Popularity Adaptive Search in Hybrid P2P Systems

Xiaoqiu Shi[†], Jinsong Han[‡], Yunhao Liu[‡], and Lionel M. Ni[‡]

[†]Wenzhou University
Dept. of Computer Science
Wenzhou, Zhejiang, 325035 China
sxq@wzu.edu.cn

[‡]Hong Kong University of Science and Technology
Dept. of Computer Science and Engineering
Clear Water Bay, Kowloon, Hong Kong
{jasonhan, liu, ni}@cse.ust.hk

Abstract

In a hybrid peer-to-peer (P2P) system, flooding and DHT are both employed for content locating. The decision to use flooding or DHT largely depends on the population of desired data. Previous works either use local information only, or do not consider dynamic factors of P2P systems. In this paper, we propose a Popularity Adaptive Search method for Hybrid (PASH) P2P systems. By dynamically detecting the content popularity, PASH properly selects search methods and efficiently saves query traffic cost and response time. We comprehensively evaluate PASH through synthetic and trace-driven simulations. The results show that PASH outperforms existing approaches and it also scales well.

1. Introduction

Since the emergence of Napster [4] in 1999, the Peer-to-Peer (P2P) model has been increasingly popular along with deployment in file sharing, distributed directory service, web cache, storage and grid computing [8, 9, 14, 21-24, 26]. Many P2P systems report huge numbers of simultaneously active participants, with millions, if not billions, of participating machines. Those participants include home PCs as well as enterprise computers. They are called peers or servants, each acting as both a server and/or a client.

P2Ps used to be classified as centralized, decentralized structured, and decentralized unstructured [12] models. Centralized model, such as Napster, has a constantly-updated directory hosted at central locations. The main drawback of centralized models is the vulnerability to single point failure. Decentralized unstructured P2P systems, such as Gnutella [1] and KaZaA [3], eliminate centralized directory and employ flooding-based search techniques. Although flooding-based search is effective for locating popular items, it often performs poorly in search latency and result quality when queries are focused on rare items. This is because that flooding-based search sometimes fails to return matches of desired data due to the partial coverage problem of flooding scope. As an alternative, decentralized structured models, such as Chord [19], CAN [15], Pastry [16], and Tapestry [7], are proposed . In those systems, P2P topology is highly organized and files are placed at specified locations based on Distributed Hash Tables(DHT), which make the queries for rare items easier to answer. Figure 1 makes a comparison between the flooding-based and DHT-based search.

To improve search quality and efficiency for both popular and rare items, hybrid P2P approaches are introduced [10, 11]. By combining the unstructured P2Ps with a structured DHT-based global index, queries are handled in a hybrid manner: popular items are found via flooding, while rare items are found via DHT-based algorithm. The major challenge in Hybrid P2Ps is how to identify rare items in a decentralized and self-organized environment.

Many efforts have been made for hybrid search, such as the SimpleHybrid [10, 11] and GAB [25]. However, they do not successfully deal with the dynamic nature of P2P networks where peers frequently join and leave the systems. Aiming at improving the hybrid search in dynamic environments, we propose a Popularity Adaptive Search for Hybrid P2P (PASH). PASH employs a network dynamic sensing infrastructure and uses the results to guide/adjust the search decision-making. We evaluate PASH through synthetic and trace-driven simulations and contrast it with a most recent work GAB [25]. The results show that PASH outperforms existing approaches in terms of both query traffic cost and search response time.

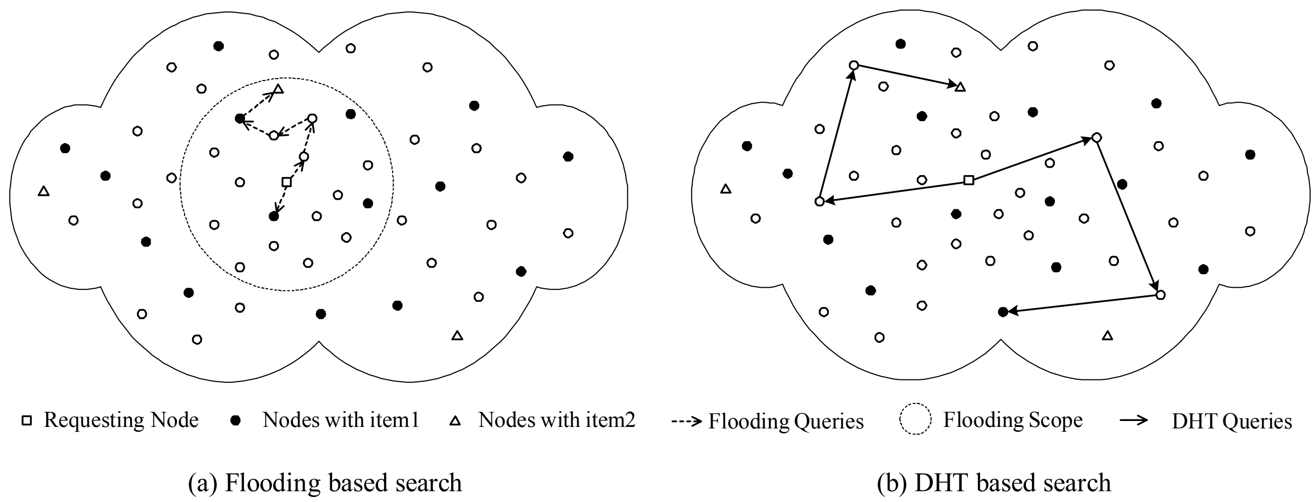


Figure 1. Comparison for Flooding with DHT-based search. Flooding is effective in searching popular items (only 1 hop needed to find item1), but performs poorly for rare items (5 hops needed to find item2). In contrast, DHT is efficient to find the rare items, while the maintenance of DHT is costly.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 discusses the major challenges in current hybrid P2Ps. Section 4 proposes our PASH approach. We describe the simulation methodology in Section 5 and present performance evaluation on PASH in Section 6. We conclude this work in Section 7.

