Exchanging Peers to Establish P2P Networks

Mursalin Akon, Mohammad Towhidul Islam, Xuemin (Sherman) Shen, and Ajit Singh

Abstract Structure-wise, P2P networks can be divided into two major categories: (1) structured and (2) unstructured. In this chapter, we survey a group of unstructured P2P networks. This group of networks employs a gossip or epidemic protocol to maintain the members of the network and during a gossip, peers exchange a subset of their neighbors with each other. It is reported that this kind of networks are scalable, robust and resilient to severe network failure, at the same time very inexpensive to operate.

1 Introduction

In the Internet world, Peer-to-peer (P2P) computing is an emerging model for service distribution. In contrast to the traditional client-server and push models, the P2P model is characterized by decentralization, self-organization, cooperation among peers and heterogeneity. In P2P model, participant peers work together to reach a common goal. According to this model, an overlay networks is created among peers, and peers bind each other in a logical neighbor relationship. Most often, such an overlay network remains as a pure virtual entity over the physical network.

M. Akon
Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, e-mail: mmakon@uwaterloo.ca

M. Islam
Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, e-mail: mtislam@uwaterloo.ca

X. Shen
Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, e-mail: xshen@bbcr.uwaterloo.ca

A. Singh
Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, e-mail: asingh@uwaterloo.ca

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_6,

Based on the nature of the binding or relationship among peers, P2P networks is often distinguished into two categories: (1) structured and (2) unstructured. The topology of the members in a structured network is ruled by explicit constraints. Contents are distributed among the members using either some hints [4] or the topology of the members [13, 15, 20, 22]. The unstructured networks do not have a predetermined scheme to bind peers in neighbor relationship.

In this chapter, we explore a specific kind of unstructured P2P networks that create the networks by interchanging a subset of neighbors. The heart of these networks is a gossip protocol where a peer talks to a neighboring peer about other neighbors. The key idea is to introduce randomness in the system and eliminate all sorts of global administration. This helps in sustaining dynamics of P2P networks, i.e., constant join and leave of peers, and shows the capacity of self-healing in severe network disasters.

We divide this chapter into several sections. We present four different P2P networks which are created and maintained using the concept of exchanging peers. These networks are PROOFS, CYCLON, IPPS, and Gradient Topology Network, and are elaborated in Sections 2, 3, 4 and 5, respectively. In Section 6, we end our discussion with concluding remarks and future research discussions.

2 The PROOFS Network

Stavrou et al. propose P2P Randomized Overlays to Obviate Flash-crowd Symptoms or PROOFS [19] to manage Internet flash crowd. Internet flash crowd takes place when an object reaches its peak popularity. During the pinnacle popularity of an object, the number of requests may become so tremendous that a significant number of the users are left out and thus the objects become unavailable to them.

2.1 Evolution

Previous solutions to the problem of flash crowd are either administration-wise impractical or expensive. Such solutions are provisioning accessibility based on peak demand, creating dynamic hosts with dynamic domain names, etc. The traditional solution of replicating servers increases availability but involves extensive amount of communication efforts to exchange information and synchronize data with each other, and thus is not scalable. In contrast, the PROOFS network provides a scalable solution that can reliably deliver objects which are extremely popular to be handled by standard delivery techniques.

There exist other structured P2P content distribution systems – such as CAN [13], Chord [20], Past [6, 15], Tapestry [22], Pastry [15], SCRIBE [3, 16], etc. These systems facilitate easy and inexpensive object searches. However, objects or contents with explosive popularity impose the same difficulties of traditional hosts. Moreover, such networks performs poorly in highly dynamic environments where participants or peers join or leave at a very high rate. In contrary, PROOFS networks are robust and possess self-healing property.

2.2 Components

The design of the PROOFS network consists of two components: (1) *client* and (2) *bootstrap server*. The clients[1] are participants of the P2P overlay and they interact with each other to search and retrieve (popular) objects. A bootstrap server caches a finite set of recently joined peers or clients and introduce them to a joining peer. Thus a joining peer becomes familiar with the network.

2.3 Protocols

A PROOFS network is created and maintained with help of two protocols – (1) *ConstructOverlay* and (2) *LocateObject*.

2.3.1 `ConstructOverlay`

The `ConstructOverlay` begins when a peer joins the PROOFS network and obtains initial neighbors from a bootstrap server. During lifetime, a peer maintains at most C number of neighbors. Here, if peer p includes q as its neighbor, p is allowed to be the initiator of a communication involving q. Peer q can communicate with p by only responding to p, unless p is also a neighbor of q. Each peer, in the network, performs a periodic operation called *exchange*.[2] The exchange operation at peer p is described in Algorithm 1.

Algorithm 1: The exchange operation of PROOFS

1 p finds $\mathscr{E}_p \subseteq \mathscr{N}_p$, where each element of $\mathscr{E}_p$ is selected randomly
2 p selects $q \in \mathscr{E}_p$ randomly
3 p sends a request to q with the set $(\mathscr{E}_p \cup \{p\})\backslash\{q\}$ to be a participant
4 **if** *q agrees to the request* **then**
5 $\quad$ q finds $\mathscr{E}_q \subseteq \mathscr{N}_q$, where each element of $\mathscr{E}_q$ is selected randomly
6 $\quad$ q sends a respond back to p with the set $\mathscr{E}_q$
7 $\quad$ p updates $\mathscr{N}_p \leftarrow (\mathscr{N}_p \backslash \mathscr{E}_p) \cup \mathscr{E}_q$
8 $\quad$ q updates $\mathscr{N}_q \leftarrow (\mathscr{N}_q \backslash \mathscr{E}_q) \cup \mathscr{E}_p$

In the algorithm, $\mathscr{N}_p$ is used to designate the neighbor set of peer p. As evident, the sets of peers exchanged and the participant is chosen entirely randomly (line 1, 2,

[1] The term *client* does not refer to the clients in traditional client-server model. In P2P networks, such components are designated as peers. Though the authors used the term client solely, we use client and peer interchangeably to be in symphony with rest of the chapter.

[2] The authors [19] used the term *shuffle* to designate this operation. We use the term *exchange* to identify shuffle in PROOFS and other similar operations in other networks. The exchange operation of PROOFS is used as the basis of other exchange operations discussed in this chapter.

and 5). While updating the neighbor lists (lines 7 and 8) caution should be exercised so that – (1) each neighbor does not exist more than once in a list, (2) a peers is not included as its own neighbor, (3) the number of neighbors are always bounded by C and if not, new members are added until the bound C is reached. Figure 1 shows an example of an exchange operation in PROOFS network. In the figure p initiates the operation with q.

