DIMPLE-II: Dynamic Membership Protocol for Epidemic Protocols[1]

Jin Sun and Byung K. Choi
Departments of Computer Science
Michigan Technological University, Houghton, MI, USA 49931
{jinsun, bkchoi}@mtu.edu

Kwang-Mo Jung
Korea Electronics Technology Institute
jungkm@keti.re.kr

Received 9 November 2007; Revised 18 February 2008; Accepted 2 May 2008

Epidemic protocols have two fundamental assumptions. One is the availability of a mechanism that provides each node with a set of $log(N)$ (fanout) nodes to gossip with at each cycle. The other is that the network size N is known to all member nodes. While it may be trivial to support these assumptions in small systems, it is a challenge to realize them in large open dynamic systems, such as peer-to-peer (P2P) systems. Technically, since the most fundamental parameter of epidemic protocols is $log(N)$, without knowing the system size, the protocols will be limited. Further, since the network churn, frequently observed in P2P systems, causes rapid membership changes, providing a different set of $log(N)$ at each cycle is a difficult problem. In order to support the assumptions, the fanout nodes should be selected randomly and uniformly from the entire membership.

This paper investigates one possible solution which addresses both problems; providing at each cycle a different set of $log(N)$ nodes selected randomly and uniformly from the entire network under churn, and estimating the dynamic network size in the number of nodes. This solution improves the previously developed distributed algorithm called Shuffle to deal with churn, and utilizes the Shuffle infrastructure to estimate the dynamic network size. The effectiveness of the proposed solution is evaluated by simulation. According to the simulation results, the proposed algorithms successfully handle network churn in providing random $log(N)$ fanout nodes, and practically and accurately estimate the network size. Overall, this work provides insights in designing epidemic protocols for large scale open dynamic systems, where the protocols behave autonomically.

[1]This work has been supported in part by the State of Michigan via Research Excellence Fund (REF) at Michigan Tech., and Korea Electronics Technology Institute (KETI).

Categories and Subject Descriptors: Peer-to-Peer [**Network Protocol**]

General Terms: Membership Protocols

Additional Key Words and Phrases: Membership Management, Peer-to-Peer, Epidemic/Gossiping Protocols, Unstructured Overlays, Random Graphs

1. INTRODUCTION

Epidemic algorithms function by randomly disseminating information throughout the network in a manner similar to the spread of a disease [Demers et al. 1987], [Eugster et al. 2004], [Eugster et al. 2001], [Ganesh et al. 2003]. In each round, any given node randomly selects a set of peers from the entire network to communicate the given message to and continues to do so in subsequent rounds for the determined lifetime of the message. Each subsequent recipient then also gossips received messages in a similar manner for the remainder of the message lifetime. This paradigm of communication has been proved to reliably deliver the message to all nodes in the network [Birman et al. 1999], [Gupta et al. 2002] while reducing communication overhead and increasing scalability with respect to traditional broadcast schemes [Portmann and Seneviratne 2003], [Tanaraksiritavorn and Mishra 2004]. Due to these advantages, epidemic protocols have been utilized in many research areas, for instance, to manage databases [Demers et al. 1987], to maintain routing tables in large peer-to-peer (P2P) overlay networks [Voulgaris and van Steen 2003], to locate resources [Voulgaris and van Steen 2004], and to provide anonymous communication [Bansod et al. 2005].

Another advantage of epidemic protocols is that their parameters have been rigorously studied [Birman et al. 1999]. In order for the studied properties to be valid, however, each node running the epidemic protocol must randomly select $log(N)$ nodes to communicate a given message to, where N is the size of the entire network. In order to achieve this, nodes must therefore (1) have a method of choosing a random set of nodes, and (2) must know the size of the network. Both of these assumptions can be realized if information on the entire membership is contained either at each individual node or at some centralized node. Both approaches have a problem, however, as the network size increases. In the first approach, the maintenance of membership information consistency on all individual nodes becomes difficult. In the second approach, a centralized node can cause both a single point of failure and be a communication bottleneck. In addition, for dynamic systems like unstructured peer-to-peer (P2P) systems with churn (frequent node leaves and joins), a special care needs to be given to ensure that membership information is valid.

In order to use epidemic protocols in dynamic systems with frequent membership change, this paper investigates an engineering approach in an effort to find a good solution to the two assumptions of (1) randomly choosing $log(N)$ nodes and (2) knowing the network size under the constraint of network churn. By randomly choosing $log(N)$ nodes we mean that each node should be able to find different $log(N)$ nodes in different cycles of epidemic. As a result, $log(N)$ nodes should be chosen randomly and uniformly from the entire membership. By knowing the network size we mean that each node should be able to estimate the current network size with a

practically tolerable difference from the actual network size. As a result, estimated network sizes will allow epidemic protocols to use proper values of $log(N)$. Estimating network sizes should not be overwhelming such that each node should be able to follow network size changes closely in time domain. In addition, network size estimates on all nodes should show only a small variance such that most of the nodes recognize about the same value of N most of the time. By the constraint of network churn we mean not only node leaves and joins but also asymmetric network churn which shows either more leaves than joins (shrinking networks) or more joins than leaves (expanding networks).

The engineering approach proposed in this paper is based on a set of distributed algorithms, which (1) allocates a tiny subset of the entire membership at each node, where an entry is a node pointer such that different nodes have a different subset, (2) exchanges part of the tiny subset of a node with a set of randomly chosen node pointers and, (3) performs statistical analysis on the tiny subset to estimate the dynamic network size. Following a previous work [Voulgaris et al. 2005], in this paper, the tiny subset is called the *local view*, and exchanging part of a local view with part of another local view is called *shuffle*.

The essence of this approach is to utilize two concepts: *shuffle* and sampling. First, the concept of shuffle was previously studied in [Stavrou et al. 2004] and in CYCLON [Voulgaris et al. 2005]. Shuffle was used as a way to disseminate information quickly to flash crowds in [Stavrou et al. 2004] and was also used as an efficient way to manage dynamic membership. The work of CYCLON is an enhanced version of [Stavrou et al. 2004]. In addition, a methodology to extend the work of CYCLON to address the issue of network churn with shuffle, DIMPLE (DynamIc Membership ProtocoL for Epidemic protocols) has been proposed [Sun et al. 2007]. Second, the concept of sampling has long been studied to estimate the system size or population. The proposed method of network size estimation, in this paper, is inspired by [Mane et al. 2005], where two independent samples were used to estimate the system size. DIMPLE-II adapts the methodology of [Mane et al. 2005] to the structure of shuffle. This results in an efficient and unique estimation, utilizing the infrastructure of shuffle without introducing additional protocols and communication overhead. The new contribution of this paper is twofold; one is to further utilize the notion of shuffle to estimate the dynamic network size, and the other is to enhance the performance of DIMPLE-II by improving the shuffle under network churn.

The remainder of the paper is organized as follows. Section 2 details the related work within the field of membership and network size estimation. Section 3 next describes the work of CYCLON to provide the background for the current work. Section 4 provides the system model of DIMPLE-II while the main ideas of this paper are presented in Section 5. After that, Section 6 evaluates the main ideas by simulation and comparison between DIMPLE-II and CYCLON. Lastly, Section 7 concludes the paper with a summary of the results and also describes avenues of possible future research.

2. RELATED WORK

The two related areas of this work are *dynamic membership management* and *dynamic*

network size estimation. By 'dynamic' we mean network churn which can be of high rates and also asymmetric with frequent node joins and leaves.

Within the area of membership, a network can have centralized information, fully distribute the information amongst its nodes, or use some sort of middleground. A group of work ([Aguilera et al. 1999], [Carzaniga 1998], [Das et al. 2002]) takes the centralized approach. In this research, the membership information is all handled at a centralized server or set of servers. Upon entering, each node must communicate with the server and upon leaving must do likewise (or the server must perform routine checks for connectivity). As a result, the centralized server always has a consistent view of the membership of the system and therefore a correct perception of the network size also. Utilizing this information, the server can then assign a random set of nodes for each of the members to communicate with. The drawback to these schemes is that they leave themselves vulnerable to a single or fixed number of points of failure (i.e. the server(s)). In addition, performance bottlenecks are created at the servers and communication overhead is increased due to the necessity of informing the server. As a result, others have chosen to distribute the membership information at the individual nodes, giving each of them either full or partial views.