2. Related Work

A spectrum of approaches have been proposed to improve search quality and efficiency in decentralized P2P systems, such as DHT-based, flooding-based, caching-based, topology optimization and hybrid search.

DHT-based search techniques [7, 15, 16, 19] have been widely studied. An overview of DHT-based search approaches can be found in [5]. In such a scheme, items are inserted into a distributed hash table and found by specifying their unique keys. To efficiently implement a DHT, the DHT-based search algorithm must be able to determine which node is responsible for storing the item associated with the given key and then map the keys to nodes in a load-balanced manner. Meanwhile, each node needs to maintain a routing table based on the DHT overlay to forward messages and lookup items.

Flooding-based approaches are widely deployed in practical decentralized and unstructured P2P systems such as Gnutella. In those systems, peers are interconnected in an ad-hoc pattern, and queries are flooded in P2P overlays. To make Gnutella-like networks scalable, a random walk based search algorithm [12] is introduced. GIA [6] modifies the k-walker algorithms and includes flow control, dynamic topology adaptation, and one-hop replication to handle the problem of nodes' heterogeneity. In [27], authors proposed a popularity biased algorithm to enhance the efficiency of random walk based search. In each step of queries' random walks, the next destination is determined based on the content popularities of current peer's neighbors.

Since Gnutella-like P2P is effective for locating highly replicated items but reluctant to find rare ones, a hybrid search infrastructure, SimpleHybrid, is proposed. SimpleHybrid combines Gnutella and the DHT-based PIERSearch [10, 11]. To improve the search efficiency, GAB [25] makes search decisions based on global statistics of documents' popularity (In this paper, we will use the terms of "item" and "document" interchangeably). GAB outperforms SimpleHybrid by using a gossip algorithm to make popularity-biased search possible. However, global statistics of documents' popularity obtained from gossip in GAB do not well reflect the dynamic characteristic of peers' frequently joining and leaving. This leads to unnecessary flooding or inappropriate document publishing in DHT.

3. Challenging Issues in Popularity Based Search

In hybrid P2P systems, different heuristics have been used to identify which items are rare or popular, such as search frequency of keywords, keyword pair frequency, searching result size history, sampling of neighboring nodes, and historical statistics on item replicas [25]. For example, in [10, 11], a query is flooded with a limited depth first, and, if no result is returned, the query goes to the DHT. Such a design is simple and effective but inefficient in that it incurs extra overhead. It also increases the response time when searching rare documents and wastes the bandwidth due to the unfruitful flooding.

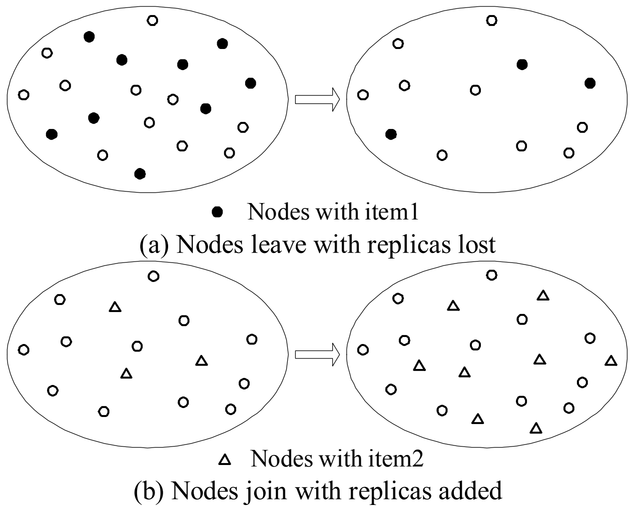

● Nodes with item1
(a) Nodes leave with replicas lost
△ Nodes with item2
(b) Nodes join with replicas added

Figure 2. Without considering the dynamic feature, gossip-obtained popularity values are not accurate in P2P environments.

The most recent work is GAB [25]. In the design of GAB, when a node receives a document title it has not indexed before, it tosses a coin up to k times and counts the number of heads before the first tail appears. It stores this value of this title and then exchanges it with other nodes by gossip to compute a maximum value. As a result, the maximum value of this title can be represented the popularity of this document. Thus, a more widely replicated document would have a larger popularity value.

Each node maintains a histogram of the popularity values of the documents it holds and a flooding threshold. When a peer would like to issue a query with a set of keywords, it first seeks the popularity values for that set of keywords and then compares them to the flooding threshold to determine the search selection.

Our observations show that GAB is effective when peers are stable and active in a long run, for example, acting as the ultrapeers. In most P2P systems, however, not all "normal" hosts have as long expected uptimes as ultrapeers and they leave or join the P2P network randomly. The study in [17] shows that in Gnutella about half of the peers in the system are replaced by new comers within one hour. Under a dynamic environment, as shown in Fig. 2, popularity computed from GAB does not well reflect the snapshot of resources' replication in P2P networks.

Our analysis below will show that the statistic of item replicas in GAB is always a monotone increasing function of the gossip rounds.

Let θ denote a round of gossip, $v_i(\theta)$ is replicas statistics of item i in gossip round θ and v_{ij} is the popularity value of item i from node j. In GAB, for all $v_i(\theta)$ and θ, we have Eqs. (1) and (2).

$$\mathrm{v}_i(\theta) = v_i(\theta-1)\forall v_{ij} = \begin{cases} v_i(\theta-1) & if \quad v_i(\theta-1) > v_{ij} \\ v_{ij} & if \quad v_i(\theta-1) < v_{ij} \end{cases} \tag{1}$$

$$\Delta\mathrm{v}_i = v_i(\theta) - v_i(\theta-1) = \begin{cases} 0 & if \quad v_i(\theta-1) > v_{ij} \\ >0 & if \quad v_i(\theta-1) < v_{ij} \end{cases} \tag{2}$$

Obviously, the statistics of replicas in GAB will remain unchanged or increase even though there is a decrease in item replicas because of node departure. Indeed, under the dynamic environment, GAB statistics are far from accurate.