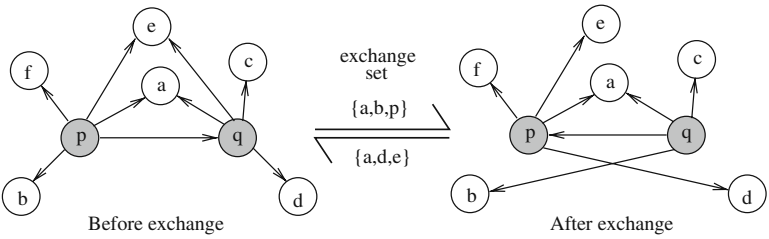


Fig. 1 An example of exchange operation in PROOFS

Note that, a request to participate in an exchange may be denied or even ignored. When the initiating peer does not receive a response back and times out, it assumes that the target peer is no longer maintaining membership with the network, i.e., left the network. A request is declined by an active peer, if and only if another exchange operation is pending. In case of an unsuccessful exchange operation, the initiator waits for a random time amount, picked from a uniform distribution and re-initiates another exchange.

Like other communication networks, in PROOFS, the neighborhood relationship is represented with a graph with directed edges. Peer q being the neighbor of p is indicated by the directed edge from p to q. An exchange operation introduces new edge and may eliminate existing edges, and always reverses the direction of the edge between participants.

2.3.2 `LocateObject`

When a peer wants to retrieve an object in a PROOFS network, it initiates a query using the `LocateObject` protocol. A query includes four vital piece of information – (1) identification of the requested object, (2) time to live (TTL), (3) a fanout value (f), and (4) the address of the query initiator as the return address. A query is allowed to traverse at most TTL number of overlay hops.

When a peer receives a `LocateObject` query, at first it checks the local object repository for the requested object. If available locally, the object is sent to the return address. Otherwise, the query is forwarded to f randomly selected neighbors after the TTL value is decremented by one. Note that, in case the TTL value is decremented to a negative value, instead of forwarding to the neighbors, the query

is discarded. When the query initiating peer does not get a response back (due to timeout), it may re-initiate the same query again, possibly with a higher TTL value.

2.4 Properties of the PROOFS Networks

In graph representation, a PROOFS network is depicted with a directed graph. That is, if q is the neighbor of p, p does not have to be the neighbor of q. The biggest challenge of having an undirected graph appears when a peer leaves the network. Each neighbor of the leaving peer has to find a new neighbor who is also willing to accept an additional neighbor. The ease of not having a bi-directional neighborhood relationship is counteracted by network partitioning.

Let $G_d = (V_d, E_d)$ be the proper directed graph representation of a PROOFS network, where each vertex $p \in V_d$ represents a peer and each directed edge from vertex p to vertex q, i.e., $(\overrightarrow{p,q}) \in E_d$ indicates that q is a neighbor of p. Let $G_u = (V_u, E_u)$ be the undirected version of G_d, i.e., V_u and V_d are the same and for each $(\overrightarrow{p,q}) \in E_d$ there exists $(\overrightarrow{p,q}) \in E_u$ and $(\overrightarrow{q,p}) \in E_u$ or simply $(p,q) \in E_u$.

Property 1: Given that, G_u is the undirected representation of a PROOFS network and is connected. If an exchange operation drives the graph representation to G'_u from G_u, G'_u is also connected.[3] In other words, no exchange operation partitions a connected PROOFS network.

Property 2: Let G_d and G_u be the directed and undirected graph representation of a PROOFS network. Let there exists a path from peer p to q in G_u but not in G_d. There exists a series of exchange operations that introduces a path from p to q in G_d. To have the series of operations, consider a path, consisting of the sequence of vertices $< p, (p+1), (p+2), \ldots, (q-2), (q-1), q >$, from p to q in G_u. Considering the same sequence of vertices (and related edges) in the G_d graph, there will be some edges those point towards p and others towards q. Let $(\overrightarrow{u,v})$ be an edge on that sequence of edges pointing towards p. An exchange initiated by u with participant v would make the edge pointing towards q. To have a path from p to q, the direction of all those edges pointing towards p has to be reversed, and any combination of related exchange operations serves the purpose.

2.5 Results

Some of the important results about PROOFS are presented in this section.

[3] For details proof, readers may refer to the original paper.

2.5.1 Connectivity

The authors investigate the effect of dynamic join and leave on the connectivity of the network. In simulations, for each peer p in the PROOFS network, the fraction of other peers in the network reachable from p using the directed path is computed. It was found that at least 95% of the time the average reachability is one, i.e., each peer in the network can reach all other peers. The reported lowest reachability, considering all the peers, is 20%. However, such a poor connectivity occurs in extreme situations such as when the expected time a peer remains in the network is 50 times smaller than the expected time a peer exists the network. In practice, this kind of extreme situation is hardly found.

2.5.2 Noncooperative Peers

The peers in PROOFS are simply applications running on user computers. As a result, it is extremely difficult, if not impossible, to make all the peers fully cooperative. Besides cooperative peers, there may exist peers with different levels of cooperations – (1) a *query-only* peer simply forward queries irrespective of availability of the requested object in the local cache, (2) a *tunneling* peer is same as a query-only peer but considers fanout to be 1, and (3) a *mute* peer drops all the queries from any other peers in the network.

It has been found that if the number of query-only or tunneling peers grows up to 80%, almost 100% queries turn out to be successful in finding the target objects. The worst query success rate is observed with mute peers. When the population of mute peer reaches as high as 80% the query success rate drops but stays above 80%.

3 The CYCLON Network

Spyros et al. propose the CYCLON network [21] as a gossip-based network membership management protocol in unstructured P2P networks. The goal of the research is to design a management protocol that results in a network having low diameter, low clustering, highly symmetric node degree and at the same time is highly resilient to massive node failures.

3.1 An Enhanced Exchange Operation

To achieve the goal, the authors propose an enhanced peer exchange[4] operation. The enhanced operation uses the similar working steps of the basic exchange operation

[4] Spyros et al. use the term enhanced shuffle to designate their peer exchange mechanism.

discussed in the previous section. The critical difference is that unlike the basic one, in enhanced exchange, an initiating peer does not choose the participant randomly.

