Persistence Service for Non-Persistent P2P Systems

Reaz Ahmed[1], Nashid Shahriar[2], Mahfuza Sharmin[2], Raouf Boutaba[1] and Bertrand Mathieu[3]
[1]David R. Cheriton School of Computer Science, University of Waterloo
{r5ahmed | rboutaba}@uwaterloo.ca
[2]Dept. of Comp. Sc. and Engg., Bangladesh University of Engg. and Tech.
{nshahriar | sharmin}@cse.buet.ac.bd
[3]Orange Labs, Lannion, France
bertrand2.mathieu@orange-ftgroup.com

(Technical Report: CS-2012-18)

Abstract

Ensuring content persistence with minimal replication overhead is a prerequisite for providing any consistent service over a peer-to-peer (P2P) overlay. This paper introduces S-DATA, a bandwidth efficient protocol for achieving highly available P2P systems with minimal replication overhead. When considering a global P2P system, the cyclic behavior of peers situated at different time zones can be found complementary of one another. In S-DATA, peers with complementary diurnal availability patterns collaborate in small replication groups and host each other's content in turn to ensure 24/7 availability. In this work we present a mathematical model for measuring time-based availability with $(\beta - 1)$ redundancy as a function of replication group size and peer uptime behavior. We also simulate the S-DATA protocol in the PeerSim simulator and compare its performance against a few other time-based replication protocols.

1 Introduction

Since its inception, peer-to-peer (P2P) technology has been applied for numerous distributed applications, including file sharing, distributed computing, multi-player gaming, media streaming and instant messaging. None of these applications require or assume a persistent service guarantee from the underlying P2P overlay. Yet there exists other applications like web hosting, online backup, content distribution *etc.*, that require persistence in resource/service availability. P2P systems rely on commodity machines, voluntarily participating at network edge. As a result, it is challenging to use P2P technology for deploying any application that requires persistent resource/service availability.

Existing proposals in P2P systems use replication as the primary means for increasing resource availability. Replication strategies in P2P systems can be broadly classified as time-based replication and quantitative replication. In quantitative replication approaches availability is ensured by consistently maintaining a fixed number of replicas per resource. On the other hand, time-based replication approaches utilize a peer's uptime history to reuse a replica from the peer's previous session.

In quantitative replication approaches content availability is proportional to the number of its replicas. But, increasing the number of replicas has a number of side effects. First, it incurs increased network overhead for replica placement and update propagation between the replicas when the original content is updated. Second, storage overhead increases linearly with the number of replica. Third, it requires additional mechanisms for keeping track of the replicas for efficient query forwarding. And last but not the least, query load balancing among the replicas of a specific content becomes an important issue from fairness point of view. Existing availability approaches ([1, 2, 3, 4]) that solely rely on replication are either bandwidth hungry or require complex predictive knowledge for replica updates and relocation. These approaches frequently burden the peers with longer uptime, which results into a skewed load distribution and a negative impact on availability.

Time-based replication strategies, on the other hand, utilize daily uptime behavior of the peers to replicate a content. Cyclic diurnal pattern in peer availability has been observed in a number of previous studies including [5, 6, 7, 8]. Rzadca et al. [9] have shown that diurnal behavior of peers can be a useful characteristic for improving availability if the system has a truly global scope. For example, consider two peers separated by 12-hours difference in time zone. They will exhibit complementary availability patterns, if both of them remain online during daytime and off-line at night. Even for the peers located in same time zone, mutually exclusive availability pattern may be observed due to their Internet usage habits or job nature.

In a P2P network of a million peers, it is a challenging problem to match and tie peers in small groups in such a way that the following conditions hold. First, group size should be as small as possible. Second, at any given time, at least one peer (or a pre-specified number of peers) should be available within a group. Third, the group formation process should be globally optimized and should not incur significant network overhead. Existing time-based availability approaches ([10, 9, 11]) are rely on unstructured, gossip-based protocol and do not deliver guarantee on the the above mentioned requirements.

In this paper, we propose a globally optimized and efficient protocol, named *S-DATA (Structured approach for Diurnal Availability by Temporal Assemblage)* that maximizes 24/7 content availability in a P2P network, while minimizing the aforementioned shortcomings of the existing time-based availability schemes. Our contributions in this work can be summarized as follows:

- We have applied the Plexus protocol [12] (in Section 2) for constructing globally optimal time-based replication groups without incurring significant network or storage overhead.
- We have provided a generic mechanism in Section 3 for combining time-based replication with quantitative replication, which ensures β-availability. By β-availability mean that β peers from a group will be available at any given time.
- In Section 4 we have formulated a mathematical model for measuring β-availability as a function of peer-uptime duration and replication group size.
- The availability architecture and mathematical models presented in this work have been validated in Section 5 using extensive simulation on real world trace data.

In Section 6, we present the related works on P2P availability and finally we conclude in Section 7.

2 Conceptual Overview

2.1 Architecture

As depicted in Fig. 1, S-DATA architecture evolves around three conceptual components: *replication group*, *Group Index Overlay* (GIO) and *Content Index Overlay* (CIO). Replication groups provide a persistent storage by exploiting diurnal uptime-behavior of regular peers, GIO maintains peers' and groups' availability information, while CIO retains an indirect mapping from content name to content location. In the following we explain each of these three components.

Replication group : In S-DATA, peers are clustered into small groups based on their diurnal availability pattern. Within a replication group, peers have mutually exclusive uptime with little overlap. In a replication group with β-availability, it is ensured that at least β members from that group will be online at any given time. All members within a group replicate each others content and work as proxy for off-line members of that group.

Group index overlay : It has two functions. First, during group formation, it works as a distributed agent for match-making peers with complementary uptime behavior. Second, it acts as an indirection structure during content lookup. Initially each peer advertises its availability pattern as a bit-vector to this overlay. During group formation, peers willing to form a group search for other peers (or groups) having complementary uptime behavior. To the best of our knowledge, Plexus is the only Distributed Hash Table (DHT) technique that supports approximate bit-vector

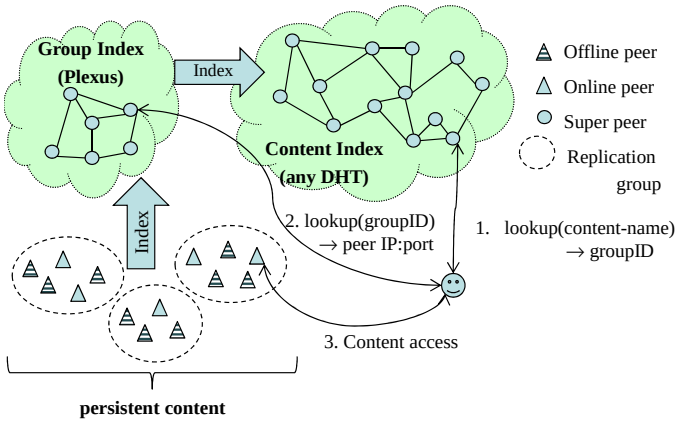


Figure 1: Conceptual Architecture of S-DATA

matching in an efficient manner. Hence we used Plexus as the indexing and routing protocol for GIO. At any given time, this overlay maps a group ID to one (or β) online peer from that group.