Let η denote the ratio of the real popularity value of an item to the popularity value obtained from gossip. If the latter value is less than the former one, $\eta > 1$; otherwise, $\eta < 1$.

If the flooding threshold is set at T_f, when $\eta > 1$, queries for the items whose real popularity values are within the interval $[T_f, \eta T_f]$ would be incorrectly published to DHT, because their gossip-obtained popularity values are among $[T_f/\eta, T_f]$. On the other hand, when $\eta < 1$, queries for the items whose real popularity values are within the interval $[\eta T_f, T_f]$ would be incorrectly flooded since their gossip-obtained popularity values are within the interval $[T_f, T_f/\eta]$.

Both situations would result in a decline of the search performance described by Eqs. (3) and (4). Figure 3 plots the integrated metric, utility function $U(p_i)$, where p_i is the popularity value for item i, proposed by the authors of GAB [25] and shows the performance decline when $\eta > 1$ and $\eta < 1$.

$$\Delta U_1 = \sum_{p_i=T_f}^{\eta T_f}\left[U_{DHT}(p_i) - U_{flood}(p_i)\right] < 0, \quad \eta > 1 \tag{3}$$

$$\Delta U_2 = \sum_{p_i=\eta T_f}^{T_f}\left[U_{flood}(p_i) - U_{DHT}(p_i)\right] < 0, \quad \eta < 1 \tag{4}$$

Motivated to achieve a flexible hybrid P2P search mechanism that can work well in dynamic environment, we propose PASH, a hybrid search method which adaptively adopt the dynamic of items' popularities, to improve the search accuracy and efficiency.

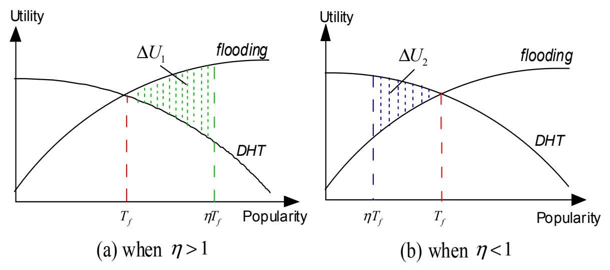


Figure 3. Utility decline.

4. PASH Design

In PASH, the popularity value of an item is computed based on historical statistics on item replicas as well as their dynamics information. While remaining the GAB's gossip algorithm to collect the global statistics of item replicas, PASH designates "smart nodes" acting as sensors to detect the network dynamic. PASH also uses the sensing results to adjust the gossip-obtained popularity values to reflect the items' popularities more accurately. This is shown in Fig. 4.

Now we define the systems parameters of PASH. Given that there are R replications for all the items, and R_i replications for item i in a P2P system, such that $\Sigma R_i=R$, we define

(1) β_i, the dynamic rate of the item i's replicas per time unit in Eq. (5),

$$\beta_i = \frac{1}{R_i}\frac{dR_i}{dt} \tag{5}$$

(2) β, the dynamic rate of the replicas of all items per time unit in Eq. (6).

$$\beta = \frac{1}{R}\frac{dR}{dt} \tag{6}$$

In general, nodes randomly join and leave the P2P systems. Accordingly, the dynamic of those nodes would have even influences on replicas dynamics of different items. Suppose in the time interval dt, there is a replicas change dR of all items, and a replicas change dR_i of item i, we have Eqs. (7) and (8).

$$dR_i = \frac{R_i}{R}dR \tag{7}$$

$$\frac{1}{R_i}\frac{dR_i}{dt} = \frac{1}{R}\frac{dR}{dt} \tag{8}$$

Equation (8) means that in decentralized P2P, we can use the value of β to represent β_i. Let C denote the number of shared files a node can provide. We also suppose that

- at time t, there are n nodes in the system and among them there are λ_1 node with C_1, λ_2 node with C_2, $\cdots$, λ_j node with C_j, $\cdots$, λ_u node with C_u, respectively, where $\Sigma\lambda_j = n$ and $\Sigma\lambda_j C_j = R$.
- in the time period Δt, there are $\Delta\lambda_1$ nodes with C_1, $\Delta\lambda_2$ nodes with C_2, $\cdots$, $\Delta\lambda_j$ nodes with C_j, $\cdots$, $\Delta\lambda_u$ nodes with C_u leaving the system, and ξ_1 nodes with C_1, ξ_2 nodes with C_2, $\cdots$, ξ_i nodes with C_i, $\cdots$, ξ_v nodes with C_v joining the system. The distribution of the rate of nodes' joining or departure is statistically homogeneous.
- at time $t + \Delta t$, there are n' nodes in the system, where $n' = n - \Sigma\Delta\lambda_j + \Sigma\xi_i$

Thus, we derive Eq. (9) from above assumptions.

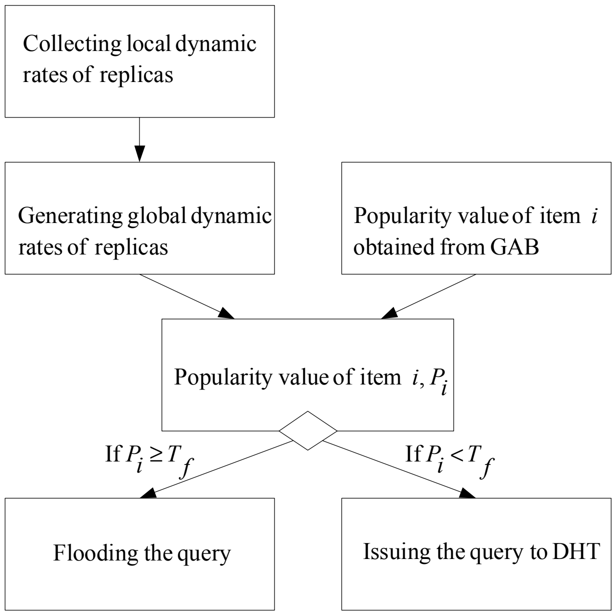

Collecting local dynamic rates of replicas
Generating global dynamic rates of replicas
Popularity value of item i obtained from GAB
Popularity value of item i, P_i
If $P_i \ge T_f$
If $P_i < T_f$
Flooding the query
Issuing the query to DHT