To facilitate the enhancement, the exchange operation is performed periodically with an interval of Δt. Each peer not only maintains a list of its neighbors but also *age* for each of the logical outgoing links (or edges to neighbors). The age of an edge gives an approximate estimation of time, in Δt unit, since the edge is created by the peer the edge points to. Algorithm 2 shows the steps of the enhanced exchange operation, initiated by peer p with participant q.

Algorithm 2: The enhanced exchange operation of CYCLON

1 p increases the age of all outgoing edges pointing to the neighbors
2 Let $q \in \mathcal{N}_p$ be a peer, such that $t_{(\overrightarrow{p,q})} \geqslant t_{(\overrightarrow{p,r})}$, where $r \in \mathcal{N}_p \wedge q \neq r$
3 Let $\mathcal{E}_p \subseteq \mathcal{N}_p$, where each element of $\mathcal{E}_p$ is selected randomly and $|\mathcal{E}_p| = l - 1$
4 p sends a request to q with the set $\mathcal{E}_p \cup \{p\}$ to be a participant
5 **if** *q agrees to the request* **then**
6 $\quad q$ finds $\mathcal{E}_q \subseteq \mathcal{N}_q$, where each element of $\mathcal{E}_q$ is selected randomly and $|\mathcal{E}_q| = l$
7 $\quad q$ sends a respond back to p with the set $\mathcal{E}_q$
8 $\quad p$ updates $\mathcal{N}_p \leftarrow (\mathcal{N}_p \backslash \mathcal{E}_p) \cup \mathcal{E}_q$
9 $\quad q$ updates $\mathcal{N}_q \leftarrow (\mathcal{N}_q \backslash \mathcal{E}_q) \cup \mathcal{E}_p$

In the algorithm, p picks up the peer that is pointed by the oldest edge (line 2, where $t_{(\overrightarrow{p,q})}$ is the age of the edge $(\overrightarrow{p,q})$). The number of peers exchanged is called *exchange length* and determined by the system parameter $C \geqslant l > 0$. When q updates its neighbor list, a new edge pointing towards p is to be created with an age of 0. All other edges, including those, which are sent over during the exchange, continue to maintain their respective previous ages. So, while exchanging peers, not only a list of peers are sent out but also their respective ages. As the basic steps of both basic and enhanced exchange operation are the same, the properties described in Section 2.4 also hold for enhanced exchange operation.

3.2 Results

Spyros et al. reports some very interesting property of the CYCLON network. In this section, we discuss some of their findings.

3.2.1 Average Path Length

To compute the average path length, the undirected version of the graph representation of a network is considered. The undirected graph conveys the idea of peers being *informed* of the neighbors in the undirected sense, i.e., if q is a neighbor of p, at some point in the future p will become a neighbor of q. The concept is brought

forward due to the second property of exchange networks described in Section 2.4. It is observed that CYCLON network can converge to the average path length of a random network within a hundred cycles, where a cycle is defined by the maximum time duration allowing all peers to engage in a single exchange operation or Δt. It is also observed that the average path length increases logarithmically with the number of peers in the network. These observations indicate robustness of CYCLON in applications where the entire network has to be reached out.

3.2.2 Average Clustering Coefficient

The clustering coefficient is defined as the ratio of the number of existing edges between neighbors of a peer and the total number of possible edges between them. An average over this coefficient for all peers gives an idea of how many peers are neighbors of their own neighbors. The authors demonstrate that average clustering coefficient of a CYCLON network converges to that's of a random network[5] within a few hundred cycles.

3.2.3 Degree Distribution

Degree of a peer is a very important performance metric in unstructured networks. *Degree* of a peer is defined as the number of edges from the peer in the graph representation. The degree related to the number of outgoing edges is defined as out-degree and the number of in coming edges is in-degree. The out-degree of each peer in the CYCLON network is fixed and is always C. In-degree is the factor we are most interested in.

The distribution of in-degree reveals some very significant characteristics of the network as described below:

- Existence of a peers with significantly low in-degree many result in partitioned or disconnected network, in case the peers referring to the concerned peer die or leave the network. Similarly, a peer with a very large in-degree also represents a weak point, as failure of the peer may result in a disconnected network.
- The distribution of in-degree represents how search or other epidemic protocols behave. For example, a massively connected peer may receive same query from many other peers several times which not only waste resources but also give poor performance for the protocols.
- The distribution of in-degree also bears the indication of how loads are distributed among peers. For example, a massively connected peer has to provide responses from many other peers where as a weakly connected peer may simply be idle.

Figure 2 shows an example distribution of in-degree in a basic exchange (such as PROOFS) and enhanced exchange (such as CYCLON) network. As can be seen

[5] Average clustering coefficient of a random network is effectively defined as $\frac{2\times C}{N-1}$, where N is the total number of peers in the network.

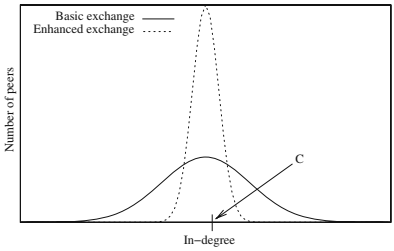


Fig. 2 In-degree distribution

in the figure, most of peers have in-degrees which are very close to the out-degree (i.e., C). This illustrates that the most of the peers have similar load. It also indicates that the load per peer is fair. The amount of services, a peer is expected to deliver is equivalent to the amount of service a peer is expected to receive. Figure 2 can be explained by investigating the introduction, deletion and lifetime of an edge. As shown in Fig. 1, at the end of the exchange, the edge from p to q is deleted and a new edge from q to p is created. The enhanced exchange mandates that the new edge is assigned an age 0 and the deleted edge is the oldest edge towards a neighbor.

At each cycle, a node engages in one exchange operation, and thus creates a new edge and abolishes an old one. Say, at cycle t, a new edge, with age 0, is created towards peer p. At cycle $(t+1)$, another new edge towards p will be created and the age of the last created edge will be incremented to 1. As a peer maintains C number of neighbors and at each cycle the ages of the edges towards neighbors are incremented by 1, a peer can host edges aged from 0 to $(C-1)$ only. So, a newly created edge will be deleted within C cycles and a peer can have an edge as old as C cycles pointing towards itself. In other words, a peer can have C edges pointing towards itself at a single point of time.