The first approach would be to fully replicate the membership information at each node, making the choosing of a random set of nodes to communicate with trivial. However, in large-scale networks, this approach is not possible due to the memory resources it would occupy (e.g. [Cuenca-Acuna et al. 2003] has this constraint). Partial views, however, can reside at individual nodes since they take up significantly less space. The goals then become ensuring that by using these partial views, an overall randomness is still guaranteed and an accurate network size is also known by each individual node.

One avenue that does attempt to alleviate the problem of providing randomness for overlay networks is the concept of Newscast Networks [Jelasity et al. 2003]. In these networks, the focus is on "macro behavior"-a term used to define overall functions of a network regardless of whether individual functions perform correctly (from biological inspirations such as ant colonies). In simulations, it is shown that the path length and clustering of the randomly-generated networks provide a reasonable amount of randomness. The protocol makes use of exchanging parts of local views of the current membership in order to attempt to achieve a random subset of the global membership at each node. Another closely related method randomizes the partial views by probabilistically keeping or forwarding subscription messages from joining or leaving nodes [Ganesh et al. 2003]. This latter research, however, is not concerned with maintaining the network size at each node.

Another approach to providing random partial views is through the use of a shuffling mechanism. Such mechanisms are necessitated by the fact that left on their own, P2P networks do not form random graphs, but rather power law graphs [Lv et al. 2002]. Within the area of shuffling, the Peer Sampling Service considered three dimensions making the tuple (peer selection, view selection, view propagation) [Jelasity et al. 2004]. The options for these are shown in the following brackets: {random, head, tail} for peer selection, {random, head, tail} for view selection, and {push, pull, push-pull} for view propagation. Within these options, head and tail correspond to

the most recently added (to the view) and least recently added nodes, respectively. Push, pull, and push-pull depict whether the initiating node sends its view, receives the view of the other node, or both. Within this taxonomy, the previously described Newscast Networks and the shuffling mechanism of CYCLON would both correspond to a (random or tail, head, push-pull). For the 27 possible combinations of the classifications, the Peer Sampling Service measured partitioning, path length, clustering coefficient, and the degree distribution of nodes throughout the network.

While the work mentioned above focuses on the detailed algorithms of shuffling, the work of Allavena et al. [Allavena et al. 2005] provides an interesting theory predicting the network partition probability of the general idea of shuffling. As one can imagine, with a reasonably large size of local views and an appropriate shuffle length, the probability of creating a network partition is extremely small. We consider that this work [Allavena et al. 2005] generalizes the resilience of the shuffling mechanism demonstrated by simulation in CYCLON [Voulgaris et al. 2005].

The problem of estimating dynamic network size is not unique to this research. Because the system size is one of crucial information for many applications, a number of methodologies have been proposed. One typical trade-off to this problem is communication overhead and accuracy. One straightforward approach to measure the network size is to count the nodes using some communication infrastructure. As the network size increases, however, this would result in impractically large communication overhead and time. Moreover, when the network changes in size at a high rate, i.e., is highly dynamic, counting itself may not be feasible. This has led researchers to sampling-based approaches, where typically the entire system size is guessed based on a sampled data set. The problem then becomes not only to provide practical accuracy but also timely estimation.

Recently, two generic methods of network size estimation have been proposed in [Massoulie et al. 2006], namely, Random Tour and Sample and Collide. The former is based on the return time of continuous time random walk to the node originating query. The latter is based on counting the number of random samples gathered until a target number of redundant samples are obtained. As these two methods are network architecture independent, they can be used for dynamic systems such as large scale P2P systems. Another interesting work of P2P system size estimation can be found in [Mane et al. 2005], where two independent random samples of population are used for statistical analysis of entire population. Although, this method was originally used in oceanography and epidemiology, the generality of this method has some potential to be used for dynamic P2P systems. The method of size estimation in DIMPLE-II, in this paper, is inspired by this approach. Thus DIMPLE-II adapts this approach to the infrastructure of shuffle. Other network architecture-specific methods such as [Bolot et al. 1994], [Horowitz and Malkhi 2003], [Kostoulas et al. 2005], [Manku 2003], [Psaltoulis et al. 2004] can also be found in the literature. Interested readers are referred to these works.

Although the problem of dynamic membership management and the problem of dynamic network size estimation have been studied in separate two topics in the literature, DIMPLE-II suggests an integrated methodology, which is able to manage dynamic membership and estimate network size at the same time by utilizing the

infrastructure of shuffle.

3. PRELIMINARY

This section starts with an informal description of epidemic protocols, for which this paper provides distributed algorithms for dynamic membership management and network size estimation. The two technical backgrounds for the distributed algorithms are then followed: CYCLON shuffle for membership management and the capture-recapture method for network size estimation.

3.1 Epidemic Protocols

An epidemic protocol delivers a message from a sender to all members in $log(N)$ rounds of forwarding, where N is the number of peers in a P2P system. In the first round, a sender transmits a copy of the message to $log(N)$ randomly chosen peers. After that, each peer that received a copy of a message repeats the same job, i.e. it forwards a copy of the message to another $log(N)$ randomly chosen peers. The message lifetime is represented as a number of protocol rounds and is decremented by one at each intermediate forwarding. A copy of a given message is held at a node until its lifetime is exhausted. Hence, a copy of the message is gossiped (randomly forwarded) as many times as its current lifetime and then is discarded. The time interval between two consecutive protocol rounds can be set arbitrarily. As a result, all peers are highly likely to receive a copy of the original message in $log(N)$ rounds of cascaded forwarding.

3.2 CYCLON Shuffle

The shuffling mechanism of CYCLON is shown in Algorithm 1 and the join mechanism is in Algorithm 2. Each node initiates CYCLON Shuffle exactly once in each cycle. Nodes are not synchronized to perform shuffling in the same order at each cycle. Different orders can be seen in different cycles. Thus, a node, which shuffles at ith in a cycle, can shuffle at jth in the next cycle ($i \neq j$). In effect, each node shuffles views probabilistically twice in each single cycle; once initiated by itself and another initiated by another random node.

Data structure. Each node maintains a local view which consists of c number of entries (The method of determining the value of c is outside the scope of this work). Each entry contains the information of a node address (ID) and an associated age. Hence, 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. As a result, a node pointer represents connectivity and when all pointers are viewed, the connectivity of the entire system is revealed. If the system is randomly connected, the map will show the same properties of a random network. In this paper, a node pointer that points to a node that has already left the system and thus is not available anymore (dead) is called a *dangling* pointer.

Description of Algorithm 1. The essence of the CYCLON shuffling mechanism is the exchange of a portion of each individual local view between nodes. Upon receiving

a shuffle request from a node P, a node Q selects a subset of the same size l out of its own local view, and then sends this subset to the shuffle requester P. By repeating the shuffle operation at every cycle, the system automatically converges to a steady state where both the average shortest path length[1] between any pair of nodes and the clustering coefficient remain comparable to those of a random network. A random network is typically defined as one that has the same number of nodes and the same number of edges, and is connected randomly. While this convergence is desirable and is a fundamental characteristic of the shuffling operation, the convergence speed and the difference in the two metrics from those of a random network were concerns of CYCLON.

The age starts from 0 (zero) when a node pointer is first created in a local view, and increases by one at each subsequent cycle thereafter. Furthermore, the age is transferable such that whenever a node pointer is shuffled to another local view, it preserves the age. As a result, a node pointer is chosen as a shuffle partner by the time the age reaches c. This means that a node pointerΓfis life time has a maximum value of c. As seen in Algorithm 1, once a pointer is chosen as a shuffle partner, the age of the pointer is reset to 0 (zero).

Algorithm 1: CYCLON Shuffle

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

Algorithm 2: CYCLON Join

1. P contacts a randomly chosen node Q, an introducer.
2. Upon receiving P's join request, Q launches c number of random walks where each one has a unique entry of Q's local view as its starting address. Each one has a Time To Live (TTL) whose minimum value is the average shortest path length.
3. An intermediate node decrements the TTL by one and forwards the join request to a node I randomly chosen out of its own local view.
4. At node R, where the TTL is decremented to 0 (zero), a randomly chosen entry is replaced with P's address and age 0. R sends the replaced entry to P.
5. P collects all the replaced entries and starts using these entries as its own local view.

