The CoQUOS Approach to Continuous Queries in Unstructured Overlays

Lakshmish Ramaswamy, *Member, IEEE,* and Jianxia Chen, *Student Member, IEEE*

Abstract—The current peer-to-peer (P2P) content distribution systems are constricted by their simple on-demand content discovery mechanism. The utility of these systems can be greatly enhanced by incorporating two capabilities, namely a mechanism through which peers can register their long term interests with the network so that they can be continuously notified of new data items, and a means for the peers to advertise their contents. Although researchers have proposed a few unstructured overlay-based publish-subscribe systems that provide the above capabilities, most of these systems require intricate indexing and routing schemes, which not only make them highly complex but also render the overlay network less flexible towards transient peers.

This paper argues that for many P2P applications implementing full-fledged publish-subscribe systems is an overkill. For these applications, we study the alternate continuous query paradigm, which is a best-effort service providing the above two capabilities. We present a scalable and effective middleware called CoQUOS for supporting continuous queries in unstructured overlay networks. Besides being independent of the overlay topology, CoQUOS preserves the simplicity and flexibility of the unstructured P2P network. Our design of the CoQUOS system is characterized by two novel techniques, namely cluster-resilient random walk algorithm for propagating the queries to various regions of the network and dynamic probability-based query registration scheme to ensure that the registrations are well distributed in the overlay. Further, we also develop effective and efficient schemes for providing resilience to the churn of the P2P network and for ensuring a fair distribution of the notification load among the peers. This paper studies the properties of our algorithms through theoretical analysis. We also report series of experiments evaluating the effectiveness and the costs of the proposed schemes.

Index Terms—Peer-to-peer networks, Continuous queries, Publish-subscribe systems, Random walk.

1 Introduction

Unstructured peer-to-peer (P2P)-based content/resource sharing platforms such as Gnutella [1] and Kazaa [2] have experienced tremendous growths in the past decade. The popularity of unstructured P2P networks can be attributed to the simplicity of their designs and their flexibility towards transient node population. Searching in these networks is essentially performed by circulating query messages in an ad-hoc manner and probing individual peer nodes.

Despite their popularity, most of the current unstructured P2P content distribution systems suffer from certain serious limitations. One such limitation is their simple, on demand mechanism for content discovery. Peers in these systems discover data items by circulating queries within the overlay network. A peer receiving a query responds back to the initiating node if it has any matching content. Upon processing a query, the recipient node removes it from its local buffers[1]. Thus, a query *expires* after it completes its circulation within the network. In other words, the network *forgets* the queries once they have completed their circulation. For clarity purposes, we call this the *ad hoc* query model, and we refer to the queries as *ad hoc queries*.

The ad hoc query model suffers from two main shortcomings. First, an ad hoc query is only capable of searching and retrieving content that exists in the P2P network at the time the query was issued. Consider the scenario when a peer P_i issues a query at time T_b. Assume that the query completes its circulation in the network at time T_c. Clearly, this query cannot discover data items that were added after $(T_c + \delta)$, where δ indicates the short duration of time for which the query might be cached at different peers. Further, the query will not reach a peer that joins the network after time T_c, and hence cannot discover matching content on the new peer. In this scenario, the only way for a peer to discover newly added data items would be to repeatedly issue the same query. This is not desirable, since it creates unnecessary traffic within the network. Second, P2P systems that are purely based on the ad hoc query model provide no support for peers to advertise or announce the data items they own to other interested peers. Advertisements are important for P2P applications where peers trade content.

These shortcomings render the ad hoc query model inadequate for many advanced P2P applications. Consider a P2P community of researchers. In such a community, a researcher would not only be interested in searching for research papers, but would also want to be proactively informed when new papers in her research areas are added to the network. In other words, we need a mechanism for the peers to register their long-term

• *Lakshmish Ramaswamy and Jianxia Chen are with the Department of Computer Science, University of Georgia, Athens, GA, 30602.*
E-mail: {laks, chen}@cs.uga.edu

[1] Some systems *cache* recently received queries. But it is done in an ad hoc fashion and for very short durations

interests, and be notified by the system when matching data items are added. As a second example, consider a community comprising of amateur musicians and patrons interested in buying the music produced by the musicians. The musicians would need a mechanism to advertise their new music works to prospective buyers. A naive approach for tackling this problem would be to send advertisements to large subsets of peers through flooding. However, this approach is unviable. Besides heavy messaging overheads, this scheme could overwhelm the peers with unwanted advertisements. What is really needed is a *targeted advertisement* service that sends advertisements to peers who would be interested in the content being advertised.

The well-studied paradigm of publish-subscribe (pub-sub) systems [7], [9], [15] is a possible approach to address the above limitations. The pub-sub interaction paradigm is a *guaranteed notification service* that lets the subscribers to register their interests[2]. When an event producer publishes an event, the system checks the registered subscriptions and generates notifications to all the subscribers who have registered a matching subscription.

Recently, researchers have proposed utilizing unstructured P2P networks for developing pub-sub systems [12], [32]. Unfortunately, implementing these systems is a complicated endeavor. Most of these systems require specialized overlay topologies, and they also involve intricate indexing and routing mechanisms. For example, Sub-2-Sub [32] — a content-based pub-sub system, employs a complex epidemic algorithm to organize the peers into a special overlay in which peers subscribing to the same events are clustered together. These complexities adversely impact the scalability of the systems as well as the flexibility of the underlying P2P network towards transient node populations (please see Section 7 for a discussion on other unstructured P2P-based pub-sub systems).

1.1 Paper Contributions

The difficulties in implementing pub-sub systems on top of unstructured overlays can be attributed to the inherent mismatch between the design requirements of pub-sub systems and the very nature of unstructured P2P systems. In order to provide strict notification guarantee, pub-sub systems have to maintain subscription information in well-organized structures. In contrast, unstructured overlays, by their very nature, are decentralized, loosely coupled and, to certain extent, haphazard. This makes it difficult to build full-fledged pub-sub systems on these platforms.

In this context, an immediate question is *whether guaranteed notification is absolutely necessary for all P2P applications? Or, does a model that provides weaker guarantees, but is much simpler and inexpensive to implement, suffice for certain class of applications?* In this paper, we argue that for a class of P2P applications exemplified by the content distribution scenario, guaranteed notification is not absolutely necessary. Instead, *best effort notification paradigm*, wherein a subscription is informed of most, but not necessarily all, of the data items and advertisements that match its registered interests, is more appropriate for these applications. This assertion is based upon the design principle of most current P2P content sharing systems like Gnutella and Kazaa. Queries in these systems are not guaranteed to discover all matching data items in the networks. Instead, the design objective is to maximize the discovered data items, while ensuring that overheads are not too high.

In this paper, we focus on an alternate notification paradigm called the *continuous query model*. Similar to content-based pub-sub systems [7], [9], [28], this model provides a mechanism through which peers can register their queries, which are maintained in the network for extended durations of time. However, in contrast to traditional pub-sub model, a system implementing the continuous query model provides a best-effort notification service for the registered queries informing their initiating nodes of new content that may have been added in the recent past.

A natural question that comes up is *whether the continuous query model is amenable to efficient and significantly less complex implementations?* In other words, *what techniques and mechanisms are necessary to implement this model on top of generic unstructured overlay networks such that the system is not only effective and scalable but is also resilient to the churn of the overlay network?* Towards answering this question, we design a lightweight middleware called *CoQUOS* (**co**ntinous **q**ueries in **u**nstructured **o**verlay**s**) for supporting continuous queries and advertisements in unstructured P2P networks. One of our goals in designing the CoQUOS system has been to preserve the design simplicity of the underlying unstructured P2P networks, and their flexibility towards network churn. In this regard, the design of the CoQUOS system exhibits two unique features. First, the CoQUOS system does not impose any topological constraints on the underlying P2P network, and it can be implemented as an independent module in any unstructured overlay network. Second, the CoQUOS system does not require complex index structures or routing mechanisms. Instead, our design is based upon very lightweight P2P primitives.

The fundamental idea of the CoQUOS system is to register the continuous query on a set of peers that are located in various topological regions of the overlay network. These query replicas are used by the CoQUOS system to notify the respective source peers of matching advertisements issued by other peers. Considering the decentralized nature of unstructured P2P networks, an important research challenge is to develop a completely distributed mechanism to register the continuous queries at various regions of the P2P network so that the system

[2] Individual implementations of may not provide notification guarantees due to system failures or churn. But the pub-sub paradigm itself is a guaranteed notification service

has high notification effectiveness. Further, it is also necessary to develop low-cost techniques for providing resilience to the churn of the overlay network and for achieving fair distribution of notification load among the peers in the network. Towards addressing these challenges, this paper makes four novel contributions.

- First, we present a novel query propagation technique called *Cluster Resilient Random Walk (CRW)*. This technique retains the overall framework of the random walk paradigm. However, at each step of propagation, CRW favors neighbors that are more likely to send messages deeper into the network thereby enabling the continuous queries to reach different topological regions of the overlay network.
- Second, a *dynamic probability scheme* is proposed for enabling the recipients of a continuous query to make independent decisions on whether to register the query. In this scheme, a query that has not been registered in the past several hops has a higher chance of getting registered in its next hop, which ensures that registrations are well distributed along the path of a query message.
- Third, we discuss a *passive replication*-based scheme for preserving high notification effectiveness of the system even when the underlying P2P network experiences significant *churn*.
- Finally, we propose a local load re-distribution strategy to achieve fair distribution of notification loads among the participating peers.

This paper presents a mathematical analysis of the CRW algorithm which demonstrates that CRW is significantly better in propagating messages to different regions of the overlay when compared with the random walk algorithm. We have also performed a range of experiments to study the properties of the CoQUOS system. The results show that the proposed techniques are effective and efficient.

2 The CoQUOS System

In this section, we present a high-level description of the CoQUOS system architecture. We begin by introducing a few concepts that are used in the rest of the paper.