In CYCLON, exchange operations between all the peers are periodic but are not synchronized. Thus, the number of edges pointing towards each peer may vary a little around C. Departure of peers and non-responding peers may allow an edge to stay little longer or delete an edge prematurely. However, the system can recover within C cycles. Unlike the enhanced version, basic exchange does not impose any condition on the length of lifetime of an edge. As a result, at one extreme, an edge may stay in the system for indefinite time and at the other extreme, an edge may be deleted in the next cycle of its creation. That's why, the basic exchange operation results in a in-degree distribution with much higher variance as compared with that's of the enhanced version.

4 The IPPS Network

Inexpensive Peer-to-Peer Subsystem (IPPS) is an unstructured platform, proposed for wireless mobile peer-to-peer networks by Akon et al. [1, 2]. The platform

addresses the constraints of expensive bandwidth of wireless medium, and limited memory and computing power of mobile devices. It uses a computationally-, memory requirement- and communication- wise inexpensive protocol as the main maintenance operation, and exploits location information of the wireless nodes to minimize the number of link-level messages for communication between peers.

4.1 The Problem and the Goal

A wireless mobile network is a cooperative network where each node requires to collaborate with each other to forward packets from a source to a destination. In such a network, the entire available channel capacity may not be available to an wireless application, and the actual throughput is also determined by the forwarding load generated by other wireless nodes. Besides, mobile devices are battery operated. Unlike electronics, advances in battery technology still lag behind. Minimizing the number of link-level wireless hops helps in increasing the capacity available to the applications. Reduced number of link-level hops also means less number of transmission and less power consumption for a mobile node. Along with being thrifty about bandwidth consumption, a suitable application for mobile devices is required be computationally inexpensive to ensure prolonged battery life and memory requirement-wise economical to confirm accommodation in the small system memory.

In spite of the limitations of wireless mobile networks, P2P over high capacity cellular networks and wireless LANs can provide a wide range of services such as sharing files [9]. In scenarios where accessing a commercial network is expensive, members of a P2P network can share downloaded objects with each other or even can collaborate to download a large popular object. This not only provides a cheaper way of sharing resources, but also enables low latency access to remote objects. Dissemination of rescue or strategic information in a disaster or war zone can be accomplished using mobile wireless P2P network. Short message broadcast, multimedia broadcast, text, audio and / or video based conference are some other examples.

There are some proposals to use existing or modified structured networks in wireless and mobile networks. For example, XScribe [11] is modified from SCRIBE [3] to suite in mobile networks. However, a structured P2P network faces a high network maintenance cost and the ability of this type networks to work in extremely unreliable environments has not yet been investigated. In contrary, an unstructured P2P network is a low cost network which can sustain any extreme environment [19]. Although such a benefit is achieved at the expense of higher search cost, the network assumptions and the overall gain have made this kind of P2P networks so attractive

that several unstructured P2P networks have been deployed and are being used by a huge user communities.

In a wired network, due to the abundance of resources, performance metrics of many applications are abstract. However, P2P networks in wireless mobile environment should be very economic about the resources of the wireless medium and devices. The goal of IPPS is to provide an inexpensive and well performing P2P platform on which different P2P applications can be developed. To achieve the goal, an unstructured P2P network, exploiting location information, is examined. While designing the platform, careful choices are made to make the platform flexible, robust and fault tolerant.

4.2 System Model

IPPS system model consists of a set of collaborative computing nodes, each equipped with a wireless interface. Those nodes are assumed to have the capability to form a network on-the-fly using an ad-hoc networking technology, such as GeRaf [23, 24], an efficient location aware transmission (MAC) and forwarding (routing) scheme, to manage the network. In this model, for each node, participation in the P2P network is optional. However, irrespective of its membership in the P2P network, each node participates in routing messages from one node to another as a low level service. The network is equipped with low level (lower than application level) point-to-point unicast primitives, and each of the mobile devices has access to some form of location service [5, 12]. Through this location service, a node in the network can obtain the physical location of itself or other nodes. The information from the location service is used by the lower level network management (i.e., GeRaf) as well as by the P2P modules (i.e., IPPS library). Figure 3 shows an example of the considered network.

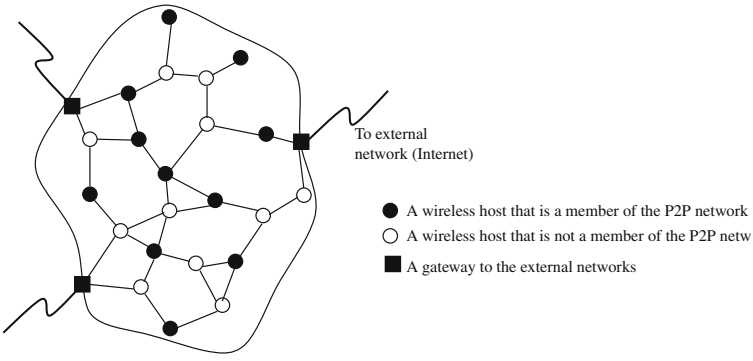


Fig. 3 An IPPS network

4.3 Topology Maintenance

In this section, we discuss some of the important components and properties of an IPPS network.

4.3.1 The Exchange Operation

IPPS borrows the concept of exchanging neighbors[6] from *PROOFS* [19]. However, the goals of the operation in these two networks are exclusive. In PROOFS network, the operation provides randomness, where as, in IPPS, the operation makes attempts to being neighboring peers closer to each other. The authors make the following claim about their operation.

Claim: It is expected that exchange operations reduce link level hop count between neighboring peers.

Similar to CYCLON, each peer in IPPS performs exchanges at a regular interval. During a exchange, l neighbors are interchanged between the initiator and the participant. Peer p chooses the participating peer q among its own neighbors with the intention of reducing the total distance between the peers. Distance between two peers convey the idea of physical distance between them. Considering the system model and the underlying network infrastructure, the hop count between two neighboring peers is proportional to the distance between them.

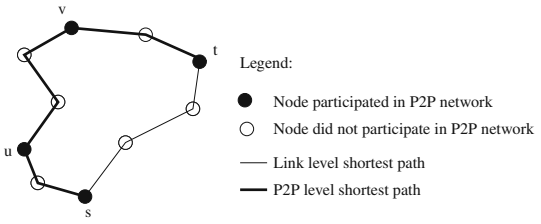


Fig. 4 Shortest path in P2P and link level network

Claim: Exchange reduces the bandwidth requirement to forward P2P messages.