Figure 4. Flooding/DHT determination in PASH.

$$\beta = \frac{\Delta R}{R\Delta t} = \frac{k\sum_{i=1}^{v} C_i\xi_i - \sum_{j=1}^{u} C_j\Delta\lambda_j}{(\sum_{j=1}^{u} C_j\lambda_j)\Delta t}$$

$$= \frac{1}{R\Delta t}(k\sum_{i=1}^{v} C_i\xi_i - \sum_{j=1}^{u} C_j\Delta\lambda_j),\quad 0 \le k \le 1 \tag{9}$$

If the file set that the newly coming nodes bring has no overlap with the current file set, k = 0. If the file set brought in is a subset of the current file set, k = 1. Normally, k would be a value between zero and one according to the overlap ratio.

Suppose that the statistics value of item i's replicas, $v_i(\theta)$, is obtained through a gossip message at time $t + \Delta t_0$. At time $t + \Delta t$ ($\Delta t_0 < \Delta t$), we hope PASH is able to adjust it with a parameter γ to better reflect the real item replicas in the system. Then we have:

$$\mathrm{v_i}(\theta) \leftarrow \mathrm{v_i}(\theta)\cdot\gamma \quad , \quad where \quad \gamma = 1 + \beta(\Delta t - \Delta t_0)$$

We also need to compute η, the ratio of the real popularity value to the gossip-obtained popularity value. Thus, $\eta = \gamma \times n / n'$. It is difficult, if not impossible, to measure the accurate η or γ in a real P2P environment. Therefore, our later discussion will focus on how to estimate them and use them to help the search selection.

4.1. Node type and message type

In PASH, all nodes are divided into three groups: *smart nodes*, *backup smart nodes*, and *lazy nodes*. Smart nodes act as sensors to detect the dynamic of item popularity in

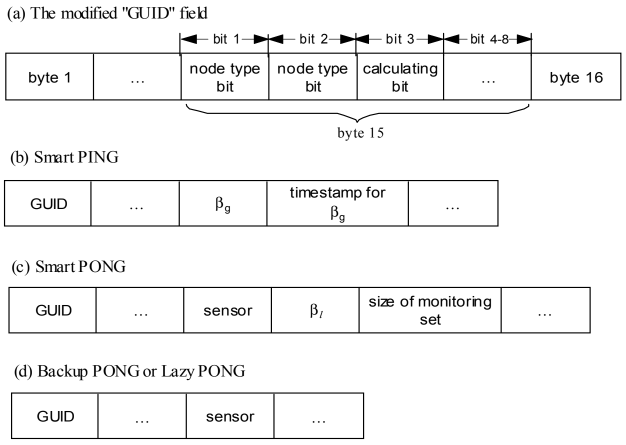


Figure 5. PING/PONG messages in PASH.

the system. Each of them maintains a host list, SmartNodes List, and a list of the monitored nodes as well as the global replica dynamic rate β_g and the local replica dynamic rate β_l. Backup smart nodes are candidates for smart nodes, maintaining a host list and β_g as well. The others are lazy nodes. Each node whether it is a smart, backup smart or lazy one maintains the information of the smart node it belongs to.

In the Gnutella 0.6 [2], byte 15 of the GUID field is reserved. We utilize the first two bits of this byte to identify the node's type, where "00" means lazy, "01" means backup smart, and "10" or "11" means smart. Furthermore, "10" means a smart node is "idle" and "11" means a smart node is "busy". When a smart node is capable of monitoring extra nodes, it marks itself as "idle", otherwise as "busy". Each node marks its neighbors as "smart", "backup smart", or "lazy". For a smart node, the third bit of byte 15 in GUID field is used as a "calculating bit". The default value of this bit is set to "0".

To reduce the traffic cost, we further modified the PING/PONG messages of Gnutella 0.6 protocol to piggyback the dynamics information of the system. We present the modified PING/PONG messages in Fig. 5. All types of PONG message have a "sensor" field to indicate to which smart node the responding node belongs.

One key issue here is how many smart nodes we need to deploy in the P2P system. An obvious tradeoff is that too many smart nodes incur unnecessary overhead and too few might lead to inaccuracy. We need to consider the size of the P2P system, computational capability of each node, sensing coverage, expected sensing accuracy, and acceptable traffic overhead, etc. In PASH, we select a proper ratio of the number of smart nodes to the number of all nodes through experiments, and we have more discussions on this issue in Section 5.

Each smart node maintains a SmartNodes List. It includes a number of other smart nodes. When initializing, a smart node creates its SmartNodes List from its host list. All smart nodes construct a mesh, as illustrated in Fig. 6. A smart node would utilize the information of "sensor" field in the received PONG messages. For example, a smart node s receives a PONG message, in which the "sensor" field has indicated a node u is a smart node but not included in s's local SmartNodes List. The node s will try to contact node u. If node u is active, node s includes u into its SmartNodes List. In this way, smart nodes can expand their SmartNodes Lists. As an option, SmartNodes List could be implemented as a subset of the host list in order to reduce the storage cost.

4.2. Smart node selection

Suppose the P2P system has n nodes and x smart nodes. PASH hereby sets the probability of a node becoming a smart node to x/n. Each smart node would monitor approximately n/x nodes and include them into its monitoring list.

When a fresh node joins the system, it obtains a host list from the bootstrapping node. It then calculates a probability $p = S \times (x/n)$, where S is the number of its neighboring host. If $p < 1$, it acts as a lazy node. If $p > 1$, and the number of smart nodes in its host list has exceeded p, the node would become a backup smart node. Otherwise, if the number of smart nodes in its host list is less than p, the node becomes a smart node.