2.1 Concepts and Notations

Consider an unstructured P2P system comprising of peers $\{P_0, P_1, \ldots, P_{N-1}\}$. Let $\{L_0, L_1, \ldots, L_{M-1}\}$ represent the logical links (connections) in the network. For simplicity, we assume that the links are bidirectional. Two peers P_i and P_j are said to be *neighbors* of each other if there exists a link $L_v = (P_i, P_j)$ connecting them. The network is dynamic with peers entering and leaving the network at arbitrary points in time. Further, new links may be established in the network and existing links torn down. We assume that each data item D_r in the system has associated metadata (represented as $MData(D_r)$) that describes it. In the current context, the metadata is a list of keywords describing the data item.

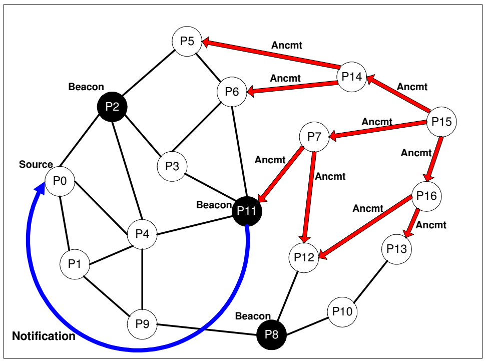


Fig. 1. CoQUOS System Overview

Continuous query is the means through which a peer can register its long term interests with the CoQUOS system. A continuous query, represented as $Q = (SID, Predicate, VTime)$, is essentially a tuple of three components, namely, *source ID* (SID) , *query predicate* ($Predicate$) and *validity time* ($VTime$). The source ID uniquely identifies the peer issuing the query. The query predicate is the matching condition of the query, and is used by the source peer to specify its interests. In general, the predicate can be of any form such as range predicates or even a regular expression. We assume that the predicate is a list of keywords describing the content the source peer is interested in. Validity time ($VTime$) represents the time until which the source node is interested in receiving notifications.

Peers announce their new data items through *announcements*. An announcement is represented as $Ad = (AID, MData)$. The *announcing peer ID* (AID) identifies the advertising peer and the metadata ($MData$) is the metadata of the content being advertised. A data item D_r (and analogously its announcement) is said to *match* a continuous query Q_m, if D_r's metadata contains *all* the keywords in Q_m's predicate.

2.2 Design Overview

Our goal in designing the CoQUOS system is to design a highly effective notification service on top of arbitrary unstructured overlay networks. A common way of quantifying notification effectiveness is through the overall notification success rate of the system. However, our design of the CoQUOS middleware strives for a stronger notion of notification effectiveness, wherein the goal is not only to achieve high overall success rate, but also reasonably high success rates for each individual query.

Our system works essentially by maintaining each continuous query at one or more peers of the overlay. If a continuous query Q_m is registered at a peer P_i, then P_i is called the *beacon node* of the query Q_m. A peer that registers a query implicitly agrees to assume the responsibility of notifying the source node of any matching data items that it might discover. Beacon nodes discover new data through incoming peer announcements. When a peer P_i receives an announcement message, it checks all the queries that are registered to see if any of the queries match the received announcement. Upon finding a matching query Q_i, P_i sends a notification to Q_i's source node. In addition, beacon nodes can take more pro-active roles, and they may periodically circulate the registered queries in their close vicinity to search for new data items. In this paper, however, we only consider peer announcement-based data item discovery by beacon nodes.

In the current design, the announcements are circulated through a Gnutella-like broadcast scheme [1]. However, in order to ensure that the communication overheads of the system are small, the TTL of the announcement messages are set to very low values. Specifically, a peer that needs to announce a new data item, creates an announcement message with corresponding keywords, sets the TTL to a pre-specified value (Announcement TTL) and sends the message to all its neighbors. The recipients of the message decrement the TTL and send it to their neighbors. This process is repeated until the TTL reaches zero. Thus, an announcement essentially reaches the set of nodes that are within a small number of hops from the peer issuing it. We are currently exploring advanced strategies such as iterative deepening and directed breadth first search [34] for further reducing the announcement message load.

Figure 1 illustrates the functioning of CoQUOS middleware. Peers $\{P_2, P_8, P_{11}\}$ are the beacon nodes of a continuous query issued by P_0. The node P_{15} issues an announcement that matches the query. The TTL of the announcement is set to 2. This announcement reaches the beacon node P_{11}, which notifies P_0. A source peer may receive multiple notifications for the same advertisement, in which case it ignores all but the first notification.

3 Selecting Beacon Nodes of a Query

The discussion in the previous section highlights the crucial role played by the beacon nodes in notifying the source peer of matching data items. Hence, the choice of beacon nodes would have a significant impact on the notification success rates of a continuous query. So, an important research question is: *how do we select the set of peers that will serve as the beacon nodes of a query?*. In other words, *which set of peers should host a particular continuous query in order to realize high notification effectiveness?*

Towards answering this question, let us first understand the characteristics of a good beacon set. First and foremost, the beacon nodes of a query should be distributed in every major region of the overlay network. This implies that most peers in the network should be reachable from at least one node in the beacon set in a very small number of hops. This property is essential for achieving high notification success rates, since the announcements reach only the peers located within a small number of hops from their respective source peers. Second, the beacon nodes of a query should not be located very close to one another. If there are many registrations in close proximity, a single announcement would reach multiple beacon nodes of the same query, thus generating duplicate notifications.

The highly decentralized nature of unstructured overlay network makes the problem of selecting beacon nodes that satisfies the above two properties particularly challenging. The CoQUOS system incorporates a completely decentralized technique for beacon node selection. In this scheme, a continuous query is circulated in the network (by neighbor forwarding), and each peer that receives the continuous query decides independently whether to register and store the query. An important feature of our scheme is that although each peer makes a local decision regarding query registration, the resulting beacon node sets manifest the above two important characteristics to a very high degree.

In this context, we need to address two important problems: **(1)** *What mechanisms should be adopted for circulating continuous queries?*; and **(2)** *How should a peer receiving a continuous query decide whether to register the query?* We answer these questions by describing the two novel components of our beacon node selection scheme, namely *cluster-resilient random walk (CRW)* mechanism for query dissemination and *dynamic probability (DP)* technique for query registration.

3.1 Cluster Resilient Random Walk

Flooding-based broadcast is an option for circulating continuous queries. However, this would be analogous to a breadth first traversal of the network. As previous studies have reported, in this scheme, messages remain in close vicinity of the source node and do not go deep into the network [18], [19]. Random walk is another message propagation paradigm that has received considerable attention from the P2P research community [18], [23]. In the context of P2P networks, random walk works as follows: When a peer node P_i receives a message whose TTL has not expired, it selects one of its neighbors completely at random and forwards the message to that peer. Since, at each step the message is forwarded to only one neighbor, the message load imposed by random walk is very low. Random walk corresponds to a depth-first traversal of the network, and a message propagated through random walks has a higher probability of reaching remote regions of the network than its flooding-based counterpart. In this paper we use the terms random walk and pure random walk (PRW) interchangeably.

The above property of the random walk makes it an attractive paradigm for propagating continuous queries. Unfortunately, the random walk protocol suffers from one significant drawback that undermines its utility for propagating queries in the CoQUOS system. Prior studies have shown that the ability of random walk in propagating messages to remote topological regions diminishes significantly on overlay networks that exhibit significant degrees of node clustering [18]. In these

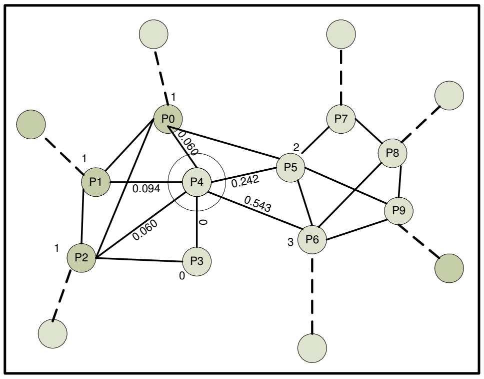


Fig. 2. Illustration of Cluster Resilient Random Walk

networks there are distinct clusters of nodes with large numbers of connections among them, whereas the connections flowing across clusters are comparatively small in number. For these networks, the random walk protocol suffers from the following drawback. When a message enters a cluster, it is likely to keep circulating within the cluster for large number of hops before exiting the cluster. Thus, the message *spends* a significant fraction of its TTL before reaching a different topological region of the overlay. In Figure 2, a random walk message that reaches peer P_4 has a high probability of visiting majority of the other peers in P_4's cluster (i.e., peers $\{P_0, P_1, P_2, P_3\}$), possibly multiple times, before going to other regions of the network. In other words, the message gets *trapped* in the cluster for considerable number of hops thereby adversely affecting its ability to reach remote regions of the overlay.

Towards overcoming the above drawback, we have designed a novel query dissemination scheme called *cluster resilient random walk* (CRW). This scheme is motivated by a crucial observation: *Two peers belonging to the same cluster generally have large numbers of common neighbors*. Thus, a peer P_j has a lesser likelihood of being in the cluster of another peer P_i, if a large fraction of P_j's neighboring peers are not neighbors of P_i. Based on this observation, the CRW scheme forwards messages to *out-of-cluster* nodes with high probability, thereby mitigating the possibility of messages getting trapped in clusters. In this scheme, a peer computes the overlap between its neighbor list and those of each of its neighbors, and uses this information while making message forwarding decisions. A peer that has little overlap has higher probability of receiving the query in the next hop, and vice-versa.