Content index overlay : This overlay can be implemented using any DHT-technique depending on application-specific requirements. This overlay maps a content name to a group ID. In order to search and download a content, a peer will first search the CIO and discover a group ID. Then it will lookup the group ID in the GIO and find the location (IP:port) of an alive peer currently hosting that content and download it. Mapping a content name to a group ID, instead of directly mapping to a peer ID incurs an additional lookup. But, this lookup is necessary to facilitate dynamically associate a content name to the currently online peer hosting that content.

From uptime point of view, we assume that the peers in replication groups are regular peers with moderate online time (4-8 hours) on a daily basis. While the peers in the indexing overlays are superpeers with longer uptime, higher communication bandwidth and storage capacity.

2.2 Plexus routing

Before diving into the details of S-DATA, we present a brief overview of the Plexus protocol for better understanding of this work. Plexus is a DHT approach that enables Hamming distance-based approximate matching between an advertisement and a query, which are represented as n-dimensional bit-vectors (or n-bit patterns) in a Galois Field F_2^n. Plexus uses $< n, k, d >$ linear binary code for partition the pattern space (i.e., F_2^n) and message routing. Here, k is the dimensionality of the code - meaning there are 2^k codewords and d is the minimum Hamming distance between any to codewords. In simple words, a pattern (query or advertisement) is mapped to the codewords, closest in Hamming distance. Each peer in Plexus overlay, is assigned a set of codewords based on its prefix-bits. For advertisement or query, the closest codewords (in Hamming distance) are computed and the message is multicasted to the peers responsible for those codewords. Routing in Plexus is based on the *generator matrix* of the linear binary code. Upper bound for routing between any two peers is $\frac{k}{2}$, which is logarithmic on network size. Additionally, Plexus has a built-in mechanism of index replication for improved reliability. In S-DATA we use the Hamming distance-based approximate matching for group formation and index-replication for improved reliability.

2.3 Availability Vector

The traditional definition of peer availability is simply measured by the fraction of time a peer is online [2] within a certain time period. If a peer joins and leaves m times during a period of T hours, and every time remains up for t_k hours, then its availability can be computed as, $\frac{\sum_{k=1}^{m} t_k}{T}$. This formula does not take the diurnal availability pattern in peer uptime behavior into account. This fact has been mathematically proven by Yang et al. in [13].

In this work, we divide 24-hours of a day in K equal-length time-slots *w.r.t.* GMT+0, and estimate the probability

of a peer being online in each time-slot based on its historical behavior. Thus the availability of a peer, say x, is defined as $\mathcal{A}_x = \{a_{x1}, a_{x2}, ..., a_{xk}, ..., a_{xK}\}$, where $\mathcal{A}_x$ is the K-dimensional availability vector for peer x, and a_{xk} is the probability of peer x being online in slot k.

The responsibility of computing and maintaining the availability vectors can be dedicated to the P2P client software or to GIO. Each of these alternatives has its own merits and demerits, and can be considered as an implementation specific choice. Computing and maintaining availability vectors at the client software will give more accurate estimates and will generate minimal network traffic. However, a client software can be maliciously modified to report a fake availability vector. Alternatively, the availability vectors can be computed and maintained at GIO. This approach can generate more reliable probability values of the availability vectors, though at the expense of increase network traffic and decreased accuracy of the computed availability.

3 S-DATA Protocol Details

3.1 Terminology

In S-DATA we use four indexes (see Table 1) for group formation and content lookup. $\mathcal{I}_e$ represents an indexing peer in CIO which is responsible for storing the ID of e (ID_e), where e can be a regular peer or a group. $\mathcal{I}_e$ works as e's proxy for meta-information exchange. For a regular peer, say x, $\mathcal{I}_x$ stores an $\mathcal{M}_x$ record, which contains the availability vector (A_x), ID (ID_x) and network location (Loc_x) for x, as well as the group ID (ID_{G_x}) and index location ($\mathcal{I}_{G_x}$) for x's group G_x. For a group G, $\mathcal{I}_G$ contains index record $\mathcal{N}_G$, which contains group availability vector ($\mathcal{A}_G$), group ID (ID_G), and for each member x of G, its ID (ID_x), index location ($\mathcal{I}_x$) and network location (Loc_x). To enable approximate matching between peers' and groups' availability vectors, we maintain $\mathcal{V}_e$ indexes that contain availability pattern (S_e, explained in Section 3.2.1), availability vector ($\mathcal{A}_e$), ID (ID_e) and index location ($\mathcal{I}_e$) for e. $\mathcal{V}_e$ is stored in all peers $\mathcal{L}_e$ within a pre-specified Hamming distance from S_e. Finally, for content lookup another set of indexes ($\mathcal{K}_w$) is maintained in CIO. For each keyword w attached to a content an index ($\mathcal{K}_w$) is stored in CIO at peer $\mathcal{J}_w$, which is responsible for keyword w. $\mathcal{K}_w$ retains the content's ID (ID_{doc}), other keywords describing the content ($\{w_i\}$), group ID (ID_G) and index location ($\mathcal{I}_G$) of the group that hosts the content.

Table 1: List of Indexes in S-DATA

Name	Overlay	Indexed information
$\mathcal{M}_x$	GIO/$\mathcal{I}_x$	$< \mathcal{A}_x, ID_x, Loc_x, ID_{G_x}, \mathcal{I}_{G_x} >$
$\mathcal{N}_G$	GIO/$\mathcal{I}_G$	$< \mathcal{A}_G, ID_G, \{< ID_x, \mathcal{I}_x, Loc_x > \mid x \in G\} >$
$\mathcal{V}_e$	GIO/$\mathcal{L}_{S_e}$	$< S_e, \mathcal{A}_e, ID_e, \mathcal{I}_e >$
$\mathcal{K}_w$	CIO/$\mathcal{J}_w$	$< ID_G, \mathcal{I}_G, ID_{doc}, \{w_i \mid w_i \in doc\} > \mathcal{L}_{\acute{S}_x}$

3.2 Indexing Availability Information

To cluster regular peers in globally optimized replication groups, we need to index each peer's availability information ($\mathcal{V}_e$) to GIO. This indexing process involves two steps: i) encoding availability vector ($\mathcal{A}_e$) to bit-vector (S_e) and ii) advertisement using Plexus protocol. These two steps are explained in the following.

3.2.1 Availability Vector Encoding

It can be easily seen that the availability vector $\mathcal{A}_i$ is a K-dimensional vector of uptime probabilities, whereas the advertisement (or query) patterns in a Plexus network built on an $< n, k, d >$ code are n-bit values. Hence, we need a means to encode a K-dimensional availability vector into an n-bit pattern.