[1]The averaged shortest path length in number of hops between all the possible pairs of nodes in the system.

3.3 Capture-Recapture Method

The capture-recapture method was originally proposed in [Darroch 1958], and subsequently used to estimate population of France in 1786 and some other similar purposes [Mane et al. 2005]. There are two critical assumptions to this method; *closed* system and two *independent* random samplings. By closed system we mean that the system population does not change during the estimation process. By two independent random samplings we mean that the two population samplings are obtained randomly and independently of each other. The essence of the algorithm is presented in Algorithm 3. While the approach seems statistically reasonable, the

Algorithm 3: Capture-Recapture

1. Take a random sampling of a system (capture).
2. Let the number of entities captured in Step 1 N_1.
3. Mark the captured entities in Step 1.
4. Release the captured entities to the system.
5. Give enough time to the system such that the captured entities in Step 1 are randomly mixed in the system.
6. Take another random sampling of the system (recapture).
7. Let the number of entities captured in Step 6 N_2.
8. Let the number of entities captured both in Step 1 and 6 n_{11}.
9. Then the total population of the system N is estimated as $\frac{N_1 \times N_2}{n_{11}}$

assumption of random samplings presents a technical challenge to open systems such as P2P systems because the population of a P2P system changes unpredictably at anytime. Also, a random sampling of a P2P system itself is not an easy task either. The work of [Mane et al. 2005], therefore, uses a random walk-based capture-recapture approach because a previous work reports that independent network samplings can be achieved by taking some random walks [Gkantsidis et al. 2004].

In this paper, however, because we have an infrastructure of shuffle, by utilizing the shuffle infrastructure, we propose a rather simple method to estimate the network size by any node at anytime efficiently and effectively. The details of the proposed method are presented in Section 5.

4. SYSTEM MODEL

Nodes can join and leave the system at anytime, which creates high network churn. When a node joins the system, it contacts an existing node chosen randomly from the system, which is called an *introducer*. A mechanism is assumed to be in place which automatically chooses one member uniformly at random for a new node to contact and join the system.

The parameters of epidemic protocols are dependent on the system size. The system size changes reasonably fast due to asymmetric network churn. Thus, the system size either increases or decreases. DIMPLE-II does not distinguish a node fault from a

node leave. Any node which does not respond to a protocol activity is considered *dead*. Intermittent faults are not considered in this work.

The time interval of the *cycle* of the shuffling mechanism is theoretically independent of that of the protocol round for epidemic protocols. In static systems, where the number of nodes is relatively constant, once the network is randomized (i.e. the entire membership information is randomly and uniformly distributed over the local views), the shuffle cycle does not affect the quality of epidemic protocols. This is because the local view remains random and uniform regardless of the cycle. In dynamic systems, however, because the membership frequently changes, larger cycles will not be able to provide the same quality of random and uniform distribution of the entire membership to the local views. In this work, it is assumed that the cycle of the shuffling and the round of epidemic protocols are the same. It would be interesting to investigate the impact to the quality of epidemic protocols caused by the difference between the shuffle cycle and the epidemic round when the former is larger than the latter.

5. MAIN IDEAS

DIMPLE-II has four components; shuffle, join, leave, and network size estimation. Shuffling is initiated by each node at every epidemic protocol round. In effect, each node performs the shuffling mechanism twice; once initiated by itself and another initiated by another random node. At the end of a shuffle, each node can estimate the current network size. Network samplings required to estimate the network size are provided by buffering the information of the dynamic local view. Nodes are not synchronized to perform a shuffle. The global order of shuffles can be different in different rounds.

5.1 Data Structure

In CYCLON, each entry of a local view consists of two pieces of information-the address and the age of the node. In this work we extend the information of the entry to have information on *visited* nodes, where a "visited" node refers to a list of nodes in whose local views a node P has resided. The size of the visited list is determined dynamically by the average shortest path length of the system. The average shortest path length is that of the corresponding random network. A visited list is created and maintained as follows. Assume a node Q has a pointer to another node P in its local view. At a shuffle round, the pointer to P is shuffled to another node R. The node R receives a new entry P from Q. The new entry comes with a pointer to P and another pointer to Q where the pointer to P comes from. From the node R's perspective, Q is the "just visited" node by P. At a later shuffle round, the node R shuffles the entry of P to another node S. The node S receives a new entry from the node R. From the node S's perspective, P has visited Q and R in sequence. Therefore, the visited list of P increases by one whenever it is shuffled to another node up to an upper bound, which is the average shortest path length of the network. When the visited list overflows, the oldest node is discarded. The oldest node is the node which has been in the list for the longest time. As a result, an entry of the local cache has the node address (ID), age, and a *visited* list. In addition, each node maintains a buffer which

contains s network samplings, where s means the number of local views captured in the buffer. For example, in case of s 30, each node allocates the buffer space proportional to $30 \times log(N) \times$ the size of a single entry of the local view. Thus, in terms of memory overhead, DIMPLE-II requires k times larger local view space and additional buffer space than CYCLON, where k is determined by the average shortest path length. Because the size of local views is $O(log(N))$, however, this linear increase of space overhead should not cause a big concern. For example, in case of local view size $2 \times log(N)$, the average shortest path length k is less than 10, up to the network size (N) one million. This implies that the local view size is practically large enough to reach any other peer node within 10 hops in a network of one million nodes. The averaged shortest path length increases with the network size very slowly. The space overhead of DIMPLE-II is highly scalable with the network size, therefore.

5.2 DIMPLE-II Shuffle

DIMPLE-II Shuffle is shown in Algorithms 4, 5 and 6. There are two differences between this shuffling mechanism and that of CYCLON. One is the shuffle length and the other is the way to choose the shuffle partner. As shown in the algorithm, a node P selects the oldest node Q from its local view, and then tries shuffle with the node Q. This single-entry shuffle is repeated until half of the local view are shuffled. In comparison to CYCLON, DIMPLE-II uses a single-entry shuffle multiple times while CYCLON performs a single shuffle with multiple entries. As a result, DIMPLE-II challenges the oldest nodes, half of the local view in size, from its local view at each cycle. Note that since DIMPLE-II shuffle always chooses the current oldest node, the system challenges older nodes much more frequently than CYCLON does. The motivation of DIMPLE-II Shuffle is to detect dangling pointers significantly faster than CYCLON does. Ideally each entry of the out-degree[2] should point to a node that is functional and still connected. While this is true for a static system, this claim is not valid when network churn is introduced. In our experiments, CYCLON produces a relatively poor quality of out-degrees with churn in that a large portion of the out-degree is actually pointing to dead nodes (which have left the system already but have not been detected yet as such). In an extreme case with high churn, there was a node whose out-degree did not point to any live (still existing) node in CYCLON. While a small number of dead node pointers in the local view may not degrade the quality of epidemic protocols significantly, we found in our experiments that the number of dead node pointers becomes larger with increasing churn rates. By executing DIMPLE-II Shuffle either a node Q will have an opportunity to fill an empty slot in the view (which may have been caused by either detecting a dead node or a large variance of the in-degree distribution) or a node P will challenge a random node to see if the node is alive.

In fact, DIMPLE-II Shuffle not only reinforces the quality of the out-degree but so for the in-degree also. Because the out-degree is a collection of pointers, reinforcing the pointers will result in a smaller variance of the in-degree by filling empty slots in the view and by removing dangling pointers. A large variance of the in-degree will

[2]The out-degree of node A is the number of pointers held by node A, pointing to other nodes.

degrade the quality of epidemic protocols because en epidemic protocol requires a local view which is uniformly random. As seen in the evaluation section, DIMPLE-II Shuffle indeed contributes to reinforcing the quality of both in and out-degrees.