Let $NbrList(P_j)$ denote the list of neighbors of P_j. Let the peers P_k, P_{k+1}, P_{k+2}, ..., P_{k+l} denote the neighbors of the node P_j. Let $UniqueNbrs(P_{k+1}, P_j)$ denote the set of neighboring peers of P_{k+1} that are *not* the neighbors of P_j (i.e., $UniqueNbrs(P_{k+1}, P_j) = NbrList(P_{k+1}) - (NbrList(P_j) \cap NbrList(P_{(k+1)}))$). Suppose the node P_j receives a query message from a neighboring peer P_k. In the CRW algorithm the probability of a neighbor P_{k+1} receiving the message in the next hop (represented as $FwdProbability(P_{K+1})$) is proportional to $(\frac{UniqueNbrs(P_{k+1}, P_j)}{NbrList(P_{(k+1)})})^{\lambda}$. Normalizing over all the neighboring nodes of P_j except P_k we obtain:

$$FwdProbability(P_{K+1}) = \frac{(\frac{UniqueNbrs(P_{k+1}, P_j)}{NbrList(P_{(k+1)})})^{\lambda}}{\sum_{P_l \in \{NbrList(P_j) - P_k\}} (\frac{UniqueNbrs(P_l, P_j)}{NbrList(P_l)})^{\lambda}} \quad (1)$$

Notice that CRW biases the $FwdProbability$ of a neighbor according to the corresponding $UniqueNbrs$ value. λ controls the extent of bias, and therefore it is referred to as the *bias factor*. Larger values of λ induce stronger bias and vice-versa. If λ is set to 0, there is no bias, and CRW becomes equivalent to PRW. Hence, the random walk technique can be viewed as a special case of the proposed approach. The appropriate λ value depends upon the topology of the network under consideration. In general, if the network exhibits high degree of clustering among its nodes, λ should be higher values (≥ 3). However, setting λ to extremely high values is equivalent to *always* forwarding the message to the neighbor with the highest $|UniqueNbrs|$ value. This eliminates all randomness from the scheme, which is not generally preferable.

Figure 2 illustrates a single step in the CRW message propagation scheme. The query message is at the node p_4 and λ is set to 2. For each neighboring node of p_4, we show its $|UniqueNbrs|$ value with respect to p_4. The numbers on the edges indicate the probabilities of forwarding the query along that link. Notice that the message has a very low probability of remaining within p_4's cluster (comprised of $\{p_0, p_1, p_2, p_3\}$) in the next hop. The probability of the message remaining in p_4's cluster is about 0.22 for CRW as against 0.66 for PRW.

In a nutshell, the query propagation in the CoQUOS system works as follows. The source peer creates a query message initializing its TTL to a default value. Each peer along the query's path forward the message to one of their neighbors according to their $FwdProbability$ values. The TTL is decremented at each hop, and the process terminates when the TTL becomes 0. Notice that the CRW scheme requires the peers to maintain the connectivity information of their neighbors. A legitimate question, therefore, is what are the storage, computational, and communication costs of maintaining this information. In Section 4.3, we discuss these overheads and also present implementation strategies to minimize them.

Our experiments (see Section 6) show that CRW is better than both PRW and flooding in disseminating queries to various regions of the P2P overlay. However, the fact that CRW forwards the query to only one neighbor at each hop can make it less effective on overlays with large numbers of peer clusters which are weakly connected (through very few peers) to other parts of the network. Introducing multiple random walkers (traversing multiple paths) is an attractive option to overcome this limitation. Having multiple walkers may also shorten the time required for query registration. However, the challenge is to decide when and where to introduce multiple walkers. In a related work [11], we have proposed the *message fission* technique to address this challenge. Broadly, the idea is to *split* a random walk message when it reaches a *bridge peer* (a peer that anchors two or more regions of the network with very sparse inter-region links), and forward the child messages to multiple neighbors. The TTL of these child messages are assigned such that their sum is one less than the TTL of the original message before splitting. The message fission technique includes a decentralized mechanism to detect bridges peers. Our experiments showed that CRW augmented with the fission technique (called FA-CRW) is very effective in overcoming the above limitation. Although, the current CoQUOS system uses CRW for query propagation, it can support FA-CRW without too much additional efforts.

3.2 Dynamic Probability Based Query Registration

The CRW scheme provides a mechanism for propagating a continuous query. But, how does a node receiving this message decide whether to register the query? A straightforward solution would be to register a query at every node it visits. However, this would result in large numbers of unnecessary subscriptions, which affects the efficiency of the network. Alternatively, each peer receiving a query message can decide register it with a certain fixed probability, say Rp. We call this scheme the *fixed probability-based query registration scheme (FP scheme, for short)* . Although this strategy seems intuitive, it cannot guarantee high notification success rates for every query. The experiments in Section 6 confirms our contention in this regard. The reason is that for some continuous queries a long series of peers in the path of the query message may all decide not to register the query, whereas another sequence of consecutive nodes may all decide to host the query. The announcements originated near the *dry patches* of a query's path might fail to reach any of its beacon nodes, thus leading to low success rates.

Considering these requirements, we have designed a novel *dynamic probability*-based technique *(DP scheme, for short)* for peers to decide whether to register a continuous query. As in the fixed probability scheme, a peer receiving a query-message registers it with certain probability. However, the registration probability of a query varies as the query traverses along its route. The central idea of the dynamic probability scheme can be summarized as follows: *The probability of registering a query at a peer node would be high if the query has not been registered at the nodes it visited in the recent past. In contrast, if the query has been registered at a node that visited in the past few hops, the probability of it getting registered at the current peer would be low.*

Specifically, the scheme works as follows. Each continuous query message Q_m is associated with a value called *registration probability* ($Rp(Q_m)$). The registration probability of a query message indicates the probability that it would be registered at the peer it visits next. When a peer issues a query, the registration probability is set to an initial value (called *initial probability*). When a peer P_i receives a query message Q_m, it registers the query with probability $Rp(Q_m)$. If P_i registers the query, it also resets the value of $Rp(Q_m)$ to the default initial value, before forwarding the message to one of its neighbors. On the other hand, if P_i decides not to register the query, it increments $Rp(Q_m)$ by a pre-determined amount (called *probability increment*) before forwarding the query to one of its neighbors. Thus, the registration probability value associated with a query message keeps increasing until it gets registered at a peer, at which point it falls suddenly to the default initial value. The number of beacon nodes of a query can be controlled through the initial probability and probability increment parameters. Higher values of these parameters result in larger number of subscriptions and vice-versa.

Experiments show that our decentralized beacon node selection scheme comprising of CRW and the DP query registration scheme, not only yields significant improvement in the overall success rates, but also ensures reasonably high individual success rates for all queries.

4 Enhancements and Discussion

The loosely-coupled and highly dynamic nature of underlying P2P network poses several additional challenges. In this section, we discuss two issues that are of particular importance to the performance of the CoQUOS system, namely **(a)** *Churn of the P2P overlay*; and **(b)** *Load distribution among peers*. We also present a brief discussion on the implementational aspects of the CoQUOS system.

4.1 Overlay Churn

P2P networks are, in general, highly dynamic systems, with nodes entering and exiting the system quite frequently. This *churn* of the overlay network can adversely impact the success of continuous queries and announcements. When a node P_i gracefully leaves the system, it asks one of its neighbors to handle all registered queries at P_i and also notifies all the beacon nodes with queries issued by P_i to remove the queries. However, when P_i exits the system unexpectedly, all the registrations are lost and the notification success rates of the respective queries and the matching announcements drop. Thus, effective mechanisms are needed to alleviate the negative effects of churn in the overlay network.

In order to counter the adverse effects of network churn, we have designed a low-cost technique wherein the query registrations present on a peer are replicated on one or more of its neighbors. Concretely, the query registrations present on a peer P_i are replicated at r_f (replication factor) of its neighbors. If P_i fails, the failure will be noticed by a neighbor, say P_k, that maintains a replica of subscriptions registered at P_i. P_k claims ownership of the queries registered at the failed node by sending messages to other neighbors of P_i, and receiving consent from them. Simultaneous ownership claims by multiple neighbors will be resolved in favor of the peer with smallest ID. Once P_k receives consent from other neighbors, it assumes the query notification functionalities of P_i (i.e., P_k becomes the new beacon node of the queries that were registered at P_i) and notifies the source nodes of the queries about the takeover. Failures are detected through periodic exchange of *heartbeat messages* between the beacon node and the peers maintaining its replicas. The beacon node that does not respond to two consecutive messages is assumed to have failed.

In the interest of better load distribution, two or more neighbors may takeover subsets of the queries registered at the failed node. The communication costs of maintaining query replicas are optimized through *lazy replication* and *piggybacking* (please see Section 4.3).

4.2 Load Balancing

Achieving good load distribution among peers is another important requirement for the performance of the CoQUOS system. The number of queries and the numbers of notifications sent out per unit time by various nodes represent two key load metrics for CoQUOS system. These load parameters can vary widely among the nodes of the CoQUOS system due to variety of reasons, including topological characteristics of the network, skewed announcement and query popularities, variation in the resource availabilities at peers or a combination of these factors. Irrespective of the cause, load imbalances not only degrade the performance of the system, but may also cause overloaded peers to exit the network.

Ensuring good load balancing in decentralized, loosely coupled systems such as unstructured P2P overlays is challenging. In fact, achieving optimal load balancing on a global scale may turn out to be prohibitively expensive as it would require collection and maintenance of load information on a global scale. Skip graphs have been used to alleviate some of the problems associated with global load balancing in the context of range data [16]. However, skip graphs require maintenance of circular linked lists consisting of all nodes in the system. Hence, they are not suitable for a large-scale, dynamic environment like unstructured P2P content distribution platforms.

Considering the above discussion, our goal for the CoQUOS system is not to achieve perfect load balancing. Rather, our aim is to ensure good load balancing in an efficient and scalable manner. The CoQUOS system includes two schemes for balancing notification loads among its peers. Each of these schemes can be used exclusively, or they may be used simultaneously. Keeping in mind the decentralized and loosely coupled nature of the P2P network, these techniques have been designed to be localized both in terms of their operations and the information they utilize. They only require interactions between neighboring peers, thus making them suitable for generic unstructured P2P networks. In both schemes, neighboring peers periodically (at the end of pre-specified cycles) exchange information about their loads. Based on the load information obtained from its neighbors, a peer decides whether it is *overloaded*. A peer is P_i is overloaded if the ratio of P_i's load in the previous cycle to the average load of its neighbors in the previous cycle exceeds a user-specified threshold α.