In this work we have used $K = 24$ slots for availability vector. While for Plexus implementation, we have used the $< 24, 12, 8 >$ Extended Golay Code G_{24}. Trivially, we can directly encode each probability value a_{ik} in $\mathcal{A}_i$ to one-bit in the 24-bit advertisement (or query) pattern. We can use a threshold, say θ, and can set the k-th bit of the 24-bit encoded pattern to 1 if $a_{ik} > \theta$. Unfortunately, this encoding will incur significant information loss and will degrade approximate matching performance in Plexus network.

Alternatively, we use a better encoding scheme based on the observation that consecutive values in the availability vector are usually similar in magnitude. To exploit this observation, we average the probability values in two adjacent slots and obtain a 12-dimensional availability vector $\acute{\mathcal{A}}_i = \{\acute{a}_{i1}, \acute{a}_{i2}, \ldots \acute{a}_{i12}\}$, where $\acute{a}_{ij}$ is computed as $\acute{a}_{ij} = \frac{(a_{i(2j-1)} + a_{i(2j)})}{2}$. Now, we encode each $\acute{a}_{ij}$ into two bits in the 24-bit advertisement pattern as follows. $\acute{a}_{ij}$ is encoded to 00 if $\acute{a}_{ij}$ is less than $\frac{1}{3}$. If $\acute{a}_{ij}$ is between $\frac{1}{3}$ and $\frac{2}{3}$ then the encoding is 01. Otherwise, $\acute{a}_{ij}$ is greater than $\frac{2}{3}$ and is encoded to 11. This encoding reflects the numeric distance in $\acute{a}_{ij}$ to the Hamming distance in advertisement patterns.

3.2.2 Advertisement

An advertising peer, say x, first computes the n-bit advertisement pattern, say S_x, as explained above. Then x sends the tuple $< \mathcal{S}_x, \mathcal{A}_x, ID_x, Loc_x, ID_{G_x}, \mathcal{I}_{G_x} >$, to $\mathcal{I}_x$. If x has not formed a group then ID_{G_x} and $\mathcal{I}G_x$ will be empty. Upon receiving the advertisement message $\mathcal{I}_x$ computes the codewords within a pre-specified Hamming distance from S_x and uses Plexus routing to route and index the advertisement ($\mathcal{V}_x$) to the peers ($\mathcal{L}_{S_x}$) responsible for these codewords.

3.3 Group Formation

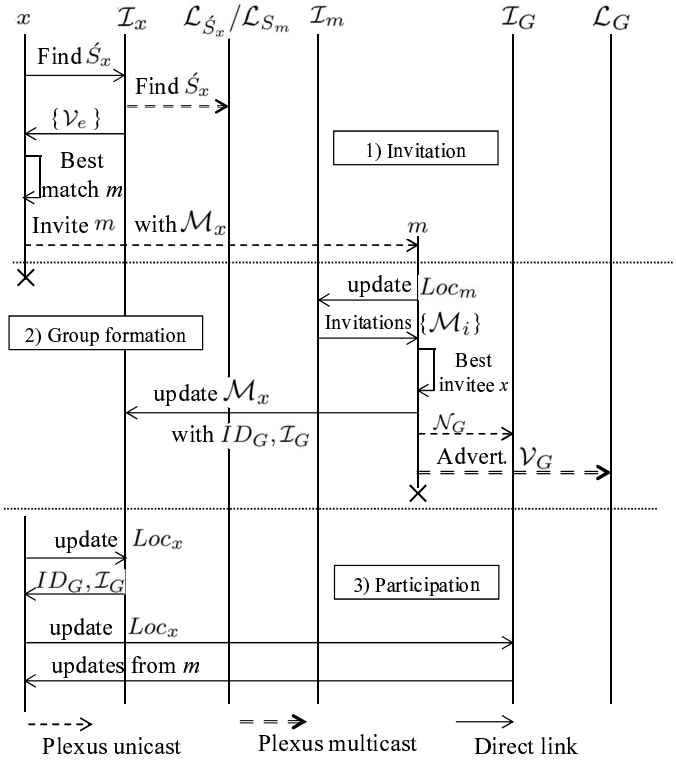


Figure 2: Sequence diagram show group formation of x with m

This process lies at the core of S-DATA protocol. Our target is to cluster peers into groups in such a way that the group sizes are minimal and at any given time at least $\beta \geq 1$ peers from a group is online with the highest possible probability.

The most challenging part of this process is to relay group formation messages between peers that may not be simultaneously online. To this end, we use GIO as a message relay. Fig. 2 presents a sequence of message exchanges between indexing peers in GIO and regular peers x and m while forming a 1-availability group G. It is worth noting that x and m are not online simultaneously and hence they have no direct message exchange. The Group formation process is composed of the following three steps:

1. *Invitation* : We assume that on average a regular peer will be online for L time-slots on a daily basis. It will be the responsibility of a peer to maintain β peers in its group during the L-slots it is online and the next L-slots. To find a suitable peer that can improve group's availability for the next L-slots, peer x computes an availability pattern $\acute{S}_x$. $\acute{S}_x$ has bits $t+L+1$ to $t+2L$ set to 1, assuming that the availability pattern S_x of peer x has bits t to $t+L$ set to 1. Once $\acute{S}_x$ is computed, peer x forwards it to $\mathcal{I}_x$. $\mathcal{I}_x$ uses Plexus multi-cast routing to find the peers ($\mathcal{L}_{\acute{S}_x}$) in CIO responsible for indexing peer/group availability records ($\mathcal{V}_e$) similar to $\acute{S}_x$. From the availability records ($\mathcal{V}_e$) returned by $\mathcal{I}_x$, peer x selects the most appropriate peer, say m, that maximizes its groups availability. A mathematical model for selecting the most appropriate peer has been presented in Section 4.3. Peer x locates the indexing peer ($\mathcal{I}_m$) for m using Plexus routing and sends an invitation request to $\mathcal{I}_m$ that includes the $\mathcal{V}_x$ record.

2. *Group formation* : Upon becoming online m updates $\mathcal{I}_m$ with its new network location (Loc_m). In response $\mathcal{I}_m$ sends all the invitations ($\{\mathcal{V}_e\}$) for m that has been accumulated during m's offline period. Among these invitations, m selects the best candidate x. If x is already a member of an existing group then m simply joins the group otherwise it creates a new group G. To create or update the group index in GIO, m may require to transmit three messages: a) if m created a new group, then it has to update the $\mathcal{M}_x$ record in $\mathcal{I}_x$ so that x can learn about G upon returning; b) m has to index ($\mathcal{V}_G$) to all peers ($\mathcal{L}_G$) within a certain Hamming distance from S_G; c) finally, m has to store the group index $\mathcal{N}_G$ to $\mathcal{I}_G$.