One concern with DIMPLE-II Shuffle is the communication overhead and the amount of time to perform multiple single-entry shuffles within a single protocol round. Currently, DIMPLE-II Shuffle takes longer than that of the CYCLON shuffling mechanism because DIMPLE-II Shuffle repeats a single-entry shuffle multiple times in simulation. Although we conjecture that performing a single-entry shuffle many times *in sequence* and performing the same job *in parallel* would produce about the same results, finding the subtle difference is left as part of the future work. In practice, a node would perform multiple single-entry shuffles in parallel not in sequence because it would not need to be in sequence. The current method of sequential shuffle in DIMPLE-II is simulation specific. Thus, the communication overhead of DIMPLE-II would rather be in the number of messages to maintain the shuffle infrastructure in each cycle, not in the communication time. The problem then becomes the number of messages DIMPLE-II can process within an interval in parallel. In turn, this leads to the question; what would be the time constraint on the length of the epidemic protocol round? If the time interval for one epidemic cycle needs to be very short, for example, a few tens of mili-seconds, DIMPLE-II would be more difficult to meet the time requirement. For cases where the protocol round is in the range of seconds, however, this will not be a concern. In this paper, the time constraint issue is not further investigated as this would be an implementation problem. Instead, this paper is focused on the scientific understanding of the proposed distributed algorithms. Investigation on the practical time interval of the protocol round for implementation is left as future work.

Algorithm 4: DIMPLE-II Shuffle

1. Increase the age of all nodes in the local view by one.
2. Repeat DIMPLE-II Shuffle Initiate (Algorithm 5) as many as half of the local view times.

Algorithm 5: DIMPLE-II Shuffle Initiate

1. Select the oldest node Q in P's local view in terms of the age.
2. Set Q's age to zero and send P's address to Q.
3. When P receives a response from Q, P adds the node information to an empty slot if there is, or replaces the entry of Q with the received information of another node pointer.
4. If P receives no response from Q, P removes the entry of Q.

Algorithm 6: DIMPLE-II Shuffle Response

1. Q adds P's information to Q's local view with age 0 if there is an empty slot. Q then returns a node which is randomly selected from its local view (before P is added) to P.
2. If Q's local view is full, Q randomly selects a node and replaces it with P's information. Q then returns the replaced node information to P.

5.3 DIMPLE-II Join

The DIMPLE-II Join mechanism is shown in Algorithm 7. Compared to the corresponding mechanism in CYCLON, DIMPLE-II Join is significantly simpler. A new node P requires only one message exchange with another node Q and DIMPLE-II Shuffle to create its own new local view. This simple procedure eliminates or substantially shortens the time delay required in CYCLON Join, where a number of parallel random walks are launched to create a new local view for P to use. This dramatic reduction of the time delay is achieved by extending the information of each entry of a local view. Two questions may arise with this procedure. One is whether a global randomness is preserved by creating a new local view out of an introducer's local view. The other is whether the newly created local view contains sufficiently recent (fresh) and few or no dangling pointers. As can be seen in the evaluation section, however, the global randomness of the system before and after this operation is still the same. Intuitively, this procedure has almost the same effectiveness of the random walks of CYCLON Join. The difference is *when* the random walk takes place. In DIMPLE-II Join it happens k cycles before the join event by utilizing the visited list while it takes the action k cycles after the join event in CYCLON Join, by newly launching random walks where k is the average path length. The time delay to handle a new join event in DIMPLE-II is therefore one message exchange between the joining node and the introducer. The upper bound of the time delay incured by a join event in CYCLON can be very large because a random walk should take at least the average shortest path length number of hops, and any one hop walk could take a long response time to timeout when the next hop has already left the system (a dangling pointer). There is a chance that the newly created local view may have some dangling pointers, but this probability is quite small.

Algorithm 7: DIMPLE-II Join

1. P contacts a randomly chosen node Q, an introducer.
2. Upon receiving the P's join request, Q builds a local view by collecting the oldest node of the visited list of each entry of Q's local view.
3. Q sends this new local view to P.
4. P initiates DIMPLE-II Shuffle Initiate (Algorithm 5).
5. P starts using this local view as its own local view.

5.4 DIMPLE-II Leave

DIMPLE-II uses a *timeout* event to detect a dead node. Essentially it is the same method as used in CYCLON. The difference, however, is in how frequently the system tries to detect a dead node. In CYCLON each node initiates the shuffling mechanism exactly once in each cycle. Therefore, regardless of the in-degree of a dead node, the system requires at worst c number of cycles to purge the dangling pointer, where c is the size of the local view. In DIMPLE-II, the worst case detection time is bounded by half of the local view because the new single-entry shuffle is repeated for half of the

local view.

5.5 Network Size Estimation

The network size estimation of DIMPLE-II (Algorithm 8) is directly from the capturerecapture method described in the previous section. The difference is in the way in which network samplings are done. Following the original concept of *two samplings*, DIMPLE-II uses two buffers; one for capture and the other for recapture. More specifically, however, the samplings are done over many cycles not at one point in time. Moreover, as seen in the algorithm below, DIMPLE-II performs capture and recapture at the same time. Although this may appear to violate the two assumptions of the capture-recapture method (two independent random samplings), DIMPLE-II Shuffle in fact supports the two assumptions. First, the information in the local view is randomly collected from the entire membership. Second, given that each entry in the local view is collected independently and randomly, there is no need to give a time interval between the two samplings. Essentially, utilizing the shuffle infrastructure eliminates the cumbersome and costly procedure of network sampling. As seen in the next section, this estimation method follows the actual network size very closely.

Algorithm 8: Network Size Estimation

1. Initially, each node creates two sampling buffers S_c and S_{rc}, where S_c is the buffer for the first sampling (capture), and S_{rc} is the buffer for the second sampling (recapture).
2. At every cycle, each node records (adds) half of its local view, to S_c and the other half to S_{rc}. The first and the second halves are chosen randomly.
3. The two buffers are used in a FIFO fashion. The oldest sampling is dropped when the buffer overflows. Step 2 is repeated continuously.
4. The current network size N is estimated as follows: $N = (N_1 \times N_2)/N_{11}$, where N_1 is the size of S_c and N_2 is the size of S_{rc}, N_{11} is the number of duplicates that appear both in N_1 and N_2.

6. PERFORMANCE EVALUATION

In this section, we evaluate the performance of DIMPLE-II by simulation. As can be seen in the following figures, DIMPLE-II converges to a steady state significantly faster than CYCLON does due to the use of the single-entry shuffles and the *visited* list under reasonably high network churn. In addition, the resultant network (when in a steady state after the churn subsides) shows an even lower average shortest path length and a slightly higher clustering coefficient [Sun et al. 2007]. Overall, DIMPLE-II embraces network churn smoothly, enhances the global randomness achieved by CYCLON, and estimates dynamic network sizes accurately for epidemic protocol purposes.

Metrics. Two groups of metrics are used in this paper: one is to show the performance of DIMPLE-II in comparison to CYCLON, and the other is to show the performance of network size estimation. The former metrics will show how fast DIMPLE-II handles join and leave events, and consequent impact on the quality of in- and out-degrees

and on the global randomness already achieved by CYCLON. As can be seen in Algorithm 2 and 7, CYCLON has to launch random walk processes as many as the number of entries of the local view to create a new local view for the joining node. In contrast, DIMPLE-II gets a new local view at the cost of one message exchange with the introducer. Each random walk in CYCLON has to visit at least a lower bound number of hops, which is the average shortest path length of the system. At each hop, however, in CYCLON, a random walk can try to reach a node which is no longer in the system by using a dangling pointer. To resume the troubled random walk, the intermediate node has to timeout such a waiting. The timeout value in the application process can be very long compared to the shuffle cycle. The chances of using a dangling pointer in CYCLON is fairly high as can be seen in the following Figures on the node degree qualities. CYCLON has to wait to collect a reply from each random walk. The join procedure completes upon receiving all the replies. Then, the new node is recognized by the system as a new node by participating the regular shuffle operation. The entire time to collect all the replies to create a new local view, therefore, is necessarily much longer than DIMPLE-II. The quality of the node degree is in fact the crucial factor determining the quality of epidemic protocols. The latter metrics will show the proximity of the estimates to the actual network size which changes fast. Two systems are used for the first metrics: one system uses DIMPLE-II to provide dynamic membership service for a P2P system while the other uses CYCLON. Some rates of network churn are used to measure the relative time required for the two systems to converge to a steady state.