Our first scheme, called the *active load balancing scheme*, is a pro-active strategy. In this scheme, an overloaded peer P_i sheds some of its load by asking one or more of its less-loaded neighbors to take over a few of the queries that are current registered at P_i. The set of queries selected for offloading is such that, after offloading, the load on P_i becomes approximately equal to the mean of the loads on its neighbors. Mathematically, suppose $\{Q_0, Q_1, \ldots, Q_t\}$ denote the set of queries registered at P_i. For simplicity, we assume that the peers are similar in terms of their resource availabilities. However, the discussion can be extended to the scenario wherein the peers are heterogeneous in terms of their resource capabilities. Let $LD(Q_d)$ denote the load caused by the continuous query Q_d on P_i in the previous cycle, and let $CLD(P_i)$ indicate the cumulative load on the peer P_i in the previous cycle due to all the queries registered at P_i. Let $OFQ(P_i)$ denote the set of queries that would be offloaded at the end of the cycle. $OFQ(P_i)$ is formed such that $CLD(P_i) - \sum_{Q_d \in OFQ(P_i)} LD(Q_d) \approx (\frac{\sum_{P_k \in NbrList(P_i)} CLD(P_k)}{|NbrList(P_i)|} + \frac{\sum_{Q_d \in OFQ(P_i)} LD(Q_d)}{|NbrList(P_i)|})$.

The second strategy called the *passive load balancing scheme* is extremely light-weight. In this scheme, an overloaded peer avoids registering any new queries until the load becomes more balanced. Instead, it just increments the registration probability of the query message and forwards it. This ensures that the query will be registered on another peer within the next few hops.

In our experiments, we use average number of notifications per unit time as the load metric. We believe our load balancing scheme would work equally well if number of registered queries is used as the load metric.

4.3 Discussion

We now discuss the overheads associated with the various component mechanisms of the CoQUOS system, and present implementation techniques to alleviate these overheads.

The CoQUOS system requires the participating peers to store a set of queries. Each query essentially consists of a few keywords and the respective source peer identifiers. Thus, the overheads of storing queries at peer identifiers are very small. Upon receiving an announcement, the peer has to check whether there are any subscriptions registered at it that match the incoming announcement. For each matching subscription, the peer has to notify the corresponding originating peers. However, as our experiments show, the communication loads due to notifications are, in general, very low. In order to efficiently process incoming announcements, beacon nodes index the queries registered at them through their respective keywords. With this index in place, processing incoming announcement at an individual peer involves three straightforward steps, namely, extracting keywords from the announcement, retrieving matching queries using the index, and sending out notifications to the respective source peers.

Similar to PRW, the number of messages due to an individual continuous query is equal to the initial TTL of the query. But, for implementing the CRW algorithm, each peer needs to store and maintain connectivity information (neighbor list) of its neighbors. We note that the memory requirements for storing the neighbor lists are minimal (on average, each peer has to store d^2 peer identifiers, where d is the mean degree of the peers in the network). However, because of the churn in the overlay network, the connectivity of the peers can change over time. In order to ensure consistency of the stored copies of neighbor lists, an arbitrary peer P_i whose connectivity changes has to inform each of its neighbors of the change. This could introduce non-negligible traffic into the network.

The CoQUOS system includes two mechanisms to alleviate the above traffic overhead. First, we adopt a *lazy update strategy* wherein the copies of the connectivity information are made consistent periodically (or when the connectivity information of the peer has undergone significant change). This allows the peers to accumulate changes occurring over certain time period and reflect them all at once. Second, peers *piggyback* connectivity information updates on announcement and query messages, which further reduces the message overhead. The above two mechanisms can be combined as follows. Upon change to its neighbor list, the peer P_i waits until a pre-specified amount of time (or until the percentage of change exceeds a pre-specified threshold) before sending explicit messages containing the new neighbors list to its neighbors. During this wait time, suppose the peer P_i has to send an announcement or a query message to its neighbor P_j, it sends the updated neighbors list along with the announcement/query message. At the end of the pre-specified time duration (or, if the percentage of change exceeds the threshold), P_i sends an explicit neighbor list update message to each neighbor peer that has not already received updated neighbor list from P_i.

Our churn resilience mechanism requires the peers to store a copy of the queries they have registered at one or more of their neighbors. The storage overheads of replicating queries directly depend upon the replication factor r_f. However, our experiments show that even when at very small r_f values the success rates improve considerably. We again adopt the lazy replication technique along with piggybacking for ameliorating the message costs of replica maintenance. The load distribution scheme requires one load-reporting message between each pair of neighboring peers at the end of every load balancing cycle. There might be a few additional load re-distribution messages. However, the cumulative traffic generated by these system maintenance tasks is, in general, very low.

TABLE 1
Notations

N	Total nodes in the network
d	Average node degree
M	Total clusters in the network
K	Average number of peers in each cluster
w	Average number of in-cluster neighbors
v	Average number of out-of-cluster neighbors

TABLE 2
Comparison of Pr_{Out}^{PRW} and Pr_{Out}^{CRW}

d	w	Pr_{Out}^{PRW}	Pr_{Out}^{CRW}		
			$\lambda = 1$	$\lambda = 3$	$\lambda = 5$
10	5	0.5	0.53	0.60	0.66
10	8	0.2	0.27	0.44	0.63
20	10	0.5	0.57	0.70	0.81
20	15	0.25	0.43	0.79	0.95

5 Analysis of CRW Technique

In this section, we present a theoretical analysis of the CRW scheme. We compare CRW with PRW and show that CRW has better ability to reach different topological regions of the network, especially, if the network exhibits considerable degree of clustering among its peers.

Consider a network of N peers exhibiting node clustering characteristics. Let the average degree of peers be d. To simplify the analysis, we make a few assumptions about the topology of the peer network. Suppose the peer network has M node clusters with each cluster containing $K = \frac{N}{M}$ peers on average. The fact that the P2P network under consideration exhibits distinct clusters implies that each node in the network maintains connections to a considerable fraction of nodes within its own cluster. Let us assume that a node maintains connections to w in-cluster peers on average. Since the average degree is d, the average number of connections to out-of-cluster peers is $v = d - w$. These notations are tabulated in Table 1.

With these assumptions, we analyze the PRW and CRW schemes with respect to the probability of a query message Q_m, that has reached an arbitrary peer P_i, moving out of P_i's cluster in the next hop. We represent this probability as Pr_{Out}.

In the PRW scheme, the message is sent to anyone of P_i's neighbors with equal probability. P_i has d neighbors of which w are in-cluster neighbors. Hence the probability that the message will go out of P_i's cluster is $Pr_{Out}^{PRW} = \frac{d-w}{d}$.

Now, let us consider the CRW scheme. Recall that in the CRW scheme, the probability that of the message being sent to a neighbor P_g is proportional to $(\frac{|UniqueNbrs(P_g,P_i)|}{|NbrList(P_{(g)})|})^{\lambda}$, where $NbrList(P_{(g)})$ denotes the neighbor list of P_g and $UniqueNbrs(P_g, P_i)$ represents the neighbors of P_g that are not neighbors of P_i (i.e., $UniqueNbrs(P_g, P_i) = \{NbrList(P_{(g)})\} - \{NbrList(P_{(g)}) \cap NbrList(P_i)\}$).

Let P_f be an arbitrary in-cluster peer of P_i. $|NbrList(P_f)|$ is d on average. Now let us estimate the cardinality of $UniqueNbrs(P_f, P_i)$. The probability that an arbitrary node P_q that belongs to the same cluster as P_i and P_f to be in the neighbor list of P_i is $\frac{w}{K}$ (since there are K nodes in P_i's cluster and w in-cluster nodes in $NbrList(P_i)$). Similarly, the probability that p_q is a neighbor of P_f is $\frac{w}{K}$. Hence the probability that P_q is a neighbor of both P_i and P_f is $(\frac{w}{K})^2$. The probability of any in-cluster peer to be in $(Nbr(P_i) \cap Nbr(P_f))$ is $(\frac{w}{K})^2$. As the average number of peers in each cluster is K, the expected value of the number of in-cluster peers in $(Nbr(P_i) \cap Nbr(P_f))$ is $K \times (\frac{w}{K})^2$ which is $\frac{w^2}{K}$. Similarly, the expected number of out of cluster peers that are in $(Nbr(P_i) \cap Nbr(P_f))$ is $\frac{(d-w)^2}{(N-K)}$. Since $N >> K$, $\frac{(d-w)^2}{(N-K)} \approx \frac{(d-w)^2}{N}$. Hence, the expected cardinality of $(Nbr(P_i) \cap Nbr(P_f))$ is $(\frac{w^2}{K} + \frac{(d-w)^2}{N})$. Therefore the expected value of $|UniqueNbrs(P_f, P_i)|$ is $d - (\frac{w^2}{K} + \frac{(d-w)^2}{N})$, which simplifies to $\frac{(NKd - Nw^2 - K(d-w)^2}{NK}$. Therefore the probability that P_f receives the message in the next hop is proportional to $(\frac{(NKd - Nw^2 - K(d-w)^2}{NKd})^{\lambda}$.