3. *Participation* : During its next online session peer x will update I_x with its new network location Loc_x. If the previous invitation from x was honored by m then I_x responds with the newly formed group's information (ID_G and $\mathcal{I}_G$). x updates $\mathcal{I}_G$ with its location information Loc_x and $\mathcal{I}_G$ responds with any update from m or other members of G. On the other hand, if the invitation from x was not accepted by m, then x has to restart the group formation process with the next best matching peer, other than m.

The above mentioned process of forming 1-availability group can be easily extended to construct β-availability groups. Two modifications in Step 1 of the above process are required. First, x should be the highest ID peer among the online members of its group (G_x). And second, x should send invitations to $\beta - f$ peers simultaneously, where f is the number of peers in x's group who shall be online in the L-time slots following the online period of x.

3.4 Group Maintenance

The diurnal availability pattern of a peer may change over time. In such a situation a peer, say x, may want to change its group. Group changing involves leaving the the current group and joining a new group. The process of joining a group has been described in the Section 3.3. To leave its current group G_x, peer x has to update two peers in CIO. First, x has to remove its index information from $\mathcal{N}_{G_x}$ record, which is stored in peer $\mathcal{I}_{G_x}$. And second, x has to clear the ID_{G_x} and $\mathcal{I}_{G_x}$ fields in $\mathcal{M}_x$ record, which is stored in $\mathcal{I}_x$. It should be noted that we use soft-state registration for advertising $\mathcal{V}_x$ records to $\mathcal{L}_{S_x}$. Hence, the $\mathcal{V}_x$ records will be automatically removed from the peers in $\mathcal{L}_{S_x}$, if x does not re-advertise before the previous advertisement expires.

3.5 Content Indexing and Lookup

In the following we describe the mechanisms for content indexing and lookup.

3.5.1 Content Indexing

Traditionally a content in a P2P network is tagged with a set of descriptive keywords, ($w \in \{w_i\}$). These keywords are used to locate the peers(s) ($\mathcal{J}_w$) in CIO for storing the $\mathcal{K}_w$ record. While advertising a content a peer, say x, may or may not be a member of a replication group. If x is a member of a replication group, say G_x then ID_{G_x} and $\mathcal{I}_{G_x}$ are stored in $\mathcal{K}_w$ record, otherwise ID_x and $\mathcal{I}_x$ are used. However, $\mathcal{K}_w$ is not updated when x forms a group. Rather, $\mathcal{K}_w$ is updated in a reactive manner during content lookup. This process is described in the following section.

3.5.2 Content Lookup

A query for keyword w will be routed to $\mathcal{J}_w$ using the routing protocol in CIO. Based on the information found in $\mathcal{K}_w$, the query will be forwarded to either $\mathcal{I}_{G_x}$ if the content host x has formed a group and $\mathcal{K}_w$ has been updated, or the query will be forwarded to $\mathcal{I}_x$. In a regular scenario, the query will be forwarded to $\mathcal{I}_{G_x}$ and the location Loc_y of the currently alive peer y in G_x will be return to the querying peer via $\mathcal{J}_w$. On the other hand, if x has formed a group but $\mathcal{K}_w$ has not been updated, then $\mathcal{J}_w$ will contact $\mathcal{I}_x$, which will respond with ID_{G_x} and $\mathcal{I}_{G_x}$. Accordingly, $\mathcal{J}_w$ will update $\mathcal{K}_w$ for future references. Finally, $\mathcal{J}_w$ will contact $\mathcal{I}_{G_x}$ to obtain the location (Loc_y) of the currently active peer (y) in G_x.

4 Mathematical Model

In this section we first present a mathematical model for calculating β-availability. Then we present a model for expressing β-availability as function of average uptime and group size. Finally, we present a utility function for computing the relative gain in β-availability that can be achieved by combining two peers (or groups) in a group.

4.1 Defining β-Availability

As presented in Section 2.3, we express the availability of peers x as K dimensional vector $\mathcal{A}_x = \{a_{x1}, a_{x2}, \ldots a_{xK}\}$, where a_{xk} represents the probability of peer x being online in time-slot k. When peers collaborate in a replication group, say G, we can model the 1-availability vector of the group as $\mathcal{A}^1_G = \{a^1_{G1}, a^1_{G2}, \ldots, a^1_{GK}\}$. The combined probability of at least one peer being online at any given slot k can be computed as follows:

$$\begin{aligned} a^1_{Gk} &= Pr[\textit{at least one member is online in slot } k] \\ &= 1 - Pr[\textit{no peer is online in slot } k] \\ &= 1 - \prod_{\forall x \in G} (1 - a_{xk}) \end{aligned} \tag{1}$$

We can extend the equation for a^1_{Gk} to compute 2-availability, *i.e.*, the probability of at least 2 peers being online, at slot k as follows:

$$\begin{aligned} a^2_{Gk} &= Pr[\textit{at least } 1 \textit{ members is online at slot } k] \\ &\quad - Pr[\textit{exactly } 1 \textit{ member is online at slot } k] \\ &= a^1_{Gk} - a^{\textit{exactly } 1}_{Gk} \end{aligned}$$

Here $a^{\textit{exactly } 1}_{Gk}$ is the probability of exactly one member of G being online at slot k. This can be computed as:

$$a^{\textit{exactly } 1}_{Gk} = \sum_{\forall x \in G} a_{xk} \prod_{\forall y \in G, y \neq x} (1 - a_{yk}) \tag{2}$$

In a similar manner, we can generalize the 2-availability equation to compute β-availability at any slot k as follows:

$$\begin{aligned} a_{Gk}^{\beta} &= a_{Gk}^{(\beta-1)} - a_{Gk}^{exactly\ (\beta-1)} \\ &= a_{Gk}^{(\beta-2)} - a_{Gk}^{exactly\ (\beta-2)} - a_{Gk}^{exactly\ (\beta-1)} \\ &= a_{Gk}^{1} - \sum_{j=1}^{\beta-1} a_{Gk}^{exactly\ (j)} \end{aligned} \tag{3}$$

Now we can average the slot-wise availability values to obtain an estimated β-availability of the group over time as follows:

$$|\mathcal{A}_G^{\beta}| = \frac{1}{K} \sum_{k=1}^{K} a_{Gk}^{\beta} \tag{4}$$

4.2 Computing β-availability

In this section we develop a mathematical model to establish the relationship between β-availability, group size and uptime distribution. Without any loss of generality we assume that each peer is online for L consecutive slots of a day with high probability, while its probability of being online for the rest of the slots is very low. Following the finding of Bustamante et al. in [14], we model the duration of a peer's online session, *i.e.*, L, using the Pareto distribution with shape parameter α. According to this distribution the expected uptime can be computed as $L = \frac{\alpha}{1-\alpha}$.

Apart from modeling uptime distribution, we have to model the positive and negative correlation between the probabilities of a peer being online between consecutive time slots. To model this we have used the short tailed Cauchy distribution, which can represent the correlation between consecutive probability values well. Accordingly, we partition the availability vector into high and low regions based on uptime L. For peer x, a_{xk} in any slot k in the high availability region can be computed as follows:

$$a_{xh[i]} = s * \frac{\gamma}{\pi[(i * \delta_h)^2 + \gamma^2]} \tag{5}$$

where, $\delta_h = \frac{h}{L}$ and $i = 0$ to $L - 1$.