Simulation Setting. Throughout the simulation, we use our own cycle-based simulator written in C++. Variable network sizes up to 100,000 nodes are used to measure the metrics. The network size estimation algorithms are evaluated separately from the first metrics. For the first metrics, in one simulation run, the network size does not change in order to decouple the network size impact from the final results. This means that whenever a node leaves the system another node with another globally unique ID is added to the system to maintain the same network size. When a new node is added it contacts an introducer, i.e. a randomly chosen node from the entire system. The network churn is implemented as a Weibull distribution $\{\lambda = 21.3, k = 0.34\}$ of the node lifetime with an average value in cycles [Stutzbach and Rejaie 2004]. Whenever a new node is added, the new node is given a lifetime (in number of cycles). The lifetime of each node is decremented by one at each cycle. For example, when the shuffle cycle is 1 second and the average lifetime is 180 cycles, the time of the average lifetime would be about three minutes. The selection of this lifetime is based on previous work [Rhea et al. 2004] (see Table I). From this it can be seen that the average lifetime of three minutes is comparable to that of the two real systems-FastTrack and KaZaA.

When the lifetime reaches 0 (zero) the node silently leaves the system without generating any notification. During its lifetime a node always conforms to DIMPLEII and never fails. A dead node is detected when another node tries to contact it to shuffle with. The timer resolution to detect a dead node may be large in this simulation because the timeout value is given in the number of cycles. Although this resolution

Table I. From Handling Churn in a DHT.

First Author	Systems Observed	Session Time
Saroiu	Gnutella, Napster	$50\% \leq 60min$
Chu	Gnutella, Napster	$31\% \leq 10min$
Sen	FastTrack	$50\% \leq 1min$
Bhagwan	Overnet	$50\% \leq 60min$
Gummadi	KaZaA	$50\% \leq 2.4min$

would not be realistic for systems where a fine resolution (e.g. a few milliseconds) is required, it is still fair to compare the two systems in a highly dynamic situation. Initially, the two systems are completely randomized, after which the network churn begins. The simulation was run on a set of different desk top computers using Linux OS. The final results are represented independently of the wall clock time.

6.1 Convergence: Churn Processing Time

In order to demonstrate the improvement on the dynamic membership made by DIMPLE-II, the churn processing time of the two systems is compared here. The churn processing time has two components: one is the time required for the system to recognize a newly added node, and the other is the time to purge a dangling pointer out of the local view. In this section, two metrics are introduced for the comparison study: *join time* and *leave time*. The former is defined as the time elapsed from when a new node contacts an introducer to the time when the new node creates a complete local view of its own. The latter is defined as the time elapsed between when a node leaves the system and when the system completely purges the dangling pointer from all local views in the system.

A comparison of the two join times is trivial. The join time for DIMPLE-II is one cycle. This is obvious from the DIMPLE-II Join (Algorithm 7) procedure in the previous section. The CYCLON system join time is bounded by the average path length for message forwarding. Although the exact time for the CYCLON join depends on each nodeΓfis workload at runtime, the join time of the CYCLON system can be equated to the number of forwards required in the join procedure. The join time of CYCLON becomes even longer when a timeout due to a dangling pointer at an intermediate node is taken into account.

The leave time measurements are shown in Figure 1. In this experiment, an increasing system size from 1,000 to 100,000 nodes was utilized. The average node lifetime was 180 cycles, the local view size was $2 \times log(N)$, and the shuffle length was $log(N)$ for both systems. While it is expected that the DIMPLE-II system removes the dangling pointers sooner than the CYCLON system, it is interesting to note that the leave time of CYCLON increases at a much higher rate than that of DIMPLE-II, i.e. the speed gain becomes larger as the network size increases. The reason it looks like a step function is that the local view size increases step-wise by $2 \times log(N)$, where N is the network size. This increase results in more cycles to purge dangling pointers.

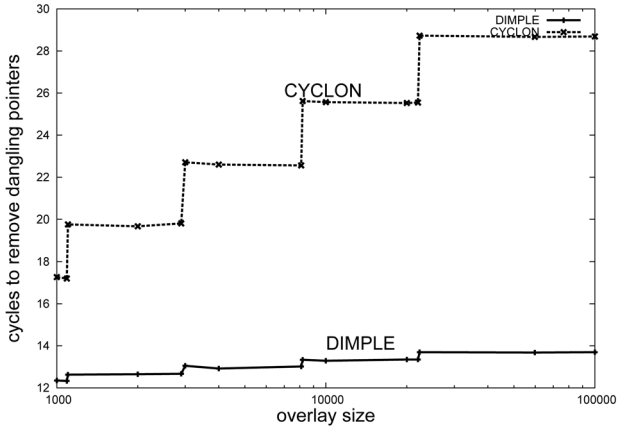


Figure 1. Cycles to remove dangling pointers.

6.2 Convergence Under Churn

The convergence of the CYCLON system is well studied in [Voulgaris et al. 2005]. According to the work, the CYCLON system converges to a steady state where the system shows the same properties as the corresponding random network even after a high percentage of node failures. In this section, both systems are investigated to see if they converge under network churn. In this experiment, the average node lifetime was 180 cycles, and the network size was 2,000. Both systems were run for 20,000 cycles to give enough time to observe convergent behavior. The metric used to measure this behavior was the *standard deviation* of the average shortest path length.

Figure 2 and Figure 3 show the standard deviation of the in- and out-degrees of both systems respectively. As can be seen, both systems converge to a steady state with the churn rate that was used throughout the experiments. Secondly, the DIMPLE-II system shows lower values. While it is very likely that lower values will support

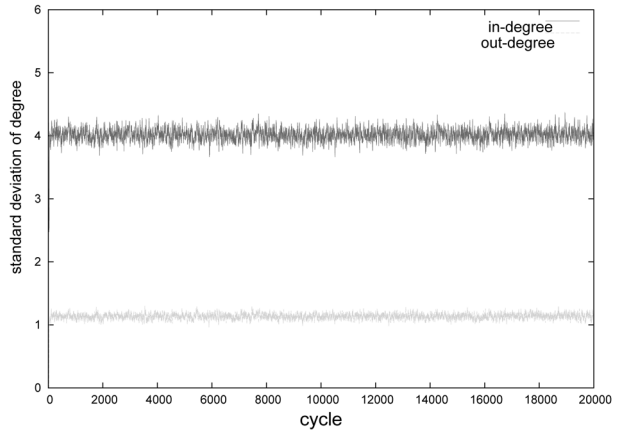


Figure 2. Node degree standard deviation of CYCLON with 2000 nodes.

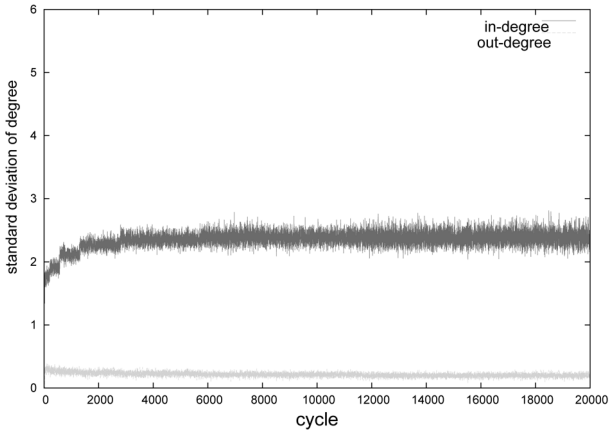


Figure 3. Node degree standard deviation of DIMPLE-II with 2000 nodes.

higher quality of epidemic protocols, further research is required to accurately measure the resultant quality of information dissemination and is beyond the scope of this paper. The rest of the figures in this paper show data obtained from a converged state.

6.3 Impact on Randomness: Degree Distribution

The quality of node degree distributions is critical to epidemic protocols because they determine the overall quality of information dissemination. First, in dynamic systems, a local view can have dangling pointers at any moment which degrade the quality of epidemic protocols eventually. The second issue is load balancing among nodes. Ideally, each node should have the same in-degrees over time because the critical parameter of epidemic protocols $log(N)$ assumes a random selection. Different in-degrees would bias the selection process of $log(N)$ in an undesirable way.