A peer usually forwards P2P messages, such as query messages, to its P2P neighbors only. As not all communication nodes participate in the P2P network, a P2P level hop may consist of several link level hops. Figure 4 shows the idea pictorially. There exists one non-P2P node between s and u (i.e., two hops), whereas there are two non-P2P nodes between u and v (i.e., three hops). In a random P2P network, on an average one P2P hop consists of average link level path length of the network. In the worst case, where two neighboring peers are located at the extreme ends, a single P2P hop has a link level hop count which is equivalent to the network diameter. Having a neighbor located at a nearby location results in reduction in number

[6] The authors designate their version of neighbor exchange protocol as *reformation*.

of hops between the peers. This helps in reducing of number of link level messages which helps in reducing the total bandwidth consumption to forward P2P messages. Moreover, fewer hops mean reduced message latency. Note that both of these properties are very much desirable for wireless mobile applications, as reduced number of link level messages slows down energy consumption and boosts battery life of mobile devices.

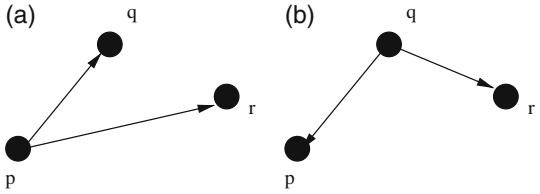


Fig. 5 An exchange operation in IPPS

To have peers located at a close geographic area, the concept of *distance gain* is introduced in IPPS. During an exchange between peers p and q, if the initiating peer p forwards another P2P neighbor r to q, the distance gain is the reduction of the distances between the pairs p and r and the second pair q and r. Figure 5 shows a exchange where a directed edge from any peer x to another peer y means that y is a neighbor of x. Now, the distance gain is formally given by:

$$d^p_{q,r} = |dist(p,r)| - |dist(q,r)| \tag{1}$$

where $dist(x,y)$ is the distance between x and y. When a peer p wants to engage in an exchange, it finds the peer which results in the maximum distance gain. To compute such a metric, for each $q \in \mathcal{N}_p$, p performs the following computations.

1. It computes a *preliminary reform-set* $\mathcal{NR}^q_p$ such that $|\mathcal{NR}^q_p| = l - 1$ and $\mathcal{NR}^q_p \subset \mathcal{N}_p - \{q\}$. The preliminary reform-set must satisfy the following condition:

$$d^p_{q,u} \geqslant d^p_{q,v} \tag{2}$$

   where $u \in \mathcal{NR}^q_p$ and $v \in \mathcal{N}_p - \mathcal{NR}^q_p - \{q\}$. In other words, $\mathcal{NR}^q_p$ includes $l-1$ number of the most distance gain contributing neighbors of p, during a potential exchange with q;
2. it then computes the net gain for the preliminary reform-set as:

$$d^p_q = \sum_{r \in \mathcal{NR}^q_p} d^p_{q,r} \tag{3}$$

Finally, p chooses $t \in \mathcal{N}_p$ as the participator of the operation where $d_t^p = \max_{q \in \mathcal{N}_p}\{d_q^p\}$. During the operation, p sends over a *REFORM_REQUEST* message to t accompanied with the reform-set $\mathcal{NR}_p^t \cup \{p\}$. When peer t receives the exchange request from p, it computes the reform-set for p and then sends the set back to p as a *REFORM_RESPONSE* message. Unlike the reform-set from p, the set, computed by t, consists of a list of l peers from $\mathcal{N}_t$ which maximizes the net distance gain for p. After a successful exchange operation, both p and t perform a merge operation as discussed in next. Detailed control flows of an initiator and a participator are given in Algorithm 3 and 4.

4.3.2 The Merge Operation

Peer p performs a merge operation after it gets back the reform-set from t. In contrary, t performs the operation after it decides about the reform-set to send out. Without lose of generality, let p be a peer performing a merge operation. $\mathcal{N}_{send}$ and $\mathcal{N}_{recv}$ be the reform-sets that are sent and received, respectively. During the merge operation peer p updates its neighbor set $\mathcal{N}_p'$ as follows:

$$\mathcal{N}_p' \leftarrow (\mathcal{N}_p \backslash \mathcal{N}_{send}) \cup \mathcal{N}_{recv} \tag{4}$$

where $\mathcal{N}_p'$ is the new P2P neighbor set of p. Note that it is certainly possible that $(\mathcal{N}_p \backslash \mathcal{N}_{send}) \cap \mathcal{N}_{recv} \neq \emptyset$. In such cases, $|\mathcal{N}_p'| < |\mathcal{N}_p|$. Measures should be taken to carefully handle such cases. This issue is further elaborated next.

4.3.3 Number of P2P Neighbors

In IPPS platform, an upper and a lower bounds is set on the size of the P2P neighbor set, a peer can have. Those bounds are defined as N_{max} and N_{min}, respectively and must satisfy the following condition.

$$N_{max} \geqslant N_{min} > l \tag{5}$$

There are some situations when the neighbor set size grows beyond the N_{max} threshold (for example, when a joining peer gathers peers from several known peers for its initial neighbor set). In those cases, the peer will keep N_{max} number of the nearest peers and discard the rest. Similarly, there are some scenarios where a neighbor list shrinks below the N_{min} threshold (for example, when a neighboring peer fails to respond to a P2P control message). Therefore, the peer requests for a neighbor list either from one of the available neighbors or from some widely known repository, following the same procedure of a joining peer.

The upper bound N_{max} puts a limit on the worst case computational and space complexity for a peer. The lower bound N_{min} provides robustness to IPPS. By tuning those parameters, the connectivity of the network can be controlled. The gap between N_{max} and N_{min}, i.e., $(N_{max} - N_{min})$, allows the platform different levels of fault tolerance. The larger the gap, the more a peer tolerates reduction of the size of the neighbor set, i.e., failure of neighbors. PROOFS can be mapped into a special scenario where N_{max} and N_{min} are equal. However, this makes PROOFS unfavorable for wireless mobile networks which suffer from temporal disconnections or for P2P networks which allow dynamic join and leave of participating peers. The reason is that to maintain a specific number of neighbors, PROOFS suffers from a huge number of initialization operation at detecting of each unavailable neighbor.