Similarly, we can compute a_{xk} in any slot k in the low availability region as follows:

$$a_{xl[i]} = s * \frac{\gamma}{\pi[(i * \delta_l + p)^2 + \gamma^2]} \tag{6}$$

where $\delta_l = \frac{l}{K-L}$ and $i = 1$ to $K - L$. Here, h, l, s and p are constants, which can be manipulated to model different uptime behaviors.

Suppose group G is composed of peers, $x, y \ldots z$. We pick a peer, say x, having peak slot in the middle of the first L slots. Then we compute its availability vector, $\mathcal{A}_x = \{a_{x1}, a_{x2} \ldots a_{xK}\}$ using Equation (5) and Equation (6). We get K Equations similar to Equations 5 and 6 corresponding to each of the K slots. We pick another peer y, whose peak slot is at the middle of the second L slots and get another K equations corresponding to the vector, $\mathcal{A}_y = \{a_{y1}, a_{y2} \ldots a_{yK}\}$ in a similar way. In this way, we pick a peer z having the peak slot at the middle of the $\frac{K}{L}$th L slots and obtain $\mathcal{A}_z = \{a_{z1}, a_{z2} \ldots a_{zK}\}$. Using these availability vectors we can find β-availability at each slot. It should be noted that the β-availability of a group depends on L, while L depends on α. According to the recurrence relation in Equation (3), finding β-availability requires computation of 1-availability, 2-availability and so on. Therefore, as the first step of β-availability computation, we present a method to calculate 1-availability $= 1 - (1 - a_{xi})(1 - a_{yi}) \ldots (1 - a_{zi})$ at any slot i in the following.

We can generalize Equation (5) and 6 as

$$f(w) = s * \frac{\gamma}{\pi[(w)^2 + \gamma^2]}$$

Table 2: 1-availability at different slots for peers of same group

Peak at	slot 1	slot L	slot $2L$	slot $3L$	slot kL
1^{st} L slots	$f(-\frac{L}{2}.\delta_h)$	$f(\frac{L}{2}.\delta_h)$			
2^{nd} L slots	$f(-\frac{L}{2}.\delta_h - L\delta_l)$	$f(-\frac{L}{2}.\delta_h)$	$f(\frac{L}{2}.\delta_h)$		
3^{rd} L slots	$f(-\frac{L}{2}.\delta_h - 2L\delta_l)$	$f(-\frac{L}{2}.\delta_h - L\delta_l)$	$f(-\frac{L}{2}.\delta_h)$	$f(\frac{L}{2}.\delta_h)$	
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
i^{th} L slots	$f(-\frac{L}{2}.\delta_h - (i-1)L\delta_l)$	$\cdots$	$\cdots$	$\cdots$	$f(-\frac{L}{2}.\delta_h - kL\delta_l)$

Table 3: 1-availability at kth slot for peers of same group

Peer at	k^{th} slot
1^{st} L slots	$f(-\frac{L}{2}.\delta_h + k\delta_h)$
2^{nd} L slots	$f(-\frac{L}{2}.\delta_h - (L-k)\delta_l)$
3^{rd} L slots	$f(-\frac{L}{2}.\delta_h - (2L-k)\delta_l)$
$\cdots$	$\cdots$
i^{th} L slots	$f(-\frac{L}{2}.\delta_h - ((i-1)L-k)\delta_l)$

where, w can represent either a high or a low availability slot. As depicted in Table 2 for an ideal scenario, a group will have $\frac{K}{L}$ peers and the peak of the 1^{st} peer will align with the 1^{st} L slots, peak of the 2^{nd} peer will align with the 2^{nd} L slots and so on. Table 3, on the other hand, presents 1-availability of peers from the same group at k^{th} slot. From these two tables we can find a^1_{G1} a^1_{Gk} for $k < L$ as follows:

$$\begin{aligned} a^1_{G1} &= 1 - (1 - f(-\frac{L}{2}.\delta_h))(1 - f(-\frac{L}{2}.\delta_h - L\delta_l))\dots \\ &= 1 - \prod_{i=1}^{\frac{K}{L}}(1 - f(-\frac{L}{2}.\delta_h - (i-1)L\delta_l)) \end{aligned}$$

And similarly,

$$\begin{aligned} a^1_{Gk} =& 1 - (1 - f(-\frac{L}{2}.\delta_h + k\delta_h)) \\ & * \prod_{i=2}^{\frac{K}{L}}(1 - f(-\frac{L}{2}.\delta_h - ((i-1)L-k)\delta_l)) \end{aligned} \tag{7}$$

To get a_{Gk} for all $k = 1$ to K, a more generalized formulation is required. Moreover, the best case scenario, as presented in Tables 2 and 3, will not be found in a real situation. Considering these two factors we average slot availability values for high and low regions separately. We use a_h and a_l to denote the average availability at high and low regions, respectively. Within a group these two average values will be same for all the peers. Hence we have not added the peer name in a_h or a_l. Evidently, we can obtain a_h and a_l by integrating the slot-wise availability values presented in Equations 5 and 6, respectively as follows:

$$a_h = \frac{2}{L}\int_0^{L/2} a_{h[i]}di = \frac{2s}{h\pi}\arctan\frac{h}{2\gamma}$$

$$a_l = \frac{2}{K-L}\int_0^{\frac{K-L}{2}} a_{l[i]}di = \frac{2s}{l\pi}\Big(\arctan\frac{l+2p}{2\gamma} - \arctan\frac{p}{\gamma}\Big)$$

Now, using a_h and a_l in Equation 1, we get a simplified form of 1-availability: $a^1_{Gk} = 1 - (1-a_h)(1-a_l)^{|G|-1}$. Here $|G|$ is the number of members in group G. To obtain β-availability in terms of a_h and a_l, we have to find the values of $a^{exactly\ j}_{Gk}$ for all $j = 1$ to $\beta - 1$ as follows.

$$
\begin{aligned}
a_{Gk}^{exactly\ 1} &= \sum_{\forall x \in G} a_{xk} \prod_{\forall y \in G, x \neq y} (1 - a_{yk}) \\
&= a_h(1-a_l)^{|G|-1} + \sum_{g=1}^{|G|-1} a_l(1-a_h)(1-a_l)^{|G|-2} \\
&= a_h(1-a_l)^{|G|-1} + (|G|-1)a_l(1-a_h)(1-a_l)^{|G|-2}
\end{aligned}
$$

$$
\begin{aligned}
a_{gk}^{exactly\ 2} &= \sum_{\forall P_x, P_y \in G, x \neq y} a_{xk}a_{yk} \prod_{\forall P_z \in G, x \neq y \neq z} (1 - a_{zk}) \\
&= a_h^2(1-a_l)^{|G|-2} + a_h a_l (1-a_h)(1-a_l)^{|G|-3} \\
&+ \sum_{g=1}^{|G|-2} a_l^2(1-a_h)^2(1-a_l)^{|G|-4} \\
&= \sum_{m=0}^{1} a_h^{1-m} a_l^m (1-a_h)^m (1-a_l)^{|G|-2-m} \\
&+ (|G|-2)a_l^2(1-a_h)^2(1-a_l)^{|G|-4}
\end{aligned}
$$