The quality of the node degree distribution, therefore, is measured by the two metrics in this section: *uniformness* and *freshness*. Uniformness refers to the distribution of the in-degree sizes considering only the *live* nodes. Freshness meanwhile refers to the distribution of out-degree sizes, again considering only the *live* nodes. Effectively, the two metrics are concerned about the same quality–i.e. node degree distribution. However, given that the size of the in-degree is not fixed at each node, measuring the number of pointers from live nodes is an effective measurement of the corresponding quality. On the other hand, because the size of the out-degree is uniformly fixed to some value at each node, measuring the percentage of dangling pointers is the preferred metric for observing the quality of the out-degrees.

Figure 4 shows the in-degree distribution of the two systems from an experiment where both systems had 100,000 nodes with an average lifetime of 180 cycles. The local view size was fixed to $2 \times log(N)$, and the shuffle length was $log(N)$ for both systems. As can be seen, the CYCLON system has a wider distribution than DIMPLE-II. Accordingly, the peak value of CYCLON is lower. Interestingly, the average values of the two systems are different. This is because CYCLON has a larger number of

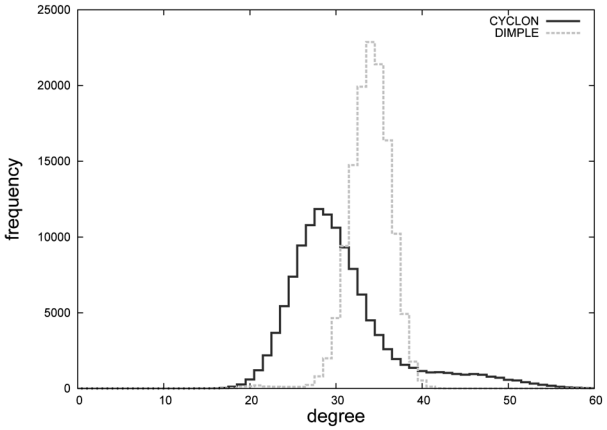


Figure 4. In-degree distribution in converged 100,000-node overlay.

pointers from dead nodes than DIMPLE-II. Consequently, the average value of CYCLON is shifted down from the value of $2 \times log(N)$, which is 34 in this particular case. On the other hand, DIMPLE-II maintains an average value very close to $log(N)$ because it can detect a dangling pointer much faster. Overall, this figure clearly demonstrates that DIMPLE-II maintains a higher quality of in-degrees under high churn.

Figure 5 shows the out-degree distribution of the two systems respectively from the same experiment. Although the size of the out-degree is uniformly fixed to $2 \times log(N)$ at each node, CYCLON shows a severe distortion of the nominal size when pointers are measured only from *live* nodes. This is, again, because the local views of nodes in the CYCLON system have a higher percentage of dangling pointers at runtime under churn. DIMPLE-II also contains dangling pointers, but due to the new shuffle procedure, the dangling pointers are eliminated from the local views in a more timely manner. This enables DIMPLE-II to maintain a majority of the local views at

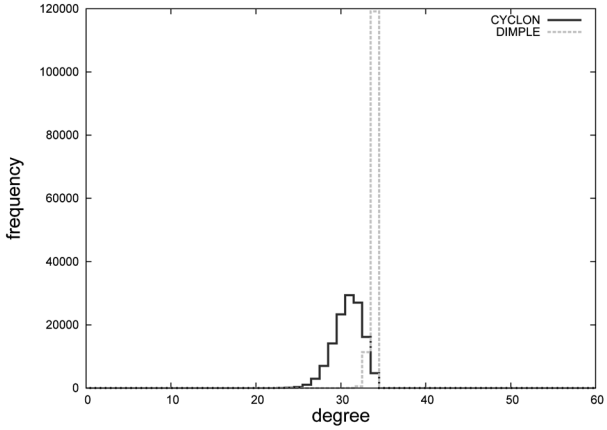


Figure 5. Out-degree distribution in converged 100,000-node overlay.

the original fixed value of $log(N)$. Taking these figures into account, DIMPLE-II demonstrates its capability to support a higher quality of node degree distributions even under high churn.

Average Shortest Path Length and Clustering Coefficient. The average shortest path length is defined as the averaged shortest path length between all possible pairs of nodes of the network. The clustering coefficient is defined as the ratio of the existing links among a node's neighbors over the total number of possible links among them. This metric is typically used to understand what percentage of the neighbors of a node are also neighbors among themselves. The average clustering coefficient is then the averaged value taking all the nodes in the system into account. In this experiment, lower values are desirable because higher values would result in both higher chances of network partitioning and poor information dissemination. Although not shown in this paper, CYCLON shows lower values than those of the corresponding random network. Further, the DIMPLE-II system slightly improves these two metrics from the CYCLON system. The details of these results can be found in the previous work (the conference version of this paper) [Sun et al. 2007].

6.4 Performance Gain with Increasing Churn

In this subsection, the standard deviation of the out-degrees is shown as a metric to show the performance difference with different churn rates. Figure 6 shows the metric for the out-degrees given a system size of 5,000 nodes. Intuitively, the performance difference between the two systems should increase with increasing churn rates and vice versa. As can be seen for average node lifetimes greater than 300 cycles, the out-degree of CYCLON is significantly affected by the network churn while that of DIMPLE-II is not. A similar trend can be found in the in-degrees too (not shown in this paper). Since an average lifetime of about 200 cycles is realistic, the performance gain that DIMPLE-II provides is justified in such situations with churn.

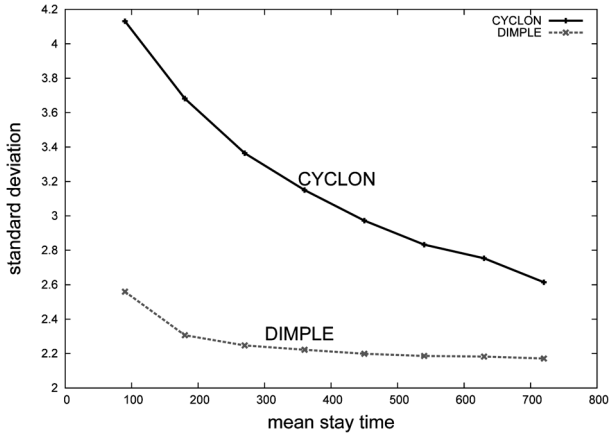


Figure 6. Out-degree standard deviation in converged 5000 node overlay with different mean node lifetimes.

6.5 Network Size Estimation

To measure the effectiveness of the proposed network size estimation, a separate set of experiments were performed. The effectiveness are measured in terms of the following three metrics: the average *difference* between estimates and actual sizes, the *standard deviation* of the estimation, and the convergence *speed* of the estimation to an abrupt change of network size.

The simulation results in this section are from DIMPLE-II only (Algorithm 8). The original capture recapture algorithm (Algorithm 3) is not directly applicable to the class of open systems incuding P2P systems, which have the same common problem, open membership or churn. In order to use the original method, the system should be static in membership. Comparsion of DIMPLE-II and the original method against dynamic open systems would technically be unfair. Comparison study of DIMPLE-II and other methods for dynamic open systems such as [Massoulie et al. 2006] are left as future work. The simulation settings for these experiments are the same as before (local view size, shuffle length, churn model, node lifetime) except the network size estimation. Several network sampling sizes were tried for the capture and recapture algorithm. For the given network sizes from 1,000 to about 60,000 nodes, however, the sampling size 30 produced good results. Smaller samplings resulted in less accuracy. More sampling sizes did not improve the results much. As a result, 30 samplings were used for the rest of experiments. As can be seen in Step 2 in Algorithm 8, one samping consists of $log(N)$ individual entries of the local view. The number of nodes in one sampling is therefore $log(N)$ × the number of nodes in a single entry of the local view. The number of nodes in a single entry in turn is upper bounded by the size of the visited list plus one. Since the visited list is upper bounded by the average shortest path length of the system, the number of nodes in 30 samplings is bounded by $30 \times log(N) \times$ (average shortest path length + 1). The value of $N_1 \times N_2$ in Algorithm 8 becomes the square of this value.