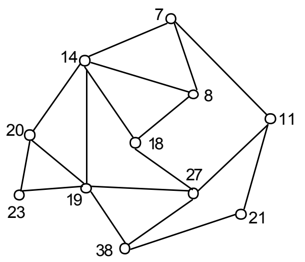
When a smart node leaves the system normally, it selects a backup smart node from its host list as its successor. The smart node then transfers the list of the monitored nodes to its successor. In addition, it informs its leaving to all other backup smart nodes in its host list with a "busy" Smart PING message. When a smart node is overloaded, it balances its work by delivering a part of its list of the monitored nodes to a backup smart node in its host list. Each leaving or overloaded smart node sends a "busy" Smart PING message to the monitored nodes to inform the handover. Meanwhile, the chosen smart node sends the "idle" Smart PING messages to those monitored nodes to confirm this replacement.

When a node joins the system, whatever role it chooses, say as a lazy, backup smart, or smart node, it selects one and only one "idle" smart node as its sensor node. During its lifetime, the node would not change its sensor node except: 1) the smart node hands over the duty to another smart node; or 2) the sensor is abnormally offline. In the second case, the node would check its host list to choose another "idle" smart node as its sensor.

4.3. Popularity dynamics sensing

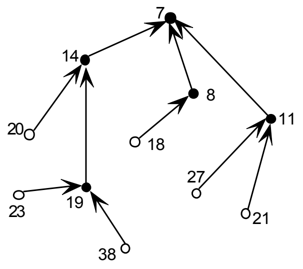
PASH requires each smart node detect the changes of its monitored nodes and share this information with other smart nodes.

Smart Node	SmartNodeList
7	14,8,11
14	7,8,18,19,20
11	7,21,27
19	14,20,23,38,27
23	19,20,27
38	19,27,21
27	18,19,11,38
20	14,19,23
21	38,11
18	14,8,27
8	7,14,18



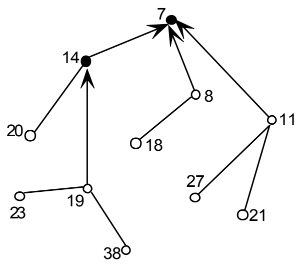
(a) Initial mesh topology among the Smart Nodes

Smart Node	SmartNodeList
7	14,8,11
14	**7**,8,18,19,20
11	**7**,21,27
19	**14**,20,23,38,27
23	**19**,20,27
38	**19**,27,21
27	18,19,**11**,38
20	**14**,19,23
21	38,**11**
18	14,**8**,27
8	**7**,14,18



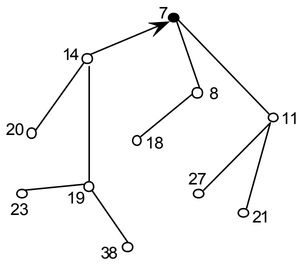
(b) Node 19,11,14,8 and 7 are requested to be as collector

Smart Node	SmartNodeList
7	14,8,11,20,27, 21,18,19
14	**7**,8,18,19,20,23, 38
11	**7**,21,27
19	**14**,23,38,27
23	19,20,27
38	19,27,21
27	18,19,11,38
20	14,19,23
21	38,11
18	14,8,27
8	**7**,14,18



(c) Node 19 forwards the request to node 14, and node 11,14, 8 forward the request to node 7.

Smart Node	SmartNodeList
7	14,8,11,20,27, 21,18,19,23,28
14	**7**,8,18,19,20,23, 38
11	7,21,27
19	14,23,38,27
23	19,20,27
38	19,27,21
27	18,19,11,38
20	14,19,23
21	38,11
18	14,8,27
8	7,14,18



(d) Node 7 is elected as a collector for all other nodes.

Figure 6. Collectors' selection.

4.3.1. Local sensing

Periodically, a smart node checks the replica changes of the node in its monitoring list and computes local dynamic rate of the replicas, β_l. To make the local sensing available, a smart node still holds the information of leaving nodes in its host list until this sensing time period ends in case it loses the trace of those leaving nodes. Suppose the sensing time period is τ and there are ψ_1 node with C_1, ψ_2 node with $C_2,\cdots$, ψ_j node with C_j, $\cdots$, ψ_s node with C_s in last sensing time point t. During this time period, there are $\Delta\psi_1$ nodes with C_1, $\Delta\psi_2$ nodes with $C_2,\cdots$, $\Delta\psi_j$ nodes with C_j, $\cdots$, $\Delta\psi_s$ nodes with C_s no longer active, and ζ_1 nodes with C_1, ζ_2 nodes with $C_2,\cdots$, ζ_i nodes with C_i, $\cdots$, ζ_g nodes with C_g joining the monitoring set. Then, β_l is given by Eq. (10).

$$\beta_l = \frac{\Delta R_l}{R_l \Delta t} = \frac{k\sum_{i=1}^{g} C_i\zeta_i - \sum_{j=1}^{s} C_j \Delta\psi_j}{\tau \sum_{j=1}^{s} C_j \psi_j} \qquad (10)$$

The local sensing period τ for each smart node is equal to the global exchange period, which is a pre-determined system parameter. The τ is adjusted accordingly when a new global exchanging period is adopted.

4.3.2. Global exchanging

Smart nodes obtain the perspective of the global dynamics by periodically exchanging the detected dynamic information of monitored nodes. During a globally exchanging period, each smart node has an option for its role: acting as a collector or a non-collector. A collector is in charge of collecting, computing, and releasing the global dynamic information, while a non-collector only provides its local information and accepts updated global dynamic information from the collector.

A smart node can be a collector candidate if and only if there is no other node in its SmartNodes List has an IP address smaller than itself. Otherwise, it will request the node with the smallest IP address in its SmartNodes List as the collector. The requested node would serve as a collector if and only if there is no other Smart node with a smaller IP address in the SmartNodes List of its own. Otherwise, it would forward/redirect the request to the node with the smallest IP address in its SmartNodes List. Figure 6 plots an example of this election routine. For saving traffic overhead, a requested non-collector node forwards/redirects all requests with only one packet instead of sending a bunch of packets for those requests.