Therefore, the probability of exactly j member of G to be available at k^{th} slot can be computed as:

$$
\begin{aligned}
a_{gk}^{exactly\ j} &= \sum_{m=0}^{j-1} a_h^{j-m} a_l^m (1-a_h)^m (1-a_l)^{|G|-j-m} \\
&+ (|G|-j)a_l^3(1-a_h)^3(1-a_l)^{|G|-6}
\end{aligned}
$$

Now, using the above formulas, we can obtain the simplified form of β-availability from Equation 3:

$$
\begin{aligned}
a_{gk}^{\beta} &= a_{gk}^1 - \sum_{j=1}^{\beta-1} a_{gk}^{exactly\ j} \\
&= 1 - (1-a_h)(1-a_l)^{|G|-1} \\
&- \sum_{j=1}^{\beta-1} \sum_{m=0}^{j-1} a_h^{j-m} a_l^m (1-a_h)^m (1-a_l)^{|G|-j-m} \\
&- \sum_{j=1}^{\beta-1} (|G|-j)a_l^j (1-a_h)^j (1-a_l)^{|G|-2j}
\end{aligned} \tag{8}
$$

Equation 8 establishes a relationship between β-availability, uptime and group size. To solve it analytically, we first estimate the average availabilities, *i.e.*, a_h and a_l, using the constant values: $\gamma = 1$, $s = 2$, $h = 1$, $\alpha = 1.5$, $p = 0.5$ and $l = 4$. These values gives the best fit for the real-trace data that can be found in [15] and [16]. Putting the values of a_h and a_l in Equation 8, we can estimate slot-wise availability for different values of β and group-size. The resulting β-availability curves are given in Fig. 3, which affirms that larger groups are required to attain a given level of β-availability for higher values of β.

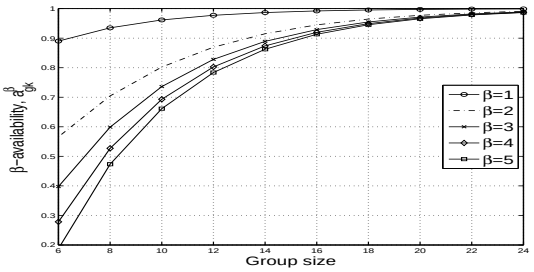


Figure 3: Estimation of β-availability

4.3 Peer Selection Metric

During the group formation process, as presented in Section 3.3, a peer needs to select the most appropriate candidate for group formation from a set of peers or groups with the desired availability pattern. We define $C_{x,y}$ to be the combined gain that can be achieved by placing peer (or group) x and peer (or group) y in a new group, say G. $C_{x,y}$ can be obtained by adding the individual gains $U_{x,G}$ and $U_{y,G}$ for the participants x and y, respectively. Equations for computing combined gain $C_{x,y}$ and individual gain $U_{m,G}$ are given below.

$$C_{x,y} = \frac{(U_{xG} + U_{yG})}{|G_x \cup G_y|} \tag{9}$$

where,

$$U_{m,G} = \sum_{k=1}^{K} (a^{\beta}_{Gk} - a^{\beta}_{mk})$$

To reduce replication overhead, we want groups to be as small as possible. We incorporate this constraint in $C_{x,y}$ computation, by placing the new group size, *i.e.*, $|G| = |G_x \cup G_y|$, as dominator in Equation (9).

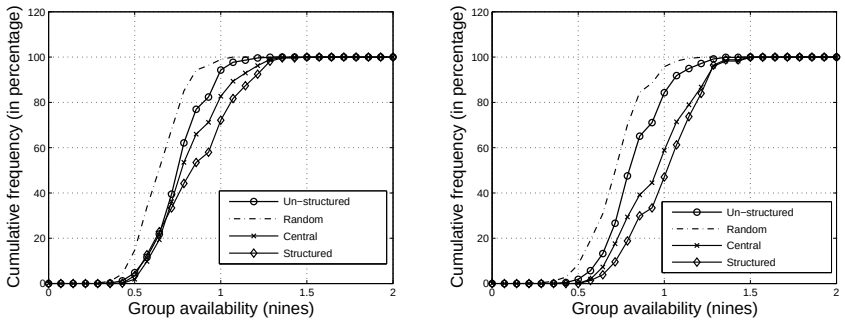
5 Experimental Results

5.1 Experiment Setup

We used the Peersim [17] simulator for the experiments. To demonstrate the effectiveness of our proposed method, we implement and compare performance of the following four grouping strategies:

1. *Structured*: Here we implement the S-DATA protocol on a Plexus network deployed using the Extended Golay Code G_{24} as described in Section 3.
2. *Unstructured*: We use the gossip protocol as proposed in [10] in this strategy, where peers reply on their local knowledge for group formation.
3. *Random*: In this strategy, a peer randomly invites a peer within two hop neighborhood without using any selection metric. The invited peer then decides to accept or deny it according to a random toss.
4. *Central*: In this scheme, a central entity, called Oracle, stores the availability vectors for all peers in the system. Alive peers communicate with the Oracle to select and invite the best matching peer according to Equation (9). Oracle chooses the best invitee, from the invitations for each peer and forms a group. This approach yields a benchmark for the other distributed schemes.

Using the trace data available in [15] and [16], we cannot manipulate peer population arbitrarily. Rather, we use those traces to model the availability vectors as outlined in Section 4.2. In line with the scope of this work, we



(a) $L = 3$ (b) $L = 6$

Figure 4: Group availability measurements

design the experiments around GIO and replication groups, and deliberately do not simulate CIO. Conforming to the finding in [18], around 18% regular peers with best uptime distribution are promoted to superpeer and they collaborate in GIO using the Plexus protocol.

5.2 Evaluation Metrics

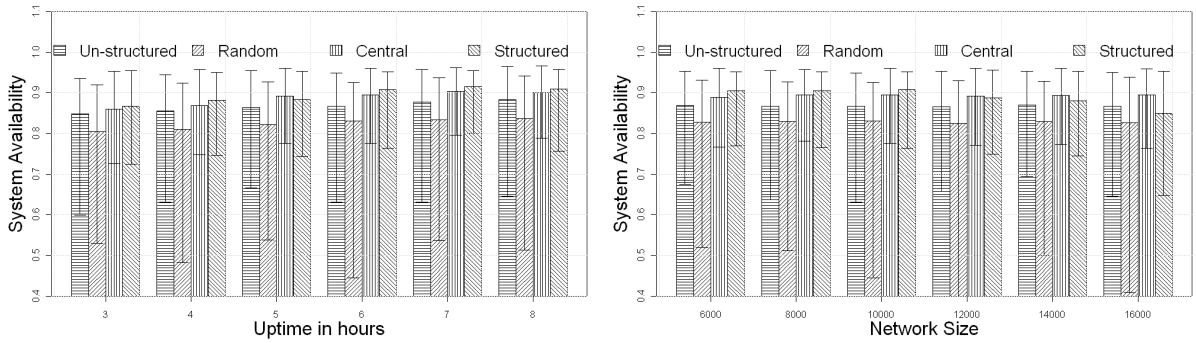
We use the following performance metrics to compare the above mentioned grouping strategies.