Two networking scenarios were used as follows. In the first scenario, in the first half of the simulation, a small network size of 1,000 nodes started to grow to about 60,000 nodes. Then, in the second half, it shrank back to 1,000 nodes (Figure 7). This is to represent the general case of fluctuating network size. The network size change is implemented as asymmetric churn scenarios, where a leaving node is replaced by two new nodes for the first half of the simulation until the network size reaches 60,000 nodes, and in the second half, two node leaves are replaced by only one new node until the size goes down to the original 1,000 nodes. In the second scenario, a network of 32,000 nodes is given and stabilized first. Then half of the nodes (16,000) are removed at one time (Figure 8). This is to measure the convergence speed of the capture recapture algorithm to a large change of network size, for example, due to a large scale network failure.

Figure 7 shows the results of the network size estimation with fluctuating network size. In order to obtain an average, the simulation was repeated twenty times, in which a random node was selected and estimation was performed based on the samplings collected by that node. As seen in the figure, the average estimation follows the actual size closely. The average is slightly behind the actual as it is a little

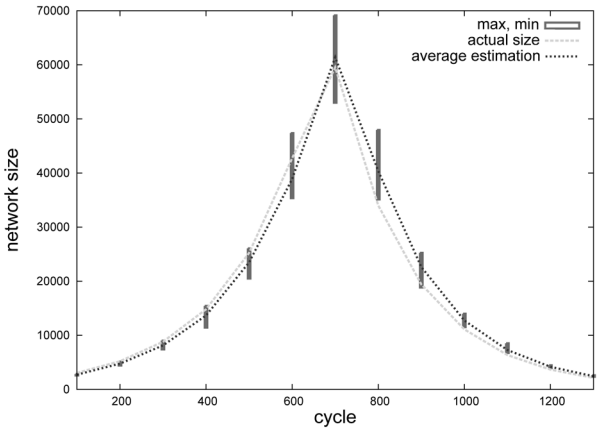


Figure 7. Network size estimation: estimates vs. actuals.

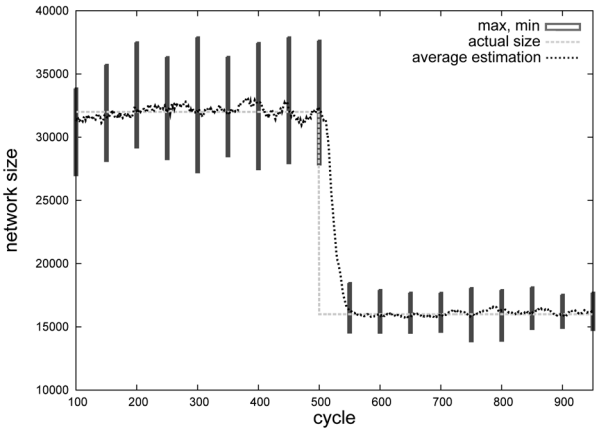


Figure 8. Network size estimation - node failures.

lower in the first half of the simulation and a little higher in the second half. The vertical bars represent the full scale of the deviation of the estimation from the actual size by twenty simulation runs.

Two questions may arise with these results. One is whether the estimation is good enough for epidemic protocols and the other is whether the network changes fast enough to reflect most realistic cases. In terms of usability for epidemic protocols, the results seem practically good enough. Because the most fundamental parameter of epidemic protocols is $log(N)$, even the most distant estimates of N in the figure in fact produce the same value of $log(N)$ as the actual N does once $log(N)$ is taken into account. This accuracy is valid as long as epidemic protocols are concerned. Whether this accuracy is extend-able to other general network management purposes is open, however. This important question needs to be answered in follow-up work in comparison with other popular methods. To the best of our knowledge, this is the first attempt to utilize a shuffle infrastructure to estimate the network size. In the

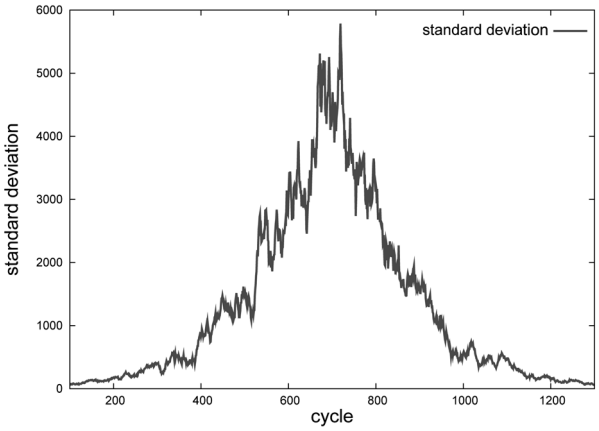


Figure 9. Network size estimation: standard deviation.

context of the second question, the network seems to change fast enough because the churn scenario uses 180 cycles as the average node lifetime. It took only 700 cycles for the initial size of 1,000 nodes to reach 60,000 nodes. If one cycle is equivalent to 1 second, it corresponds to 700 seconds, or 11.5 minutes only. This is because the average lifetime was taken based on the real measurement study [Rhea et al. 2004].

In order to see the convergence speed of the algorithm in an extreme situation, a large scale network failure was experimented (Figure 8). In this experiment, as seen in the figure, half of the nodes (16,000) are removed at the cycle 500. Then the estimate goes down toward the new network size 16,000. However, it takes a little more than 50 cycles to fully recognize the new size. Again, if one cycle is equivalent to 1 second, the algorithm takes less than a minute to learn the large scale network failure. Considering these experiments, the capture and recapture algorithm can be said to follow the actual size fairly quickly.

In order to understand the relative difference of the estimation from the mean estimation size, Figure 9 shows standard deviation of the estimation obtained from the same experiments above. The metric, as seen in the figure, goes up as the network size grows, and goes down as the size goes down. Although the distant estimates of N in Figure 7 produce the same value of $log(N)$ as the actual N does, this trend of the standard deviation being proportional to the actual network size does not seem desirable. This means that, in general cases, where more accurate estimation is required, the capture recapture algorithm may need to improve to be applied to other protocol-independent network management applications. Overall, DIMPLE-II is an effective dynamic membership protocol supporting randomness and network size estimation for epidemic protocols to be used in open systems such as P2P networks.

7. CONCLUSIONS

This paper proposes a methodology which enhances the existing shuffle algorithms, and estimates the network size fairly accurately for dynamic open systems such as P2P systems. The proposed methodology improves the quality of the node degrees

by using a repeated single-entry shuffle, and estimates the network size by utilizing the shuffle infrastructure. At each cycle, the local view is considered as part of network sampling. As shown in the simulation, the proposal helps process frequent join and leave activities efficiently and effectively, thus enhancing the dynamic membership service for large-scale systems. In addition, the estimates follow the actual network sizes closely. As a result, epidemic protocols will be able to maintain a high quality of information dissemination even with high network churn and dynamic network sizes. Also, these results will help provide a methodology to achieve self-organizability of P2P systems.

This paper solicits future work in two directions: reliable broadcasting in open dynamic systems such as P2P systems, and improvement on the proposed capturerecapture based network size estimation. Many existing work, using epidemic protocols, addressing reliable broadcasting, assume that the network size is known. To improve the understanding of the epidemic behavior as a means to achieve reliable broadcasting, the network size *should* be estimated dynamically. Depending on the quality of estimates, the effectiveness of information dissemination will vary. On the other hand, the evaluation of network size estimation given in this paper can be considered preliminary. Although it strongly suggests that the shuffle infrastructure can be easily utilized for network size estimation, possible performance optimization and comparison study with other existing methods still remain as future work.

REFERENCES

AGUILERA, M., STROM, R., STURMAN, D., ASTLEY, M., and CHANDRA, T. 1999. Matching events in a content-based subscription system. In *Proceedings of the 18th ACM Symposium on Principles of Distributed Computing (PODC '99)*. ACM, Atlanta, GA, 53–61.

ALLAVENA, A., DEMERS, A., AND HOPCROFT, J. 2005. Correctness of a gossip based membership protocol. In *Proceedings of the 24th ACM Symposium on Principles of Distributed Computing (PODC '05)*. ACM, Las Vegas, NV, 292–301.

BANSOD, N., MALGI, A., CHOI, B., AND MAYO, J. 2005. Muon: Epidemic based mutual anonymity. In *Proceedings of the 13th International Conference on Network Protocols (ICNP '05)*. IEEE, Boston, MA, 99–109.

