ , and entries that are already in the  $P$ 's local view.
7. Node  $P$  updates  $P$ 's local view to include all remaining entries, by firstly using empty local view slots (if any), and secondly replacing entries among the ones originally sent to  $Q$ .

Each node maintains a local view consisting of some number of entries. Each entry has information of a node address (ID) and an associated age. Each entry is in effect a pointer to another node. When a node  $P$  has a pointer to another node  $Q$ ,  $P$  can initiate communication with  $Q$ . So a node pointer represents connectivity. A map of connectivity of a given system will reveal how the system is connected. If the system is randomly connected, the map will show the same properties of a random network.

## 6 Jackknife Algorithm

Let  $f_i$  denote the number of nodes captured exactly  $i$  times in  $t$  samples with  $i = 1, \dots, t$ .  $f_0$  is the number of nodes never captured.  $S = \sum_{i=1}^t f_i$  is the number of distinct nodes seen in the samples. Then the population size is given by  $N = S + f_0$ . The estimates are given with different orders of  $K$  as follows.

$$N_{J1} = S + \left(\frac{t-1}{t}\right)f_1 \quad (1)$$

$$N_{J2} = S + \left(\frac{2t-3}{t}\right)f_1 - \left(\frac{(t-2)^2}{t(t-1)}\right)f_2 \quad (2)$$

$$N_{J3} = S + \left(\frac{3t-6}{t}\right)f_1 - \left(\frac{3t^2-15t+19}{t(t-1)}\right)f_2 + \left(\frac{(t-3)^3}{t(t-1)(t-2)}\right)f_3 \quad (3)$$

## 7 Sampling?

- *Local view size:* Use  $\log(N)$ , where  $N$  is the given network size.
- *Initial Condition:* Initially, you can populate the local view in an arbitrary manner. Then let the system shuffle about 1000 cycles to give it enough time to be randomized.
- *Sample:* One local view can be considered as one sample.
- *Number of samples:* Take as many samples as necessary.
- *Average value?* Repeat this more than 10 times and get a statistical average value.
- *Time gap between samples:* Give enough interval of about 100 shuffles between consecutive estimations.

## 8 Type

Individual or a team of two people. No penalty on teamwork.

## 9 What to Turn In

You turn in source code, binary code, README, and a makefile if there is. Any instruction or information necessary to remake the binary code. Show clearly at the top of the program: author names, email addresses, student IDs, and the machine name in the CS lab where you ran your code. The grader will remake the binary code and will run the binary on the machine where you produced the results. A README file is expected in each submission which details the procedure to demonstrate and test. Turn in one .tar file. Name it as “firstletter-of-the-first-name-and-lastname-p109.tar”. Mine would be “bchoi-p109.tar”, for example. If you are a team of two, name it lastname1-lastname2-p109s.tar. Use “blackboard” to turn in your work.

## 10 Grading

The maximum points of this project is 150. All work which produces reasonably good estimates will get full points.