Algorithm 3: Control flow of an exchange initiating peer p

```
1  while true do
2      Compute the participating peer
3      Let, t be the participating peer
4      Let, N_send be the reform-set
5      Send a REFORM_REQUEST to t with the reform-set N_send
6      if t responds before timeout then
7          Let, N_send be the received reform-set in REFORM_RESPONSE
8          N_p ← (N_p \ N_send) ∪ N_recv
9          if |N_p| < N_min then
10             call AddNeighbor()
11         else
12             Shrink N_p to size min(|N_p|, N_max)
13         end
14         break
15     else
16         N_p ← N_p − {t}
17         if |N_p| < N_min then
18             call AddNeighbor()
19         end
20     end
21 end
```

Algorithm 4: Control flow of an exchange participating peer q

```
1 Let, N_recv be the reform-set received from p
2 Compute the reform-set N_send to send to p
3 Send back a REFORM_RESPONSE to p with reform-set N_send
4 N_q ← (N_q − N_send) ∪ N_recv
5 if |N_q| < N_min then
6     call AddNeighbor()
7 end
```

Procedure `AddNeighbor`

Let r be the executing peer
repeat
    Send a SHARE_REQUEST to a known repository
    Let, $\mathcal{N}_{recv}$ be the set received in SHARE_RESPONSE
    $\mathcal{N}_r \leftarrow \mathcal{N}_r \cup \mathcal{N}_{recv}$
until $|\mathcal{N}_r| < N_{min}$
Shrink $\mathcal{N}_r$ to size $\min(|\mathcal{N}_r|, N_{max})$

4.4 Results

An event driven simulation tool is developed to evaluate the performance of IPPS. In the simulations, a rectangular area of size 175×175 *square units*, where 5000 mobile nodes are randomly distributed according to a Poisson process, is considered. Different status from the network were collected at a fixed interval. The authors compare IPPS with PROOFS wherever possible.

4.4.1 Computational and Memory Complexity

The computational complexity of exchange in IPPS is the complexity faced by the initiating peer. This is due to the fact that the initiating peer incurs more computational complexity than the responding or participating peer. The following is an analysis of the complexity with simple data structures and straight forward algorithms:

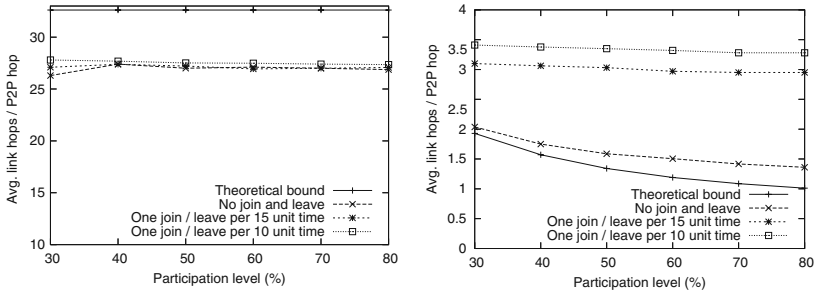
1. The complexity to find the net distance gain for a specific neighbor is $\Theta(|\mathcal{N}| + (l-1)) = \Theta(|\mathcal{N}| + l) = \Theta(|\mathcal{N}|)$;
2. For all neighbors, the complexity turns out to be $\Theta(|\mathcal{N}|^2)$;
3. By tracking properly during the previous computations, the neighbor with maximum net gain can be found in $\Theta(1)$ time.

Therefore, the total complexity becomes $\Theta(|\mathcal{N}|^2)$. The worst case scenario arises when $|\mathcal{N}| = N_{max}$ and then the computational complexity becomes $\Theta(N_{max}^2)$. A peer faces the worst case memory requirement when the neighbor list grows beyond N_{max} and this requirement can be formally expressed as $\Theta(N_{max} + l)$. For a given network, N_{max} and l are constants and small positive integers.

4.4.2 Number of Link Level Hops per P2P hop

Figure 6a, b show the average number of link level hops per one P2P hop using the PROOFS and IPPS, respectively. The figures also show the theoretical upper bound

on the average number of link level hops, considering that each node has global view of the entire network and no existing node either leaves the P2P network nor a new node join in. In case of PROOFS, due to the randomness of the network, the theoretical upper bound is fairly followed. On the other hand, in case of IPPS, a node does not have the global view and it may not choose the optimal neighbors with the lowest distance between them. As can be seen in Fig. 6b, IPPS performs slightly poorer than the optimal upper bound. As the percentage of mobile nodes participated in the P2P network increases, the number of link level hops per one P2P hop decreases. In fact, as the participation level increases, the chance to find a P2P neighbor at a nearer location also increases. However, if a network uses the PROOFS system (which is random in nature), this metric remains approximately the same, irrespective of different levels of participation. In this case, as the neighbors of a peer are uniformly distributed all over the network, the average link level hop count is not affected at all by the participation level. Actually, the simulation results presented in [21] show that only in an ideal situation (which is a perfect random system with no network dynamics), PROOFS or similar systems can achieve the best performance where the average length of a single P2P hop is equivalent to the average path length of the whole network. Comparing Fig. 6a, b, link level hops per P2P level hop is significantly lower in our proposed platform. This indicates that IPPS reduces the bandwidth requirement and energy consumption to transmit P2P messages.



(a) Basic exchange (PROOFS) (b) Exchange in IPPS

Fig. 6 Number of link level hops per P2P hop

4.4.3 Dual Cognizance

The exchange operation in IPPS establishes neighborhood relation among geographically close peers. At each exchange, a peer modifies the neighbor set with the peers that are closer than those of the previous set and the neighbors of that peer do the same. So, if p finds q to be at a closer location, it is likely that q also finds

p the same and includes each other in their neighbor set. The property of a peer being the neighbor of its own peer is defined as *dual cognizance*, by the authors. Figure 7 shows the percentage of peers satisfying the dual cognizance property in PROOFS and IPPS. Note that, in a perfect random PROOFS network, the best case dual cognizance can be analytically defined as $(\frac{N_{\max}}{N-1})^2$, where N is the total number of peers. On the other hand, for an optimal IPPS (where each peer has a global view of the entire network), this metric is 1.