Consider an out-of-cluster node P_h. Again $|NbrList(P_h)| = d$, the average node degree. Now we estimate the cardinality of $UniqueNbrs(P_h, P_i)$. Since $UniqueNbrs(P_h, P_i) = \{NbrList(P_{(h)})\} - \{NbrList(P_{(h)}) \cap NbrList(P_i)\}$, we need to estimate the cardinality of $\{NbrList(P_{(h)}) \cap NbrList(P_i)\}$. A peer which belongs to $(NbrList(P_i) \cap NbrList(P_h))$ can be of three types: (1) It can be in the same cluster as P_i; (2) It can be in the same cluster as P_h; or (3) It may be outside both clusters. Consider a node P_r that belongs to the P_i's cluster. Probability of P_r belonging to $NbrList(P_i)$ is $\frac{w}{K}$, and it belonging to $NbrList(P_h)$ is $\frac{d-w}{N-K}$ (since there are $N-K$ nodes outside P_h's cluster and the number out-of-cluster neighbors of P_h is $(d-w)$). Therefore, the expected number of nodes from P_i's cluster that belong to $(NbrList(P_i) \cap NbrList(P_h))$ is $\frac{w \times (d-w)}{(N-K)}$, which can be approximated to $\frac{w \times (d-w)}{(N)}$ as $N >> K$. Identically, the expected number of nodes from P_h's cluster that belong to $(NbrList(P_i) \cap NbrList(P_h))$ would be $\frac{w \times (d-w)}{(N)}$. Along similar lines it can be shown that the expected number of nodes that neither belong to P_i's cluster nor to P_h's cluster, but are in $(Nbr(P_i) \cap Nbr(P_h))$ is $\frac{(d-w)^2}{(N-2K)}$, which can be approximated to $\frac{(d-w)^2}{(N)}$, as $N >> K$. The expected cardinality of $(NbrList(P_i) \cap NbrList(P_h))$ is therefore equal to $\frac{2 \times w \times (d-w)}{N} + \frac{(d-w)^2}{N}$. Thus, the expected value of $|UniqueNbrs(P_h, P_i)|$ is $d - (\frac{d^2-w^2}{N})$. The probability that the message is sent to P_h is proportional to $(\frac{dN-(d^2-w^2)}{dN})^{\lambda}$.

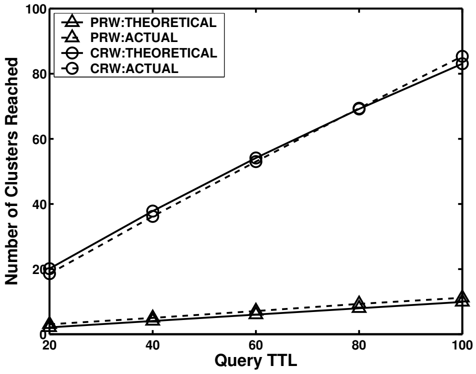


Fig. 3. Comparison of Theoretical and Experimental Values of $E(C_H^{PRW})$ and $E(C_H^{CRW})$

Now, there are w in-cluster neighbors of P_i each of which can receive the message with a probability that is proportional to $(\frac{(NKd - Nw^2 - K(d-w)^2}{NKd})^{\lambda}$. Similarly, there are $(d-w)$ out-of-cluster neighbors of P_i each of which can receive the message with a probability that is proportional to $(\frac{dN-(d^2-w^2)}{dN})^{\lambda}$. Therefore, with the CRW scheme, the message moves out of the cluster with a probability Pr_{Out}^{CRW} which is given by the equation:

$$Pr_{Out}^{CRW} = \frac{[K(dN - d^2 + w^2)]^{\lambda}(d-w)}{[K(dN - d^2 + w^2)]^{\lambda}(d-w) + [dNK - Nw^2 - K(d-w)^2]w} \tag{2}$$

In order to give an insight into the actual numbers the expressions for Pr_{Out}^{PRW} and Pr_{Out}^{CRW} yield, we tabulate their values for a network containing 5000 peers and 250 clusters (i.e., $N = 5000$, $M = 250$ and $K = 20$) at various values of d (average degree of peers), w (average in-cluster degree of peers) and λ (bias factor for CRW). Table 2 gives these values. As the results show, the probability of message moving out of the current cluster is significantly higher for CRW. This enables CRW to reach larger number of network regions.

Next, we derive an expression for the expected number of clusters traversed by message of PRW and CRW scheme. We first derive a generic expression that could be used to compute the expected number of traversed clusters for either PRW or CRW scheme. We later discuss the specific expressions for both schemes.

Consider a message forwarding mechanism wherein the probability of the message moving out of a cluster is Pr_{Out} at each step of its movement in the P2P network. Let $E(C_H)$ denote the expected number of clusters reached by the message in H hops. Now let us calculate the value of $E(C_{H+1})$ in terms of $E(C_H)$. The message remains in the same cluster with probability $(1 - Pr_{Out})$. Thus with probability $(1 - Pr_{Out})$, $E(C_{H+1})$ remains the same as E_H. Although the message goes out of the current cluster with probability of Pr_{Out}, it might not always reach a previously undiscovered cluster. Assuming that a message moving out of a cluster is equally likely to go to any of the other $(M-1)$ clusters, we see that a message moving out of a cluster can go back to a cluster through which it has traversed before with a probability $\frac{E(C_H)}{M-1}$. A message moving out of the current cluster reaches a previously undiscovered cluster with probability $1-\frac{E(C_H)}{M-1}$. Hence, the expected number of clusters traversed by a message in $H+1$ hops is given by the equation:

$$\begin{aligned} E(C_{H+1}) &= (1-Pr_{Out})E(C_H) + Pr_{Out}[\frac{E(C_H)}{M-1} \times E(C_H) \\ &\quad +(1-\frac{E(C_H)}{M-1}) \times (E(C_H)+1)] \\ &= Pr_{Out} + E(C_H) \times (1-\frac{Pr_{Out}}{(M-1)}) \end{aligned}$$

The boundary condition for this recursive equation is given by $E(C_0) = 1$, since a message is always assumed to have traversed through the cluster where it was initiated. Solving the recursive equation, we obtain $E(C_H) = (M-1)(1-(1-\frac{Pr_{Out}}{(M-1)})^H) + (1-\frac{Pr_{Out}}{(M-1)})^H$.

The expected number of clusters traversed by messages of the PRW and CRW schemes can be obtained by substituting the respective Pr_{Out} values into the above equation. Thus the expected number of clusters traversed by a message in H hops for the PRW scheme is $E(C_H)^{PRW} = (M-1)(1-(1-\frac{Pr^{PRW}_{Out}}{(M-1)})^H)+(1-\frac{Pr^{PRW}_{Out}}{(M-1)})^H$. Analogously, for CRW it is given by $E(C_H)^{CRW} = (M-1)(1-(1-\frac{Pr^{CRW}_{Out}}{(M-1)})^H) + (1-\frac{Pr^{CRW}_{Out}}{(M-1)})^H$

Validation of Analysis

We validate the theoretical results by simulating the PRW and the CRW schemes on synthetic networks. We have generated several networks. Each of these networks exhibits distinct clusters. Each node is randomly connected to several in-cluster peers and a few out-of-cluster peers. We simulate PRW and CRW on these networks and measure the average number of distinct clusters reached by messages.

Figure 3 indicates the theoretical values of the expected number of clusters reached by PRW and CRW as well as the corresponding values obtained through simulations. The parameters of the network are as follows: $N = 5000$, $M = 250$, $K = 20$, $w = 18$. λ is set to 5. The TTL of the query varies from 20 to 100. As the results show, the actual values from the simulation are very close to the theoretical values computed through our formulae. Further, CRW has better capability to reach various regions of the network, which makes it appropriate for propagating queries in the system.

6 Experimental Evaluation

The objectives of our experimental study of the CoQUOS system are four fold: **(1)** Evaluating the effectiveness of the CRW technique in propagating continuous queries; **(2)** Studying the performance of the dynamic probability approach for query registrations; **(3)** Evaluating the churn resilience and the load balancing mechanisms; and **(4)** Evaluating the communication costs.

We have developed a Java-based simulator of the CoQUOS system. The inputs to the simulator include the overlay network, the keyword corpus, and the values of the various parameters of the scheme being simulated. Queries and announcements are periodically issued by randomly chosen peers. For simplicity, it is assumed that each query and announcement contain one keyword chosen at random from the keyword corpus consisting of $10,000$ words. Our experiments use power-law distributions for keyword popularities, as prior studies have reported similar distributions for real-world P2P network like Gnutella [29]. Peer exits are simulated by suspending the activities of the corresponding nodes and terminating their connections to other nodes. When a peer re-enters the system, it establishes new connections and resumes its activities. We assume that peers retain their IDs when they exit and re-enter the system. An individual peer's entry and exit from the P2P network are modeled as Poisson processes.

We use various network topologies for our experiments. Majority of our experiments are upon 5000-node power-law (Zipf) network with exponent value set to 0.9 and a 5000-node uniform random network. For evaluating the impact of the exponent value of power-law network, we use a set of 5000-node power-law networks with exponent values ranging from 1.0 to 2.5. In addition, for studying the effects of node clustering on the performances of various query propagation schemes, we use a set of small world graphs with varying cluster coefficients. Cluster coefficient of a network indicates the extent to which the nodes of network are clustered [8], [33]. It varies from 0.0 to 1.0. The higher the cluster coefficient, the more clustered the network, and vice-versa. Given the topology type and the corresponding parameters, our overlay generator repeatedly selects a pair of nodes and introduces a connection between them while constantly ensuring that the overlay satisfies topology restrictions. If at the end of this process, the overlay is not connected, just enough links are introduced between various components to obtain a connected network. Table 3 indicates the key properties of the different topologies we have used in our experiments.

6.1 Performance Metrics

We use two performance metrics for quantifying CoQUOS middleware's effectiveness, namely *mean notification success rate* and *minimum notification success rate*. Consider a continuous query Q_m that was issued by peer P_i. Let $MCount(Q_m)$ denote the total number of matching announcements issued in the validity duration of Q_m. Suppose P_i was notified of $NCount(Q_m)$ of these announcements. The notification success rate of Q_m ($NSR(Q_m)$) is defined as $NSR(Q_m) = \frac{NCount(Q_m)}{MCount(Q_m)}$. The *mean notification success rate* (mean NSR for short) of the system is defined as the average of the $NSRs$ of all the queries that were issued during the observation time duration, whereas *minimum notification success rate* (minimum NSR for short) is their minimum.