To lessen the processing cost and traffic overhead, a smart node that currently is a non-collector would call the collector election routine only under the following three situations: a) its current collector node is no longer active; b) there exists a new coming smart node with a smaller IP address than the current collector in its SmartNodes List; c) as a newly joining node, when there is no any collector node. On the other hand, a smart node serving as a collector would call the collector election routine only if it includes a new smart node with a smaller IP address into its SmartNodes List.

The global exchange period is divided into two sub-periods, shown in Fig. 7. In the first sub-period, a collector issues Smart PING messages, in which the "calculating bit" is set to "1", to all the requesting non-collectors and waits for replied PONG messages. Each non-collector provides the dynamic rate of its local replicas, β_l, as well as the size of its monitoring set, n_l, to the collector with a Smart PONG message.

In the second sub-period, the collector collects all received β_l and computes β_g. Suppose the collector has received r PONG messages, each with information (β_{li}, n_{li}), it computes β_g as follows.

$$\beta_g = (\sum_{i=1}^{r} n_{li}\beta_{li}) \Big/ \sum_{i=1}^{r} n_{li} \qquad (11)$$

The collector then forwards the updated β_g to all of the requesting smart nodes with a Smart PING message, in which the "calculating bit" is reset to "0". When the requesting smart nodes receive the message, they deliver the information to all their monitoring nodes also by sending Smart PING messages.

4.4. Search selection

PASH utilizes the global dynamic information to adjust item popularity when publishing a query or gossiping the item popularity to other nodes.

Whenever a node receives a query for item i, it will compute the popularity value of item i before making a search decision. For any node, suppose current time is t, the popularity value for item i that the node has held is $p_i(t_0)$ with a timestamp t_0, where $t_0 < t$, it adjusts $p_i(t_0)$ into $p_i(t)$ by using Eq. (12).

$$p_{\mathrm{i}}(t) \leftarrow p_{\mathrm{i}}(t_0) \cdot \gamma = p_{\mathrm{i}}(t_0)[1 + \beta_g(t - t_0)] \qquad (12)$$

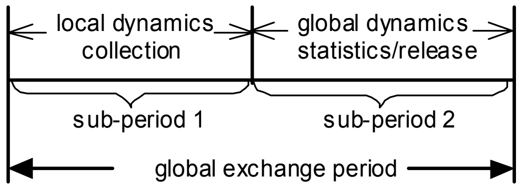


Figure 7. Global exchange period and sub-periods.

If $p_i(t)$ exceeds the threshold T_f, the query will be flooded. Otherwise, the query will be issued via DHT.

During the Gossip process, if a node has known a v_i, the replica statistic of item i with a timestamp t_1. When receiving a gossip message from a node j at time t, within which the replica statistic of item i is v_{ij} with a timestamp t_2, it updates v_i by using Eq. (13)

$$\begin{aligned} v_i &= \gamma_1 v_i \forall \gamma_2 v_{ij} = v_i[1+\beta_g(t-t_1)] \forall v_{ij}[1+\beta_g(t-t_2)] \\ &= \begin{cases} v_i[1+\beta_g(t-t_1)], & if \;\; v_i[1+\beta_g(t-t_1)] \geq v_{ij}\left[1+\beta_g(t-t_2)\right] \\ v_{ij}[1+\beta_g(t-t_2)] & if \;\; v_i[1+\beta_g(t-t_1)] < v_{ij}[1+\beta_g(t-t_2)] \end{cases} \end{aligned} \qquad (13)$$

5. Simulation Methodology and Metrics

We evaluate PASH through synthetic and trace-driven simulations. The Ion P2P Snapshots [20] we used in this work include topologies of a hybrid Gnutella system from 2004 to 2005. Based on the real P2P topologies from this trace, we construct our testbed with a P2P network including $10^3 \sim 10^4$ peers. To perform the flooding and DHT search simultaneously, we mainly include ultrapeers into our testing topologies. Each peer holds resources which follow the Zipf distribution. The physical internet layer is generated by BRITE [13], in which the internet topology holds about 30000 nodes. Peers are joining and leaving the network based on existing observations on P2P behaviors [18].

In our simulation, we construct the DHT by using the SHA-1 algorithm. Therefore, any key stored in nodes has a size of 20 bytes. The flooding search is performed by employing the Breadth First Search (BFS) algorithm. To investigate the search performance, we simulate 10^5 queries iteratively for each run and report the average of 30 runs.

We use the following metrics to evaluate PASH performance.

Estimation error. It is used to evaluate the accuracy of estimated popularity values of items when performing PASH. Since the dynamic change of P2P systems is considered by PASH, we expect PASH can report more accurate popularity values of items when making the selection between flooding and DHT.

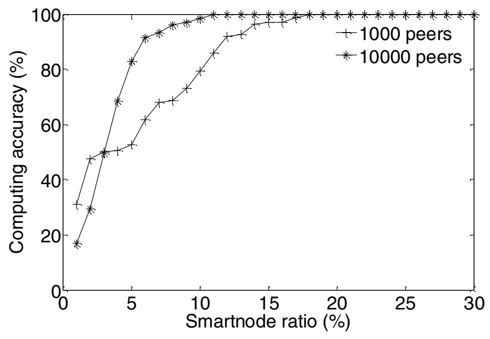


Figure 8. Smart node ratio determination.

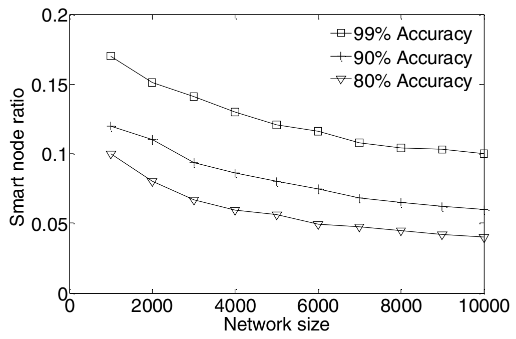


Figure 9. Computing accuracy.