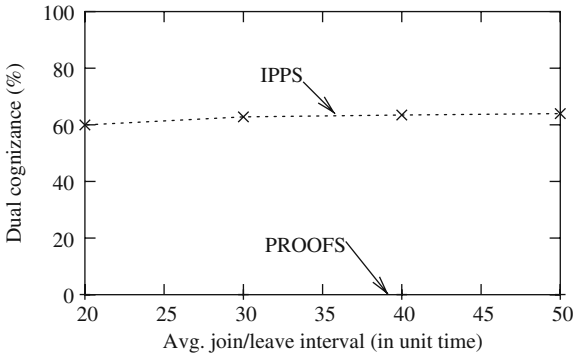


Fig. 7 Effects of join/leave interval on dual cognizance

4.4.4 Minimum Connectivity

An important property of an exchange network is – given a connected network, no exchange operation can make the network disconnected (see Section 2). However, it is possible that the P2P network becomes disconnected as peers join and leave the P2P network. Mobility may further deteriorate the scenario, when the underlying network becomes physically disconnected as mobile nodes are unreachable from one another using radio links. During the simulation, authors compute the connectivity of the P2P network. If p is a neighbor of q, q is considered to *know* p and vice versa, and are connected in both way. The simulation results fairly support the previous claims [19] that for almost all the cases more than 95% of the peers remain connected, given that they are also connected in their radio network. The worst case scenario, i.e., the minimum connectivity in the P2P network, was also investigated. Figure 8 shows the minimum connectivity of the network for different join/leave intervals. The numbers of peers in the largest connected peer graphs are computed and presented after normalizing in 1. As expected, the minimum connectivity decreases with decrement of participation level as well as with the frequency of joining/leaving the P2P network. It can be seen that the worst case connectivity is higher than 70% which provides an indication of robustness of IPPS platform.

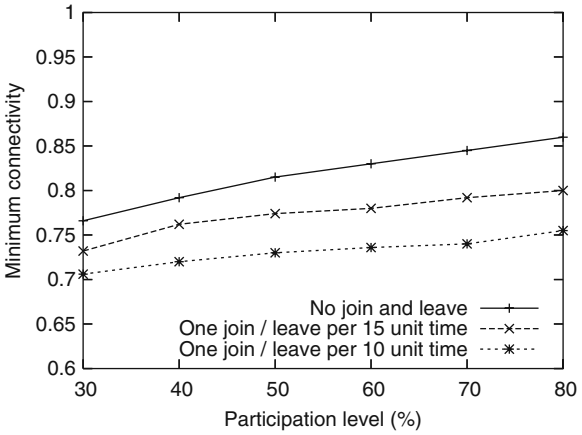


Fig. 8 Minimum connectivity among peers for different join / leave intervals

5 The Gradient Topology Network

In gradient topology, the highest priority entities are connected with each other. These connected entities are called the *core*. Lower priority entities are arranged gradually further away from the core. The position of an entity indicates its priority in the system. In this section, we discuss a gradient topology network, proposed by Jan et al., to facilitate ease of finding resourceful peers in a P2P network [17].

5.1 The Preliminaries

It is observed that distribution of peers in terms of resources is highly skewed [18] and peers with poor resource conditions can result in inferior network performance [14]. These observations lead to the concept of *super peers*. Compared to average peers in the network, a super peer is highly resourceful. To improve performance of the system, in many applications, critical and important services are assigned to these high capacity super peers.

OceanStore [7] architecture exploits a primary tier of super peers with high capacity (in terms of high bandwidth and connectivity) to preserve replicas of objects and employs them to manage updates. In Chord [20], multiple virtual servers are assigned to high performance hosts, i.e., super peers. Such peers are utilized to enhance the routing performance in distributed hash tables (DHT) [10].

The proposal of gradient topology network addresses two issues. Firstly, election of super peers – due to enormous size of P2P networks and their dynamics, it is very difficult for a single entity to maintain a global view of the entire network. So, a distributed solution is desirable. Secondly, finding super peers – it is important that

super peers of interest are searchable so that other ordinary peers can easily obtain important services from these super peers. The former problem is out of the scope of this chapter and we concentrate on the later problem, which is solved using an exchange network.

5.2 Exchange Operation

In the gradient topology network, each peer maintains two sets of neighbors. At peer p, the first set, $\mathscr{S}_p$, contains a set of peers with similar capacity (or priority) and like PROOFS, the second set, $\mathscr{N}_p$, maintains a set of random peers. For each neighbor q in both $\mathscr{S}_p$ and $\mathscr{N}_p$, peer p also maintains the capacity (U). The random neighbor set is used to discover unexplored peers in the network for similar capacity. This way, the chance of having more than one cluster of similar capacity peers is reduced. The random network also provides robustness and makes the network resilient to network partitioning. Besides, the random neighbors facilitate the distributed computation of capacity of peers and election of super peers. In this network, peer p performs periodic exchange operation, as shown in Algorithm 6.

Algorithm 6: The exchange operation of Gradient Topology Network

```
1 Let q be a randomly selected peer from S_p ∪ N_p
2 p sends a request to q with the two sets S_p and N_p to be a participant
3 if q agrees to the request then
4     q sends a respond back to p with the two sets S_q and N_q
5     call GTNReplacePeer (p, S_q, N_q) from p
6     call GTNReplacePeer (q, S_p, N_p) from q
```

Procedure `GTNReplacePeer`

input: x, $\mathscr{S}_{recv}$, $\mathscr{N}_{recv}$

```
1 Let p ∈ S_recv such that |U(p) − U(x)| is the minimum
2 Choose r ∈ S_x randomly
3 Update S_x ← (S_x ∪ {p})\{r}
4 Choose p ∈ N_recv randomly
5 Choose r ∈ N_x randomly
6 Update N_x ← (N_x ∪ {p})\{r}
```

In the `GTNReplacePeer` procedure, the calling peer replaces one entry in the similarity-based set with another peer, received during the exchange operation. The new peer is chosen such that the capacity is similar to the calling peer (lines 1–3).

Later on, one entry in the random neighbor set is replaced with another entry from received random neighbor set (lines 4–6).