TABLE 3
Topological Characteristics of Networks

	Total Nodes	Node Degree				Cluster Coeff.
		Avg.	Median	Min	Max	
Random	5000	10.0	10	2	24	0.0023
Power-Law	5000	4.0	2	1	623	0.028
Small World-1	5000	10.0	10	3	25	0.002
Small World-2	5000	10.0	10	4	20	0.085
Small World-3	5000	10.0	10	6	18	0.23
Small World-4	5000	10.0	10	8	17	0.427

The message load in the system is measured in terms of the number of messages received by each peer in unit time. Cumulative message rate of a peer P_i is defined as the number of messages received by P_i in unit time. The cumulative message rate of the system is defined as the average of the cumulative message rates of all the peers. There are four types of messages in the CoQUOS system, namely query messages, announcement messages, notification messages and system maintenance messages. Accordingly, the cumulative message rate of the system is composed of four component $-$ query message rate, announcement message rate, notification message rate and maintenance message rates. These terms are defined similar to the cumulative message rate. Message load per query quantifies the message costs imposed by each continuous query and is calculated as the ratio of total number of query messages circulated in the system to total number of continuous queries issued.

6.2 Performance of the CRW Algorithm

In the first set of experiments, we exclusively study the performance of the CRW algorithm by comparing it with PRW and flooding. Since the goal is to study the performances of query propagation schemes, we use the fixed probability technique for query registrations for all experiments in this set.

In the first experiment, we measure the mean and the minimum NSRs of the PRW and CRW query propagation schemes when the query TTLs are set to various values. The registration probability value of the FP query registration technique is set to 0.25. Figure 4 and Figure 5 show the NSRs of the two schemes on power-law and random networks respectively. The results show that the CRW scheme provides significant benefits both in terms of mean and minimum NSRs. For example, the mean NSRs of the CRW algorithm are 15% to 68% higher than the corresponding values of the PRW scheme for the power-law network. Observe that the improvements provided by the CRW scheme on the minimum NSRs are even larger. CRW yields reasonable minimum NSRs even at relatively low initial TTL values. For the power-law network, the minimum NSR of the CRW scheme is around 50% when the query TTL is just 20, whereas the PRW scheme achieves a similar value only when the query TTL is set to 100.

TABLE 4
Performance of Query Dissemination Schemes at Various Message Loads

Per-Query Msg. Load	Flooding	PRW	CRW
4	22%	22%	31%
27	41%	72%	87%

The above experiment does not include the Gnutella-like flooding approach for query propagation because of two reasons. First, flooding scheme results in extremely high message traffic for the initial TTL value ranges of the experiment. More importantly, comparing random walk-based approaches (PRW and CRW) and flooding technique at the same settings of initial TTL values is not fair, since flooding-based approaches result in much higher traffic than random walks for the same initial TTL value. A fair comparison would be to measure the NSRs of these schemes at equal message loads. Accordingly, in our next experiment, we compare the NSRs of PRW, CRW and flooding schemes at equal values of per-query message load. In Table 4, we show the mean NSRs of the three schemes on power-law network at two values of per-query message load. When the initial TTLs of queries are set to 1 and 2, the per-query message loads are 4 and 27 respectively. The initial TTLs of PRW and CRW are set to 4 and 27 to obtain same values for per-query message loads. The registration probability is set to 0.25 for all schemes. The results show that the mean NSR of the flooding approach is below 50% even when the average per-query message load is around 27. This is because, in the flooding-based approaches, the messages remain in close vicinity of the source, and do not reach different regions of the network.

The third experiment studies the effect of the announcement TTL on the NSRs of PRW and CRW schemes. Figure 6 shows the mean NSRs of PRW and CRW schemes on the power-law and the random networks. The results show that the CRW scheme consistently performs better than the PRW scheme for both networks and at all announcement TTL values. The PRW scheme performs similar to CRW when the announcement TTL is set to 4. However, the message load in the system grows rapidly with each increment of

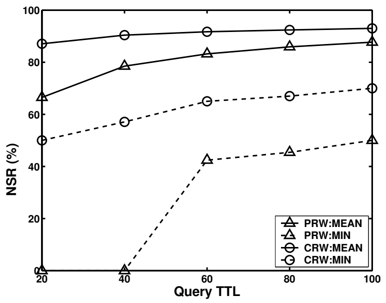


Fig. 4. Comparison of NSRs of PRW and CRW (Power-law Network)

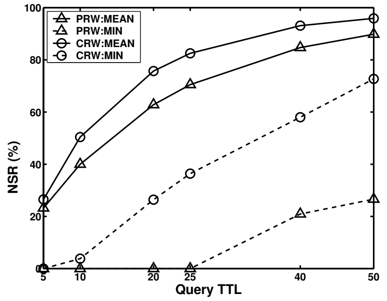


Fig. 5. Comparison of NSRs of PRW and CRW (Random Network)

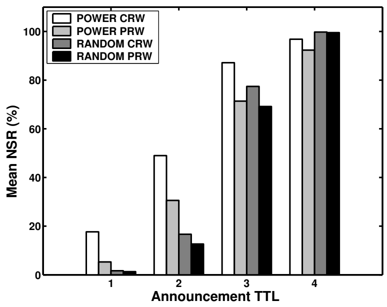


Fig. 6. Effects of Announcement TTL on NSRs

TABLE 5
Effect of λ on CRW Performance (Power-law network)

λ	0	1	2	3	4	5
Mean NSR (%))	66.9	79.1	84.2	85.0	85.3	85.5

the announcement TTL. Therefore, it is important to achieve high NSRs even at very small announcement TTL values. Our fourth experiment studies the effect of network size on the effectiveness of CRW and PRW. We consider two power-law networks – one with 5000 nodes and another with 10000 nodes. The networks are similar with respect to other characteristics. Figure 7 shows the improvements of CRW over PRW are higher for larger networks. For instance, at 30 beacon nodes, the improvements of CRW over PRW are 75% and 83% for 5000 and 10000 node networks respectively.

In the fifth experiment, we evaluate the impact of λ (bias factor) on the mean NSR. Table 5 shows the results on the power-law network when the query TTL, announcement TTL, and the registration probability were set to 20, 3, and 0.25 respectively. We observe that initially mean NSR increases considerably with increasing λ, but stabilizes when $\lambda = 4$.

The next experiment studies the effect of node clustering on the PRW and CRW schemes. Figure 8 shows the mean NSRs of the two schemes on small world graphs with various cluster coefficient values. The mean NSRs of both CRW and PRW schemes are high when the clustering coefficients are low. The performance of both schemes decline as the clustering coefficient increases. However, the NSR of the PRW scheme drops rather drastically whereas CRW provides reasonable NSRs even when the network exhibits strong cluster characteristics. In order to get a better understanding of the impact of network topology on the two schemes, our next experiment (see Figure 9) evaluates the mean NSRs of PRW and CRW on a set of 5000-node power-law overlays with varying α values. The values of query TTL, announcement TTL and registration probability were set to 20, 2, and 0.25 respectively. While CRW clearly outperforms PRW at lower α values, the improvements are less pronounced at higher α values. This is because at higher α values, the overlays have large numbers of single-degree peers. With number of nodes and edges held constant, this implies that overlays with higher α values have smaller diameters, and hence both CRW and PRW yield high mean NSRs.

6.3 Evaluating the Dynamic Probability Scheme

In this set of experiments, we evaluate the dynamic probability scheme for query registration by comparing it with the fixed probability scheme. As mentioned earlier, the query registration schemes have to be used in conjunction with a query propagation scheme. In our experiment we simulate the fixed probability and the dynamic probability schemes in conjunction with both the PRW and the CRW query propagation techniques to obtain four combinations, namely *pure random walk with fixed probability (PRW-FP)*, *pure random walk with dynamic probability (PRW-DP)*, *cluster resilient random walk with fixed probability (CRW-FP)* and *cluster resilient random walk with dynamic probability (CRW-DP)*. The CoQUOS system uses the CRW-DP query registration scheme.

We evaluated the mean NSRs of the four schemes as the average number of beacon nodes per query varies from 3 to 18. The combinations with dynamic probability scheme yields 2% to 10% higher NSRs than the respective fixed probability combinations for the power-law network. The improvements range between 1% and 9% for the random network (graphs of these experiments can be found in [25]).