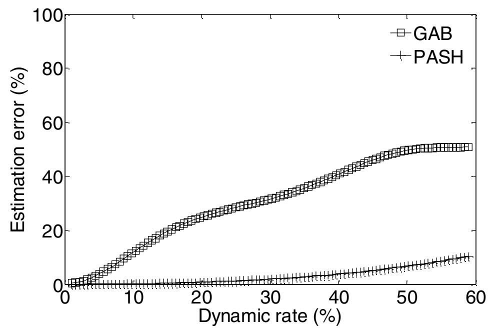


Figure 10. Estimation error vs. system dynamic rate (Topology size is 10^3).

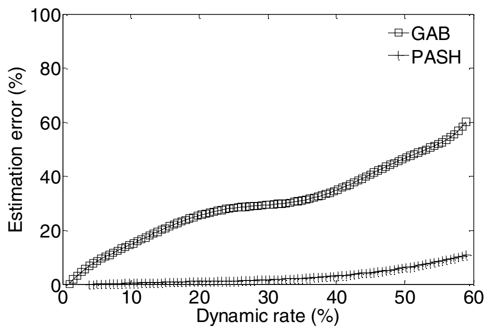


Figure 11. Estimation error vs. system dynamic rate (Topology size is 10^4).

Traffic overhead. From the network administrators' point of view, traffic cost is the most important metric to reflect the impact caused by P2P applications. In the P2P overlay, each edge is uniquely mapped into a path in the underlying internet layer, whose length is *l*. In one query cycle, we calculate the sum of the distances of the enrolled edges that this message traversed. Thus, the traffic overhead of a query cycle can be computed as $H = M \times L = \sum |m_i| \times l_i$, $1 \leq i \leq e$, where $|m_i|$ is the size of messages that traverse, and *e* is number of enrolled edges. We report the traffic overhead per link in performing PASH to reflect the reduced traffics cost.

Response time. This is an important metric which end users of P2P systems are mainly concerned. Shorter response time leads to higher degree of satisfaction and better service quality. In this work, we define the response time as the period of time from starting a search to receiving the first response, and show the response time reduction achieved by PASH.

We also examine several system parameters of PASH, such as the smart node ratio and algorithm convergence. These important parameters guarantee the effectiveness and efficiency of PASH.

6. Performance Evaluation

We first determine the best smart node ratio of PASH. As mentioned in Section 4, we employ a probability based smart node assignment scheme. We compute the expected sensing accuracy of different traces. Figure 8 plots the expected smart node ratio *x/n* in two representative P2P topologies. For any given P2P topology, the sensing accuracy increases when enlarging the *x/n* ratio. We increase the smart node ratio *x/n* from 0, and stop when the derivative of sensing accuracy tends towards zero. At this point, we say the system obtains a sufficient smart ratio. The results show that the proper value of *x/n* is about 15% when there are about 1000 nodes, whereas this value is less than 10% when the size of P2P topology reaches 10^4.

We find the proper ratio would decrease when the size of the system increases. Figure 9 depicts this trend with different computing accuracy. Indeed, the larger the P2P network size is, the less smart nodes are needed. Following our smart node generation strategy, PASH always selects those nodes with a probability $p = S \times x/n$, where *S* is the number of candidate peers. In this way, those nodes with high connection degree would be selected as smart nodes with a higher probability.

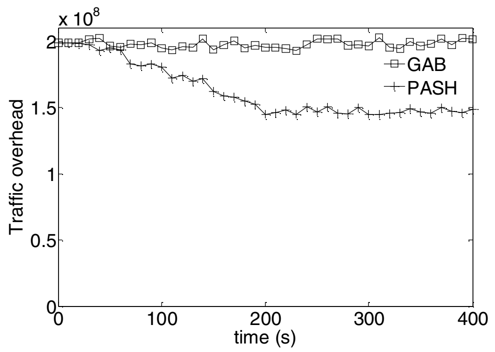


Figure 12. Traffic overhead.

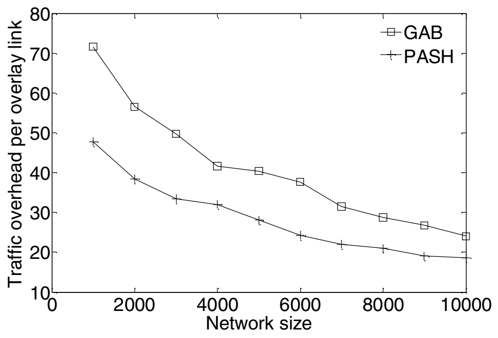


Figure 13. Traffic overhead per overlay link.

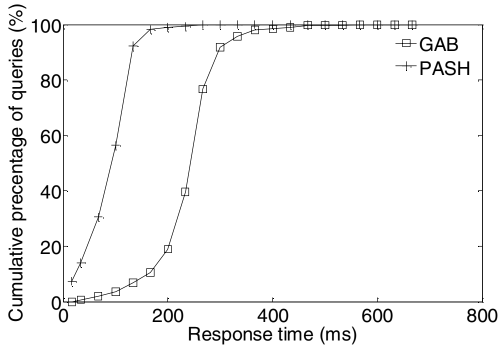


Figure 14. CDF curve of response time of 10^5 queries (Topology size is 10^3).

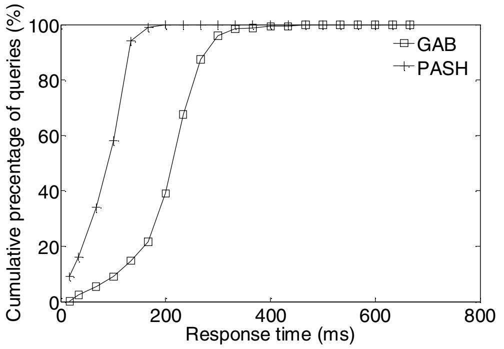


Figure 15. CDF curve of response time of 10^5 queries (Topology size is 10^4).

As shown in Fig. 9, PASH reduces the smart node ratio when increasing the system size, while the computing accuracy of popularity values can still be guaranteed.