BIRMAN, K., HAYDEN, M., OZKASAP, O., XIAO, Z., BUDIU, M., AND MINSKY, Y. 1999. Bimodal multicast. *ACM Transactions on Computer Science 17*(2):41–88.

BOLOT, J.-C., TURLETTI, T., AND WAKEMAN, I. 1994. Scalable feedback control for multicast video distribution in the internet. *SIGCOMM Comput. Commun. Rev. 24*(4):58–67.

CARZANIGA, A. 1998. Architectures for an event notification service scalable to wide-area networks. Ph.D. thesis, Politecnico di Milano.

CUENCA-ACUNA, F., PEERY, C., MARTIN, R., AND NGUYEN, T. 2003. Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities. In *Proceedings of the 12th International Symposium on High Performance Distributed Computing (HPDC '03)*. IEEE, Seattle, WA, 236–246.

DARROCH, J. 1958. The multiple-recapture census: I. estimation of a closed population. *Biometrika 45*, 343–359.

DAS, A., GUPTA, I., AND MOTIVALA, A. 2002. Swim: Scalable weakly-consistent infection-style process group membership protocol. In *International Conference on Dependable Systems and Networks (DSN '02)*. IEEE, Bethesda, MD, 303–312.

DEMERS, A., GREENE, D., HOUSER, C., IRISH, W., LARSON, J., SHENKER, S., STURGIS, H., SWINEHART, D., AND TERRY, D. 1987. Epidemic algorithms for replicated database

maintenance. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC '87)*. ACM, Vancouver, British Columbia, Canada, 1–12.

EUGSTER, P., GUERRAOUI, R., KERMARREC, A.-M., AND MASSOULIE, L. 2004. From epidemics to distributed computing. *IEEE Computer 37*(5):60–67.

EUGSTER, P., HANDURUKANDE, S., GUERRAOUI, R., KERMARREC, A.-M., AND KOUZNETSOV, P. 2001. Lightweight probabilistic broadcast. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN '01)*. IEEE, Goteborg, Sweden, 443–452.

GANESH, A.-J., KERMARREC, A.-M., AND MASSOULIE, L. 2003. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers 52*(2):139–149.

GKANTSIDIS, C., MIHAIL, M., AND SABERI, A. 2004. Random walks in peer-to-peer networks. Gupta, I., Birman, K., and Resesse, R. 2002. Fighting fire with fire: Using randomized gossip to combat stochastic scalability limits. *Journal on Quality and Reliability Engineering International 29*(8):165–184.

HOROWITZ, K. AND MALKHI, D. 2003. Estimating network size from local information. *Inf. Process. Lett. 88*(5):237–243.

JELASITY, M., GUERRAOUI, R., KERMARREC, A.-M., AND VAN STEEN, M. 2004. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*. ACM, Toronto, Ontario, Canada, 79–98.

JELASITY, M., KOWALCZYK, W., AND VAN STEEN, M. 2003. Newscast computing. *Technical Report IR-CS-006*, Vrije Universiteit Amsterdam, Department of Computer Science.

KOSTOULAS, D., PSALTOULIS, D., GUPTA, I., BIRMAN, K., AND DEMERS, A. 2005. Decentralized schemes for size estimation in large and dynamic groups. In *4th IEEE International Symposium on Network Computing and Applications (NCA)*, 41–48.

LV, Q., CAO, P., COHEN, E., LI, K., AND SHENKER, S. 2002. Search and replication in unstructured peer-to-peer networks. In *16th ACM International Conference on Supercomputing (ICS '02)*. ACM, New York, NY, 84–95.

MANE, S., MOPURU, S., MEHRA, K., AND SRIVASTAVA, J. 2005. Network size estimation in a peertopeer network. *Technical Report 05-030*, University of Minnesota, Department of Computer Science and Engineering.

MANKU, G. S. 2003. Routing networks for distributed hash tables. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*. ACM Press, New York, NY, USA, 133–142.

MASSOULIE, L., MERRER, E. L., KERMARREC, A.-M., AND GANESH, A. 2006. Peer counting and sampling in overlay networks: Random walk methods. In *International Conference on Principles Of Distributed Computing (PODC'06)*. ACM, Denver, CO, 123–132.

PORTMANN, M. AND SENEVIRATNE, A. 2003. Cost-effective broadcast for fully decentralized peertopeer networks. *Journal of Computer Communications 26*(11):1159–1167.

PSALTOULIS, D., KOSTOULAS, D., GUPTA, I., BIRMAN, K., AND DEMERS, A. 2004. Practical algorithms for size estimation in large and dynamic groups. In *PODC'04: Proceedings of the twenty-third annual symposium on Principles of distributed computing*. ACM, New Foundland, Canada, 25–28.

RHEA, S., GEELS, D., ROSCOE, T., AND KUBIATOWICZ, J. 2004. Handling Churn in a DHT. In *Proceedings of the 2004 USENIX Technical Conference, Boston, MA, USA*. USENIX, citeseer. csail.mit.edu/article/rhea04handling.html.

STAVROU, A., RUBENSTEIN, D., AND SAHU, S. 2004. A lightweight, robust p2p system to handle flash crowds. *IEEE Journal on Selected Areas in Communications 22*(7):6–17.

STUTZBACH, D. AND REJAIE, R. 2004. Towards a better understanding of churn in peer-to-peer networks. *Technical Report CIS-TR-04-06*, University of Oregon, Department of Computer Science. Nov.

SUN, J., CHOI, B. K., WEBER, P., AND KIECKHAFER, R. 2007. Dimple: Dynamic membership protocol for epidemic protocols. In *IEEE Broadnets 2007*. IEEE, Raleigh, NC, USA.

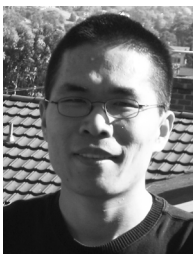
TANARAKSIRITAVORN, S. AND MISHRA, S. 2004. Evaluation of gossip to build scalable and reliable multicast protocols. *Performance Evaluation 58*(2-3):189–214.

VOULGARIS, S., GAVIDIA, D., AND STEEN, M. 2005. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management 13*, 197–217.

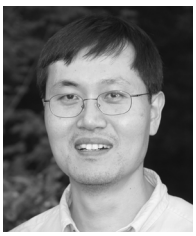
VOULGARIS, S., GAVIDIA, D., AND VAN STEEN, M. 2005. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management 13*(2):197–217.

VOULGARIS, S. AND VAN STEEN, M. 2003. An epidemic protocol for managing routing tables in very large peer-to-peer networks. In *Proceedings of the 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management (DSOM '03)*. Number 2867 in LNCS. Springer, Heidelberg, Germany.

VOULGARIS, S. AND VAN STEEN, M. 2004. Epidemic-style management of semantic overlays for content-based searching. *Technical Report IR-CS-011*, Vrije Universiteit Amsterdam. Nov.



Jin Sun received BS in Physics and MS in Electrical Engineering from Nanjing University and Shanghai University respectively in 2000 and 2003. In addition, he received MS in Computer Science from Michigan Technological University in 2007. During his graduate study at Michigan Tech, he conducted research on epidemic protocols, dynamic membership, and network size estimation. He is currently affiliated with Chicago Trading. He also worked for Huatek Software Engineering and Rocket Software in China before moving to the US.



Byung K. Choi received PhD in Computer Science from Texas A&M University in 2002. He is currently an Assistant Professor in the Department of Computer Science at Michigan Technological University. Recently he has investigated dynamic membership management for large scale peer-to-peer systems and epidemic protocols which rely on such membership management. Prior to his graduate studies he was a team leader for LG Information and Communications (LGIC) R&D Center in Korea, where he worked on various switching system designs which have been widely used in Korea.

Kwang-Mo Jung received PhD in Electronic Communication from Kwangwoon University in 2006. He is currently a researcher and manager in the Department of Ubiquitous Computing Center at Korea Electronics Technology Institute (KETI). Recently he has investigated home network systems, where many different network protocols and switching systems are integrated and interconnected through common communication protocol (CCP). Currently his research work focuses on sensor network, affective computing, and multimedia systems.