Since the differences between the mean NSRs of the two query registration schemes are relatively low, it is natural to question the usefulness of the dynamic probability scheme. However, recall that the primary motivation of the dynamic probability scheme is to ensure reasonably high success rates for *all* queries. To evaluate whether the scheme achieves this goal, we plot the minimum NSRs of the four combinations in Figure 10 and Figure 11. The results show that the dynamic probability scheme provides considerably higher minimum NSRs than its fixed probability counterpart. For instance, the minimum NSR of the PRW-DP combination is around

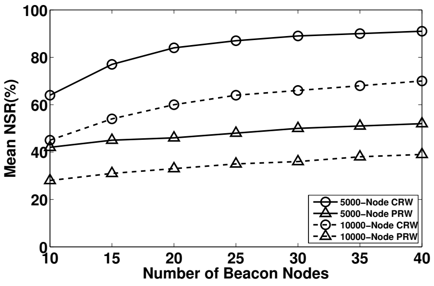


Fig. 7. Effectiveness of PRW and CRW on Varying Network Sizes

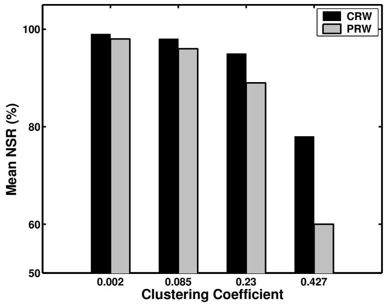


Fig. 8. Effects of Node Clustering on PRW and CRW

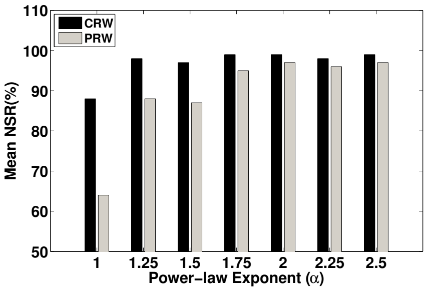


Fig. 9. Effects of Power-law Exponent on CRW and PRW

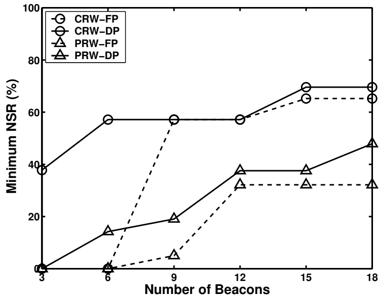


Fig. 10. Comparison of FP and DP schemes on minimum NSR (Power-law)

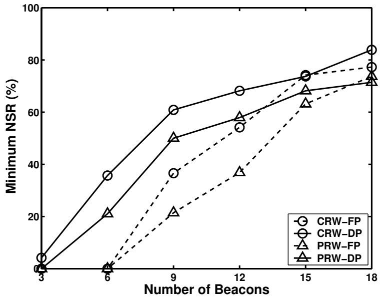


Fig. 11. Comparison of FP and DP schemes on minimum NSR (Random)

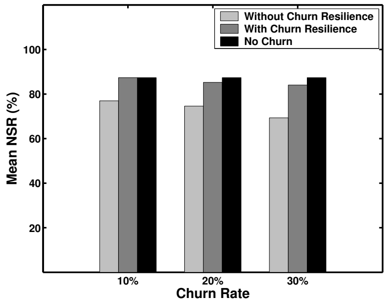


Fig. 12. Effectiveness of Churn Resilience Technique (Power-law)

57% higher than the PRW-FP combination for the power-law network when the number of beacon nodes is 9. The improvements in the NSRs are essentially due to better distribution of beacon nodes. It should however be noted that typically, there are a small number of queries ($< 5\%$) for which the fixed probability scheme yields significantly lower NSRs than the dynamic probability scheme. For most queries, the dynamic probability scheme yields similar or slightly better NSRs than the fixed probability scheme. Nevertheless, it is important to ensure reasonable NSRs for all queries, which the dynamic probability scheme achieves.

6.4 Effectiveness of Churn Resilience and Load Balancing Mechanisms

In this set of experiments, we evaluate the effectiveness of our replication-based churn resilience mechanism. The simulation setup is similar to previous experiments. However, the peers are no longer static; they can enter and exit the system at arbitrary points in time. We compare our churn resilience scheme to two other scenarios, namely a CoQUOS system with no node dynamics and a dynamic CoQUOS network but with no failure resilience. In the third scenario, when a peer exits the network, all the queries registered at the peer become unavailable. We use the CRW-DP combination for all three scenarios. The average beacon count is 6 and 4 for the power-law and the random networks respectively. For our replication-based churn resilience scheme, the replication factor r_f is set to 1. We measure the mean NSR when the churn rate is set to 10%, 20%, and 30%. Churn rate indicates the percentage of nodes entering and exiting the network per unit time.

The bar graph in Figure 12 and Figure 13 show the results of the experiments. When the network becomes dynamic, the NSRs experience a significant drop even at low churn rates. And even at a replication factor of 1, the passive query replication scheme minimizes the negative impact of churn to a considerable extent. However, there is still a small drop in the NSR when compared with an identical network with no node dynamics. This can be attributed to two reasons $-$ simultaneous failures of a beacon node and its replica and failure of a query's beacon node before the query is replicated. As a side note, flooding-based query propagation is not as sensitive to network churn as PRW or CRW. For example, on the power-law network, with per-query message load set to 27 and the beacon count set to 7, the mean NSR of flooding-based approach falls from 41% to 38% at churn rate of 20%, whereas CRW experiences a drop from 87% to 74%. This is because, with flooding-based approach the beacon nodes of a query are located in close proximity. However, having closely located beacon nodes is also a key shortcoming of the flooding-based approach for the current application. Controlled flooding and multiple random walks (through the message fission technique [11]) may provide the benefits of both flooding and CRW while minimizing their limitations. However, detailed studies are needed to establish their properties.

Next, we evaluate the effectiveness of our active load

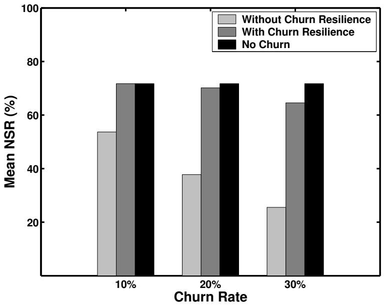


Fig. 13. Effectiveness of Churn Resilience Technique (Random)

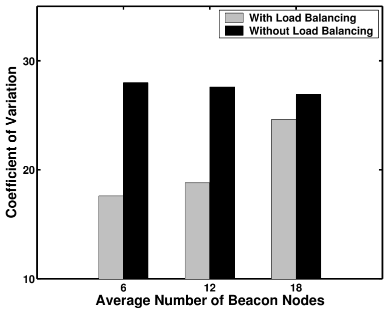


Fig. 14. Effectiveness of Load Balancing Technique (Power-law)

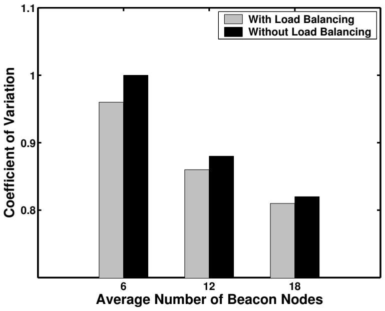


Fig. 15. Effectiveness of Load Balancing Technique (Random)

balancing scheme by comparing the load imbalances in the CoQUOS system when the load balancing scheme is activated and de-activated. We use the coefficient of variation metric on the notification loads of the peers for quantifying the load imbalance in the system. The coefficient of variation is a normalized dispersion measure, and it is defined as the ratio of the standard deviation to the mean. The lower the coefficient of variation the better the load distribution. The experiments were carried out on both the power-law network and the random network, and the coefficient of variation of notification load is evaluated at three distinct settings of average beacon count. As Figure 14 and Figure 15 show, the active load balancing scheme significantly reduces load imbalance in both power-law and random networks.

6.5 Messaging Costs of the CoQUOS System

In the final set of experiments, we study the communication costs of the CoQUOS system. We evaluate each of the four components of the cumulative message rate, namely query message rate, announcement message rate, notification message rate and system maintenance message rate. The CoQUOS system is executed on the power-law and the random networks with all its capabilities enabled. The initial TTL of the continuous queries and the announcements are set to 50 and 2 respectively for both networks. The initial probability and the probability increments are both set to 0.02. The query rates and the announcement rates of all peers are set to 0.05 per unit time and the peer churn rate is 10%. The synchronization duration for both query replicas and connectivity information is set to 10 time units.

Table 6 shows the various components of the cumulative message rate for power-law and random networks respectively. The results indicate that the announcement message rate forms the predominant component of the cumulative message rate, and that the initial announcement TTL has considerable impact on the message load in the CoQUOS system. This is because, the CoQUOS system utilizes limited broadcast for announcement circulation. Advanced strategies for announcement circulation (such as directed BFS [34]) can significantly reduce the overall message load in the system. The fact that the system maintenance message rate forms a very small fraction of the cumulative message rate demonstrates that CoQUOS can be efficiently implemented.

One issue that arises when deploying the CoQUOS system is: how to set the values for various configuration parameters? Clearly, the parameter settings have to be adjusted depending upon the overlay characteristics and run-time behaviors of the peers. Nevertheless, a few rules of the thumb are useful. First, for simplicity, the initial probability and the probability increment parameters can be set to the same value. Second, there is a tradeoff between the announcement TTL and the probability increment settings. If the announcement TTL is small, the beacon nodes of a query need to be closer, which implies that probability increment should be set to higher values. For example, probability increment can be set to $\frac{C}{Announcement\ TTL}$, where C is a constant. An interesting observation is that higher values of probability increment results in larger processing loads on individual peers due to increased query registrations, whereas higher announcement TTLs impose larger messaging overheads on the overlay. The λ parameter should be set in relation to the cluster coefficient of the overlay. A value between 3 and 5 works well for most overlays.

7 Related Work

In the past few years, both structured and unstructured P2P networks have experienced significant research [1], [2], [4], [10], [24], [26], [27], [30]. Searching through ad-hoc queries has been the predominant information discovery mechanism in P2P networks. Several researchers have studied efficient and scalable alternatives to the flooding-based searching in unstructured P2P networks [10], [21], [34]. Random walk and its variants have been explored as alternatives to the broadcast strategy [10], [18], [19], [23]. However, the ad-hoc query paradigm is inadequate for advanced P2P content sharing applications.

A second research area that is closely related to work presented in this paper is that of pub-sub systems (event-delivery systems) [7], [9], [14], [15], [28], [31], [3], [32]. The early pub-sub systems adopted a centralized architecture for subscription maintenance and matching

TABLE 6
Communication Costs of the CoQUOS system

	Query Msg. Rate	Advt. Msg. Rate	Ntfy. Msg. Rate	Mtn. Msg. Rate	Cuml. Msg. Rate
Power-law N/W	2.5	11.92	0.26	0.38	15.06
Random N/W	2.5	6.01	0.38	0.96	9.85

and notification of published data items, whereas the latter systems were either partially [7], [9] or completely decentralized [3]. More recently, researchers have designed decentralized pub-sub systems on top of P2P networks [12], [20], [31], [32]. Several pub-sub systems have been successfully implemented on structured P2P platforms [6], [20], [31], [35].