- *Group availability*: We consider a group available in a slot, if at least one member of the group remains up in that slot. Group availability is measured in units of nines [19], defined as $-log_{10}(1-T)$, where T is the fraction of the total observed time when groups are available. For instance, a group availability of 2 nines implies that the group is accessible during 99% of the total time.
- *System availability*: We compute system availability as the average value of the availability ($|\mathcal{A}_G^\beta|$) across all groups in the system. Besides the average we also measure and depict the minimum and maximum group availabilities after convergence.
- *Convergence Time* : This metric represents the convergence speed of our grouping protocols. It is measured by the number of slots required for all the peers to join some group.
- *Normalized message overhead*: It is a measure of the total number of messages exchanged by a protocol divided by the total number of groups in the system after convergence.

5.3 Comparative Study

Fig. 4 depicts the cumulative of percentage of groups as a function of group availability in nines. For non-random strategies, all the groups remain available more than 70% (0.5 nines) of time whereas all of groups formed by random strategy remain available only 55% (0.35 nines) of the time. Comparing Figures 4(a) and 4(b), we can also see that structured method outperforms all other approaches, and it performs even better as uptime is increased.

Expected system availability along with minimum and maximum values (as error bars on each bar) have been presented in Fig. 5. Fig. 5(a) presents system availability for different expected uptime (*i.e.*, L) values and a network of 16000 peers, while Fig. 5(b) presents the same for different network size and $L = 3$ hours. Evidently, structured approach has superior system availability than random and unstructured cases with smaller deviation in the error bars, which is very close to the benchmark performance of the central case. Random approach has the worst result with larger deviation in the error bar and lowest system availability, while the performance of unstructured method lies in between central and random strategies. We capped group size to $\frac{T}{L} + 1$ for this experiment, hence there is not much variation in system availability in Fig. 5(a).

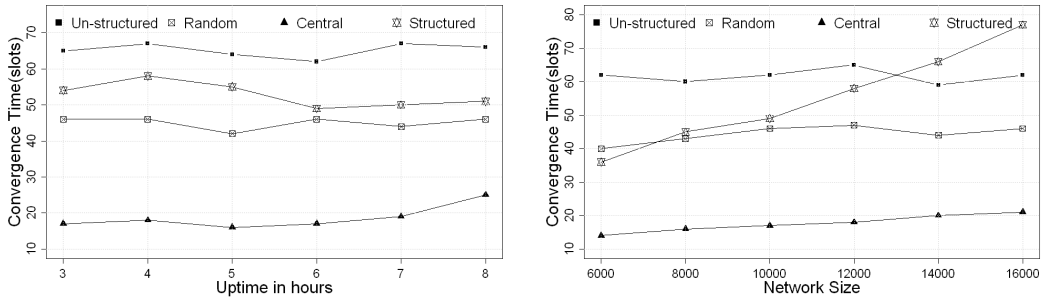


(a) Varying uptime

(b) Varying network size

Figure 5: Systems availability

Fig. 6(a) and Fig. 6(b) present system convergence time as a function of uptime and network size, respectively. Central strategy has the lowest convergence time because the task of index storage and group formation are done in one location in the network. According to Fig. 6(a), changes in uptime has no significant impact on the convergence speed. While Fig. 6(b) shows that the convergence time is not effected by network size for random and unstructured approach, whereas for central and structured approaches convergence time increases with increased network size. For structured case peers with low uptime continuously seeks to form groups with other peers with low uptime, hence the longer convergence time.



(a) Varying uptime

(b) Varying network size

Figure 6: Convergence time

The improvement we gain on availability as shown in the above results comes at the cost of some routing overhead which is illustrated in Fig. 7. However, this overhead is not significantly higher than the unstructured and the random cases. For the central case message overhead is significantly high since each peer communicates the central oracle in each slot for getting grouping updates. The structured routing based technique using Plexus requires communication in a limited search space specified by the Hamming distance and so its normalized overhead lies in the middle. The normalized message overheads for random and unstructured cases are the smallest and very close to each other due to the fact that they require to communicate peer within two-hop neighborhood only.

6 Related Works

A number of approaches to improve availability in P2P systems can be found in the literature. To the best of our knowledge, only a few of these approaches focus on increasing content availability by adopting a time-based replication strategy. In [10], we proposed the *DATA* protocol that construct replication groups using complementary

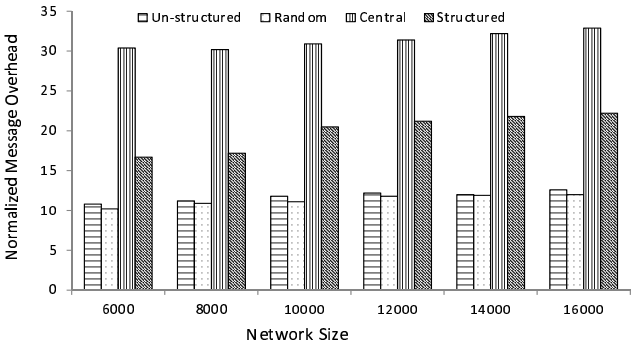


Figure 7: Normalized routing overhead

availability patterns of peers through a gossip based routing technique applicable to unstructured networks. Blond et al. [11] proposed two availability-aware applications that take into account the peers' previous availability history collected through an epidemic protocol. Using a simple predictor, they propose to find the best matching peer to meet the specific goals of the application. A group based Chord model is proposed in [20] to minimize the impact of frequent arrivals and departures of peers. The redundancy group based approach proposed by Schwarz et al. [4] tries to improve availability by utilizing the cyclic behavior of peers distributed across the World. They proposed a hill-climbing strategy to determine redundancy groups for data objects. They proposed a counter-based availability score update mechanism through periodic scans. However, the counter mechanism cannot consistently capture phase relationships within a peer and between peers. For example, a peer having diurnal availability pattern will be online for the longest consecutive stretch starting in the morning, when its counter is the lowest. But this fact is not reflected in their mechanism. Rzadca et al.[9] proposed to represent peer availability as a function of discrete time to minimize the number of replicas. In their model, availability is represented by a set of time slots in which a peer is available with certainty, *i.e.*, the used discrete on-off availability. In contrast, S-DATA represents availability by historical probability at discrete time slots. Our probabilistic model captures diurnal availability patterns more accurately, since peer connectivity cannot be predicted with absolute certainty in a real world network. Moreover, the group formation approach proposed in [9] uses a single-valued scoring function, which only considers the number of newly covered slots while making group formation decision. On the contrary, our utility function considers relative improvement from both sides and considers the size of the resulting group. Finally, their model only targets to ensure 1-availability across time slots, whereas *S-DATA* proposes to consider β-Availability to provide better reliability.