After determining the proper smart node ratio, PASH makes use of it to compute the popularity values for the requested items. Here we compare PASH with GAB in the estimating accuracy of item popularity. Both GAB and PASH would incur some estimation errors in the flooding/DHT decision. PASH reduces the errors by involving the system dynamics into the estimation. We define the estimation error ε by $\varepsilon = (E - P) / P$, where the E is estimation value and P is the exact popularity value of an item. Figures 10 and 11 show that PASH outperforms GAB in estimating the accuracy of popularity values of items in a dynamic P2P system.

Another key issue is the convergence of PASH, that is, how fast the PASH is able to enter a stable state. To this end, we keep inserting queries into the system at a fixed speed, around 100 queries per second. The desired objects follow the Zipf distribution [13]. We also assign each node a lifetime that follows a log-quadratic curve [18], which can be approximated by using two Zipf distributions. We observe the change of network traffic in the system. At the very beginning, PASH has minor impact on the system as peers do not come and leave frequently, and the traffic overhead of using GAB is similar to PASH, as shown in Fig. 12. Later, when the network tunes due to the peers' joining and leaving, the traffic overhead decreases as PASH avoids unnecessary flooding. Specifically, PASH works steadily after about 200s.

In Fig. 13, we show the average traffic overhead of PASH on P2P overlay. The curves indicate that PASH reduces traffic overhead on overlay link by about 20-35%. Note that we let both protocols search the same items. PASH is particularly effective for those items whose popularity values are close to the flooding/DHT threshold. Another benefit in using PASH is the reduction of response time. Figures 14 and 15 show that the average response time of queries is reduced by 50%. It is because that the more accurate selections between flooding and DHT help PASH outperform GAB-like protocols in both the traffic overhead and response time.

7. Conclusion

In a hybrid P2P system, flooding and DHT are both employed for content locating. The decision to use either a flooding search or a DHT search depends on the population of desired data. Existing works do not consider the dynamic factors of P2P systems when counting data popularities.

In this paper, we propose PASH, which provides an efficient and accurate search decision mechanism by dynamically detecting the P2P overlay change and hereby refining the estimation of the content popularity. By enrolling the P2P dynamic factor, not only the traffic overhead can be reduced in P2P systems, end users can also obtain the desired resource with shorter search latency. The simulation results validate the effectiveness of our overall design – making proper search selection decision and saving the traffic cost as well as the response time.

Acknowledgements

This work was supported in part by the NSFC grant No. 60573140, the Hong Kong RGC grants HKUST6264/04E and HKUST6152/06E.

References

[1] Gnutella. http://gnutella.wego.com/.

[2] Gnutella protocol specification. http://rfc-gnutella.sourceforge.net.

[3] KaZaA. http://www.kazaa.com.

[4] Napster. http://www.napster.com.

[5] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking Up Data in P2P Systems. *Comm. ACM*, 2003.

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

[7] K. Hildrum, J. D. Kubatowicz, S. Rao, and B. Y. Zhao. Distributed Object Location in a Dynamic Network. In Proceedings of *ACM Symp. on Parallel Algorithms and Architectures*, 2002.

[8] H. Jiang and S. Jin. Exploiting Dynamic Querying like Flooding Techniques for Unstructured Peer-to-peer Networks. In Proceedings of *IEEE ICNP*, 2005.

[9] B. Liu, W. C. Lee, and D. L. Lee. Supporting Complex Multi-dimensional Queries in P2P Systems. In Proceedings of *IEEE ICDCS*, 2005.

[10] B. T. Loo, J. M. Hellerstein, R. Huebsch, S. Shenker, and I. Stoica. Enhancing P2P File-Sharing with an Internet-Scale Query Processor. In Proceedings of *VLDB*, 2004.

[11] B. T. Loo, R. Huebsch, I. Stoica, and J. M. Hellerstein. The Case for a Hybrid P2P Search Infrastructure. In Proceedings of *IPTPS*, 2004.

[12] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In Proceedings of *ACM ICS*, 2002.

[13] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In Proceedings of *the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS)*, 2001.

[14] D. Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks. In Proceedings of *ACM SIGCOMM*, 2004.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-addressable Network. In Proceedings of *ACM SIGCOMM*, 2001.

[16] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-scale Peer-to-Peer Systems. In Proceedings of *ICDS*, 2001.

[17] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In Proceedings of *the Multimedia Computing and Networking(MMCN)*, 2002.

[18] M. T. Schlosser and S. D. Kamvar. Availability and locality measurements of peer-to-peer file systems. In Proceedings of *ITCom: Scalability and Traffic Control in IP Networks*, 2002.

[19] 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.

[20] D. Stutzbach and R. Rejaie. Characterizing the Two-Tier Gnutella Topology. In Proceedings of *ACM SIGMETRICS*, 2005.

[21] G. Swamynathan, B. Y. Zhao, and K. C. Almeroth. Exploring the Feasibility of Proactive Reputations. In Proceedings of *IPTPS*, 2006.

[22] C. Wu and B. Li. rStream: Resilient Peer-to-Peer Streaming with Rateless Codes. In Proceedings of *ACM Multimedia*, 2005.

[23] D. Xuan, S. Chellappan, X. Wang, and S. Wang. Analyzing the Secure Overlay Services Architecture under Intelligent DDoS Attacks. In Proceedings of *IEEE ICDCS*, 2004.

[24] L. Yin and G. Cao. DUP: Dynamic-tree Based Update Propagation in Peer-to-Peer Networks. In Proceedings of *IEEE ICDE*, 2005.

[25] M. Zaharia and S. Keshav. Gossip-based Search Selection in Hybrid Peer-to-Peer Networks. In Proceedings of *IPTPS*, 2006.

[26] Z. Zhang, S. Chen, Y. Ling, and R. Chow. Resilient Capacity-Aware Multicast Based on Overlay Networks. In Proceedings of *IEEE ICDCS*, 2005.

[27] M. Zhong and K. Shen. Popularity-Biased Random Walks for Peer-to-Peer Search under the Square-Root Principle. In Proceedings of *the International Workshop on Peer-To-Peer Systems(IPTPS)*, 2006.