Unstructured P2P networks-based pub-sub systems are relatively less explored. The main difficulty in designing pub-sub systems on unstructured P2P networks emerges from the fact that the topologies of most unstructured P2P networks have no relationship to the data (or other information) in individual peers. As a result, routing in these networks is, in general, independent of peers' contents (a few unstructured P2P networks use content-sensitive heuristics for routing). Most unstructured P2P networks-based pub-sub systems try to overcome these difficulties by carefully controlling the manner in which the topology of the overlay network evolves, and by adopting intricate indexing strategies for routing subscriptions and notifications [12], [32]. Unfortunately, these mechanisms are highly complex and they introduce significant overlay management overheads thereby limiting the scalability of the corresponding systems. As mentioned in the introduction, the Sub-2-Sub [32] requires peers to be clustered on the basis of their subscriptions. The publisher of a data item is required to reach the cluster that exactly corresponds to the data item being published, and then start the dissemination process. The RDF-based pub-sub scheme designed in the context of ELENA project [13] also suffers from similar limitations. This system also requires the peers of the overlay to be meticulously organized, in this case, to form an intricate bi-level Hypercup architecture. Besides having limited scalability, the system is very vulnerable to super peer failures.

The CoQUOS system differs fundamentally from P2P-based pub-sub schemes. All of the above-mentioned systems are essentially pub-sub systems which are implemented on a P2P platform. In contrast, the goal of our work is to enhance the content discovery capabilities of unstructured P2P content distribution networks. With the goal of achieving guaranteed notification, most unstructured P2P-based pub-sub systems include complex overlay management strategies. On the other hand, our CoQUOS system does not impose any restrictions on the topology of the overlay, employs lightweight techniques, yet it provides high success rates. However, CoQUOS might not be appropriate for applications where guaranteed notification is necessary.

SmartSeer [22] is a structured P2P overlay-based continuous query system for document repositories. Unlike SmartSeer, CoQUOS does not rely upon DHT for mapping continuous queries to peer nodes hosting them or for routing notifications. Instead, it incorporates lightweight mechanisms like CRW and dynamic probability scheme for registering continuous queries on various peer nodes of the overlay. PeerCQ [17] is a P2P-based continual query system for information change monitoring. It too uses the DHT to distribute change monitoring queries to the nodes of the network.

Researchers have studied the properties of random walk as well as its utility for various P2P applications such as computing aggregate queries [5] and uniform sampling of peers in unstructured P2P networks [18]. Several variants of random walk such as popularity-biased random walk have been proposed for P2P searching [36]. Gkantsidis et al. [18] show that the performance of random walk degrades if the overlay network exhibits considerable degree of node clustering. CRW overcomes this limitation by favoring *out-of-cluster* peers at each hop of message propagation.

8 CONCLUSION

Mechanisms that enable individual peers of unstructured P2P content sharing networks to register long-standing queries and receive notification when new matching items appear can significantly improve their utility and effectiveness. While the pub-sub paradigm can provide this capability, implementing pub-sub systems on unstructured overlays is often a very complex endeavor. The continuous query paradigm studied in this paper is similar to pub-sub, but it provides best-effort notification service. We presented the design and evaluation of a lightweight system, called CoQUOS, which supports continuous queries in unstructured P2P networks. The CoQUOS system incorporates several novel features such as cluster resilient random walk for query propagation, dynamic probability scheme for query registration, and a lazy replication technique for countering network churn. The proposed techniques have been evaluated through theoretical analysis and simulation-based experiments.

ACKNOWLEDGMENTS

This work is partially supported by National Science Foundation under Grant No. 0716357. An extended abstract of this paper appeared in the Proceedings of International Parallel and Distributed Processing Symposium 2007.

References

[1] Gnutella P2P Network. www.gnutella.com.

[2] Kazaa P2P Network. www.kazaa.com.

[3] TIB/Rendezvous. White paper, 1999.

[4] S. Androutsellis-Theotokis and D. Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Comput. Surv.*, 2004.

[5] B. Arai, G. Das, D. Gunopulos, and V. Kalogeraki. Approximating Aggregation Queries in Peer-to-Peer Networks. In *Proceedings of the 22^{nd} International Conference on Data Engineering (ICDE)*, 2006.

[6] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg. Content-based Publish-Subscribe over Structured Overlay Networks. In *Proceedings of ICDCS* , 2005.

[7] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. In *Proceedings of ICDCS 1999*, 1999.

[8] T. Bu and D. F. Towsley. On Distinguishing between Internet Power Law Topology Generators. In *INFOCOM*, 2002.

[9] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, 2001.

[10] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P Systems Scalable. In *Proceedings of ACM SIGCOMM 2003*, 2003.

[11] J. Chen, L. Ramaswamy, and A. Meka. Message Diffusion in Unstructured Overlay Networks. In *Proceedings of NCA*, 2007.

[12] P. Chirita, S. Idreos, M. Koubarakis, and W. Nejdl. Publish/Subscribe for RDF-based P2P Networks. In *Proceedings of the 1st European Semantic Web Symposium*, May 2004.

[13] P. A. Chirita, S. Idreos, M. Koubarakis, and W. Nejdl. Designing Semantic Pub-lish/Subscribe Networks using Super-Peers. *Semantic Web and Peer-to-Peer.Springer Verlag*, 2005.

[14] P. T. Eugster, R. Guerraoui, and C. H. Damm. On Objects and Events. In *Proceedings of OOPSLA*, 2001.

[15] P. T. P. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, 35(2), 2003.

[16] P. Ganesan, M. Bawa, and H. Garcia-Molina. Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems. In *Proceedings of VLDB*, 2004.

[17] B. Gedik and L. Liu. A Scalable Peer-to-Peer Architecture for Distributed Information Monitoring Applications. *IEEE Transaction on Computers*, 54(6), 2005.

[18] C. Gkantsidis, M. Mihail, and A. Saberi. Random Walks in Peer-to-Peer Networks. In *Proceedings of the IEEE INFOCOM 2004*, 2004.

[19] C. Gkantsidis, M. Mihail, and A. Saberi. Hybrid search schemes for unstructured peer-to-peer networks. In *INFOCOM*, 2005.

[20] A. Gupta, O. D. Sahin, D. Agrawal, and A. E. Abbadi. Meghdoot: content-based publish/subscribe over P2P networks. In *Middleware 2004*, 2004.

[21] V. Kalogeraki, D. Gunopoulos, and D. Zeinalipour-Yazti. A Local Search Mechanism for Peer-to-Peer Networks. In *Proceedings of CIKM*, 2002.

[22] J. Kannan, B. Yang, S. Shenker, P. Sharma, S. Banerjee, S. Basu, and S.-J. Lee. SmartSeer: Using a DHT to Process Continuous Queries Over Peer-to-Peer Networks. In *Proceedings of IEEE INFOCOM*, 2006.

[23] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Supercomputing*, 2002.

[24] W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. PeerDB: A P2P-based System for Distributed Data Sharing. In *Proceedings of ICDE*, 2003.

[25] L. Ramaswamy, J. Chen, and P. Parate. CoQUOS: Lightweight Support for Continuous Queries in Unstructured Overlays. In *Proceedings of IPDPS*, 2007.

[26] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-Addressable Network. In *Proceedings of ACM SIGCOMM 2001*, Aug 2001.

[27] P. Reynolds and A.Vahdat. Efficient peer-to-peer keyword searching. In *Middleware 2003*.

[28] B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps. Content Based Routing with Elvin4. In *Proceedings of AUUG2k*, 2000.

[29] K. Sripanidkulchai. The popularity of gnutella queries and its implications on scalability, Feb 2001. Featured on O'Reilly's www.openp2p.com website.

[30] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001*, Aug 2001.

[31] P. Triantafillou and I. Aekaterinidis. Content-based Publish/Subscribe Over Structured P2P Networks. In *Proceedings of DEBS*, 2004.

[32] S. Voulgaris, E. Riviere, A.-M. Kermarrec, and M. van Steen. Sub-2-Sub: Self-Organizing Content-Based Publish Subscribe for Dynamic Large Scale Collaborative Networks. In *Proceedings of the 5th international workshop on peer-to-peer systems*, Feb 2006.

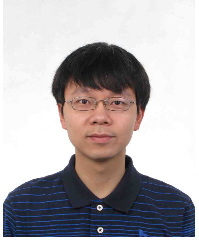
[33] D. J. Watts and S. H.Strogatz. Collective Dynamics of 'Small-World' Networks. *Nature*, 393(4), 1998.

[34] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer systems. In *ICDCS 2002*.

[35] X. Yang, Y. Zhu, and Y. Hu. Scalable Content-Based Publish/Subscribe Services over Structured Peer-to-Peer Networks. In *Proceedings of PDP*, 2007.

[36] M. Zhong and K. Shen. Popularity-Biased Random Walks for Peer-to-Peer Search under the Square-Root Principle. In *IPTPS*, 2006.

Lakshmish Ramaswamy received the PhD degree in Computer Science from Georgia Tech in 2005. He is currently an assistant professor in the computer science department at the University of Georgia. His research interests are broadly in the area of distributed systems, and more specifically in performance, scalability, security and privacy of Internet services, overlay networks, events-based middleware, and mobile systems. He is the recipient of best paper award of the 13th World Wide Web conference (WWW-2004) and the 2005 Pat Goldberg best paper award. He has served on the program committees of several international conferences and workshops. He was the program co-chair of DEPSA-2007 and SENS-2006 workshops. He is a member of IEEE and the IEEE computer society.



Jianxia Chen received the bachelor's and master's degrees in Computer Science from the Xiamen University, Xiamen, China, in 2002 and 2005. He is currently a Ph.D. student in the Department of Computer Science at the University of Georgia. His research interests are in distributed computing systems, including continuous queries, publish-subscribe systems, distributed event based systems. He is a student member of the IEEE.