A significant number of research works aim to increase availability by adopting various strategies for quantitative replication. These works vary in the type of redundancy, method of replica regeneration, and the number and location of peers storing redundant data. Bhagwan et al. explored the issues of replication granularity, replica placement, and application characteristics in [1]. In terms of replica generation approach, redundancy is achieved either by replicating the complete data or fragmenting and encoding the data by network coding in such a way that not all fragments are needed to reproduce the original content [1], [4]. Data replication is mainly done in two ways: reactive [2] or proactive [3]. Both of these approaches aim to optimally place the replicas in minimum number of peers so that overall content availability remains high. Existing solutions use information like peers' session time and churn [21], availability history [11], lifespan distribution [14], machines' uptime, downtime, lifetime, and correlation among them [22], Mean Time to Failure [23], up time score [4], recent up time [24], application specific availability [25], availability-prediction guided replica placement [2, 26], and probabilistic models [27] to reduce redundancy overhead while retaining high availability. These approaches relies on quantitative replication, whereas S-DATA combines both time-based and quantitative replication strategies. Another major difference of S-DATA with these schemes is that they make no distinction between transient and permanent disconnections and data stored at a peer is reused upon its return to the system. Ignoring stored data after peers' offline period incurs significant network overhead, which S-DATA can readily avoid by co-relating a returning peer with its previous session.

7 Conclusion

In this paper we have described an efficient grouping scheme which irrespective of peer timing and churn ensures data availability around the clock. *S-DATA* uses a structured solution to create and maintain replication groups.

Experimental results show that the proposed *S-DATA* protocol ensures very high content availability which is comparable to the centralized equivalent for group formation. Moreover, the network and storage overhead incurred by *S-DATA* protocol scales logarithmically with network size. We further intend to investigate performance of *S-DATA* by deploying it on a real world P2P system and to ensure availability for specific application requirements. The success of *S-DATA* depends largely on the willingness and truthfulness of the peers. Tackling the untrusted behavior of peers and security issues of group formation is another prospective research issue we are willing to investigate.

References

[1] R. Bhagwan, D. Moore, S. Savage, and G. Voelker, "Strategies for highly available peer-to-peer storage systems," in *Proc. FuDiCo*, May 2002.

[2] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker, "Total recall: system support for automated availability management," in *Proc. NSDI*, 2004.

[3] E. Sit, A. Haeberlen, F. Dabek, B. Chun, H. Weatherspoon, R. Morris, M. F. Kaashoek, and J. Kubiatowicz, "Proactive replication for data durability," in *Proc. IPTPS*, 2006.

[4] T. Schwarz, Q. Xin, and E. Miller, "Availability in global peer-to-peer storage systems," in *Proc. IPTPS*, 2004.

[5] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proc. IMC*, 2006, pp. 189–202.

[6] J. Chu, K. Labonte, and B. N. LevineH, "Availability and locality measurements of peer-to-peer file systems," in *Proc. ITCom*, 2002.

[7] J. R. Douceur, "Is remote host availability governed by a universal law," *Performance Evaluation Review*, vol. 31, no. 3, pp. 25–29, 2003.

[8] S. Saroiu, P. K. Gummadi, and S. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proc. MMCN*, 2002.

[9] K. Rzadca, A. Datta, and S. Buchegger, "Replica placement in p2p storage: Complexity and game theoretic analyses," in *Proc. DCS*, June 2010, pp. 599–609.

[10] N. Shahriar, M. Sharmin, R. Ahmed, M. Rahman, R. Boutaba, and B. Mathieu, "Diurnal availability for peer-to-peer systems," in *Proc. CCNC*, Las Vegas, Nevada, USA, Jan 2012.

[11] S. Blond, F. Fessant, and E. Merrer, "Finding good partners in availability-aware p2p networks," in *Proc. SSS*, 2009.

[12] R. Ahmed and R. Boutaba, "Plexus: a scalable peer-to-peer protocol enabling efficient subset search," *IEEE/ACM Trans. on Networking (TON)*, vol. 17, no. 1, pp. 130–143, Feb 2009.

[13] Z. Yang, J. Tian, and Y. Dai, "Towards a more accurate availability evaluation in peer-to-peer storage systems," *Intl. Journal of High Performance Computing and Networking*, vol. 6, no. 3/4, pp. 233–246, 2010.

[14] F. E. Bustamante and Y. Qiao, "Friendships that last: Peer lifespan and its role in p2p protocols," in *Proc. Web Content Caching and Distribution*, 2004, pp. 233–246.

[15] Repository of availability traces. [Online]. Available: http://www.cs.uiuc.edu/homes/pbg/availability/

[16] The peer-to-peer trace archive. [Online]. Available: http://p2pta.ewi.tudelft.nl/pmwiki/?n=Main.Home.

[17] Peersim: A peer-to-peer simulator. [Online]. Available: http://peersim.sourceforge.net/

[18] D. Stutzbach and R. Rejaie, "Characterizing unstructured overlay topologies in modern p2p file-sharing systems," in *Internet Measurement Conference*, October 2005.

[19] J. R. Douceur and R. P. Wattenhofer, "Large-scale simulation of replica placement algorithms for a serverless distributed file system," in *Proc. MASCOTS*, 2001.

[20] Y. Dan, C. XinMeng, and C. YunLei, "An improved p2p model basedon chord," in *Proc. 6th PDCAT*, 2005.

[21] R. Mahajan, M. Castro, and A. Rowstron, "Controlling the cost of reliability in peer-to-peer overlays," in *Proc. IPTPS*, 2003.

[22] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer, "Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs," in *Proc. ACM SIGMETRICS*, 2000.

[23] K. Kim, "Time-related replication for p2p storage system," in *Proc. ICN*, April 2008, pp. 351–356.

[24] J. Sacha, J. Dowling, R. Cunningham, and R. Meier, "Discovery of stable peers in a self-organising peer-to-peer gradient topology," in *Proc. IFIP DAIS*, 2006.

[25] S. Shi, G. Yang, J. Yu, Y. Wu, and D. Wang, "Improving availability of p2p storage systems," in *Proc. APPT*, 2003.

[26] J. W. Mickens and B. D. Noble, "Exploiting availability prediction in distributed systems," in *Proc. NSDI*, 2006.

[27] K. Ranganathan, A. Iamnitchi, and I. Foster, "Improving data availability through dynamic model-driven replication in large peer-to-peer communities," in *Proc. GP2PC*, 2002.