5.3 Search

The organization of peers in a gradient topology enables an efficient heuristic search technique to find the high capacity or super peers in the network. The search algorithm uses the capacity information embedded in the topology to restrict the procedure within a small number of peers. When a peer initiates a search, a desired capacity threshold is included within the query message. The threshold is determined based on the resources requirement of the target operation. A peer p with capacity lower than the threshold greedily forwards the query to the neighbor q with highest capacity. Formally, $q \in \mathscr{S}_p \cup \mathscr{N}_p$ and $U(q) \geqslant U(r)$, where $r \in \mathscr{S}_p \cup \mathscr{N}_p \wedge r \neq q$. The forwarding process continues until a peer with required capacity is found or time-to-live (TTL) value of the query expires. Note that, due to peer churn, a search may result in looping in a local minima. To prevent such looping, all visited peers are added the query message and a message is never forwarded to a peer that the message has already visited.

5.4 Results

A P2P network, consisting of up to 100000 peers, is simulated to evaluate performance of the proposed scheme. The capacity of the peers are assigned such that only 1% of them are considered to be super peers. The network is put under constant churn. It is observed that the evolved gradient topology have very small diameter and the average hops to find super peers is bounded by the diameter. In a network as large as to include 100000 peers, the diameter is typically in the order of 5 or 6. As a result, it takes significantly fewer steps to find super peers in the gradient topology as compared to other techniques, such as random walk [8].

6 Concluding Remarks

In this chapter, we have investigated four unstructured P2P networks which are created and maintained using the concept of exchanging peers. These networks demonstrate that simple exchange operation can harness a handful extra ordinary features. They also signify the variety of usages of the exchange operation. The PROOFS network is a robust and scalable network to handle Internet flash crowd that traditional technologies fail to manage. The CYCLON network is introduced to enable a P2P network to be load balanced. IPPS is an unstructured platform for wireless

mobile P2P networks. The platform addresses the limitations of wireless medium and mobile devices, such as, expensive bandwidth of wireless medium, and limited memory and computing power of mobile devices. IPPS is a computationally-, memory requirement- and communication- wise inexpensive protocol that is excellently suited for the target environment. Unstructured P2P networks are typically considered to sacrifice search performance for inexpensive maintenance operations. In contrast, the gradient topology network utilizes the exchanging peer mechanism to facilitate a superior search technique.

Little research on efficient searching in unstructured P2P network has been done. Working principles of networks like CYCLON and IPPS reveal interesting and fundamental properties which typical unstructured P2P network do not have. As a result, search techniques exploiting these features are yet to be explored.

References

1. Akon, M., Shen, X., Naik, S., Singh, A., Zhang, Q.: An inexpensive unstructured platform for wireless mobile peer-to-peer networks. Peer-to-Peer Networking and Applications **1**(1), 75–90 (2008)
2. Akon, M.M., Naik, S., Singh, A., Shen, X.: A cross-layered peer-to-peer architecture for wireless mobile networks. In: IEEE International Conference on Multimedia & Expo, pp. 813–816. Toronto, Canada (2006)
3. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.: SCRIBE: A large-scale and decentralised application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications (JSAC) **20**(8), 100–110 (2002)
4. Clarke, I., Miller, S., Hong, T., Sandberg, O., Wiley, B.: Protecting free expression online with freenet. IEEE Internet Computing **6**(1), 40–49 (2002)
5. Cleary, D.C., Parke, D.C.: "Finding Yourself" building location services in a peer-to-peer wireless world. In: EUROCON 2005, pp. 60–63 (2005)
6. Druschel, P., Rowstron, A.: PAST: A large-scale, persistent peer-to-peer storage utility. In: HotOS VIII, pp. 75–80. Germany (2001)
7. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: Oceanstore: An architecture for global-scale persistent storage. ACM SIGPLAN Notices **35**(11), 190–201 (2000)
8. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: ICS '02: Proceedings of the 16th international conference on Supercomputing, pp. 84–95. NY, USA (2002)
9. Mavromoustakis, C.X., Karatza, H.D.: Segmented file sharing with recursive epidemic placement policy for reliability in mobile peer-to-peer devices. In: IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 371–378 (2005)
10. Mizrak, A.T., Cheng, Y., Kumar, V., Savage, S.: Structured superpeers: Leveraging heterogeneity to provide constant-time lookup. In: IEEE Workshop on Internet Applications, pp. 104–111 (2003)
11. Passarella, A., Delmastro, F., Conti, M.: XScribe: a stateless, cross-layer approach to P2P multicast in multi-hop ad hoc networks. In: International Conference on Mobile Computing and Networking, pp. 6–11. Los Angeles, California (2006)
12. Pias, M., Crowcroft, J., Wilbur, S., Harris, T., Bhatti, S.: Lighthouses for scalable distributed location. In: Second International Workshop on Peer-to-Peer Systems, pp. 278–291 (2003)

13. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content addressable network. In: ACM SIGCOMM (2001)
14. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling churn in a dht. In: USENIX Annual Technical Conference, pp. 127–140 (2004)
15. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms, pp. 329–350. Germany (2001)
16. Rowstron, A., Kermarrec, A.M., Castro, M., Druschel, P.: SCRIBE: The design of a large-scale event notification infrastructure. In: Networked Group Communication, pp. 30–43 (2001)
17. Sacha, J., Dowling, J., Cunningham, R., Meier, R.: Using aggregation for adaptive super-peer discovery on the gradient topology. In: IEEE International Workshop on Self-Managed Networks, Systems and Services, pp. 73–86 (2006)
18. Sen, S., Wang, J.: Analyzing peer-to-peer traffic across large networks. In: ACM SIGCOMM Workshop on Internet measurement, pp. 137–150 (2004)
19. Stavrou, A., Rubenstein, D., Sahu, S.: A lightweight, robust P2P system to handle flash crowds. IEEE Journal on Selected Areas in Communications **22**(1), 6–17 (2004)
20. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: ACM SIGCOMM 2001, pp. 149–160. CA, USA (2001)
21. Voulgaris, S., Gavidia, D., Steen, M.: CYCLON: inexpensive membership management for unstructured p2p overlays. Journal of Network and Systems Management **13**(2), 197–217 (2005)
22. Zhao, B.Y., Kubiatowicz, J.D., Joseph, A.D.: Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. UCB/CSD-01-1141, U. C. Berkeley (2001)
23. Zorzi, M., Rao, R.R.: Geographic random forwarding (GeRaf) for ad hoc and sensor networks: Multihop performance. IEEE Transaction on Mobile Computing **2**(4), 337–348 (2003)
24. Zorzi, M., Rao, R.R.: Multihop performance of geographic random multihop performance of geographic random. IEEE Transaction on Mobile Computing **2**(4), 349–365 (2003)