# Usage-aware Search in Peer-to-Peer Systems

Irene Sygkouna and Miltiades Anagnostou

School of Electrical and Computer Engineering, National Technical University of Athens,
9, Heroon Polytechneiou Str., 15773, Zografou, Athens, Greece
{isygk, miltos}@telecom.ntua.gr

**Abstract.** We study solutions to a source discovery problem defined in the framework of providing time-critical context-aware services over a Peer-to-Peer communication paradigm. The proposed mechanisms, which take place in ubiquitous computing environments, exploit the locality of reference properties exhibited by context usage patterns, with efficient means provided by Active Networks. Simulation results show that the new methods reduce network traffic while they maintain the good search time of flood broadcasting methods.

**Keywords:** Ubiquitous computing, Context-aware services, Peer-to-Peer search, Locality of reference properties, Active Networks.

## 1 Introduction

Ubiquitous computing paradigm offers users the opportunity to truly perform their operations anywhere and anytime. Many practitioners envision a future empowered with context-aware computation and communication infrastructure that allow users to access data and receive service at any place and any time. A context-aware system can be seen as a human assistant given user's context to be responsible to make decisions in a proactive fashion, anticipating user needs while not disturbing the user, except for an emergency [1]. The exploitation of complete context-awareness in state-of-the-art services requires taking advantage of all types of context available to them. Thus, context-aware services (CASs) need a flow of information from and about their environment, in order to be able to adapt to it [2]. Elaborating on the efficient provisioning of CASs, the current work is based on a framework in which a CAS generates context requests and addresses them to the nearest broker. Brokers act as mediatory players between CASs and context sources. The need to support CASs that are highly robust and can scale well with the number of nodes and information sources points to Peer-to-Peer (P2P) architecture [3] as a more promising approach than most centralized solutions offer. Each peer Context Broker (CB) can make information available for distribution and depends on each other for getting information, forwarding requests, etc.

The system faces the challenge to ensure efficient and scalable distribution of context information. Among the most popular approaches applied to improve performance are replication, caching, and intelligent routing. In our framework, the first two are considered less appropriate due to the volatile nature of context information, and thus we

are concerned with routing techniques applied every time a request arises and a real-time search has to take place. In its purest form, the P2P model uses message forwarding mechanisms to search for information since there is no centralized server with a global view of all the peers in the network or the information they provide. The respective algorithms are typically implemented in the form of an application-level protocol. However, most existing techniques result in opposite extremes of bandwidth and response time. In this paper, we propose a set of usage-aware mechanisms that exploit the locality of reference properties exhibited by context usage patterns. The context demand profile is monitored by the distributed peer nodes, which exchange their knowledge with efficient means provided by Active Networks. The results show that the proposed mechanisms reduce the network traffic while they maintain the good search time of flood broadcasting methods.

Section 2 presents a literature overview on decentralized search algorithms and section 3 describes the problem we deal with. Section 4 provides first the definition of the locality properties, and then describes the role of Active Networks in searching along with the proposed search mechanisms. Simulation results are presented in section 5 and section 6 concludes the paper.

## 2 Literature Overview

*Resource Discovery* constitutes a fundamental problem in large-scale distributed systems, and even though finding resources in a network of computers is a problem probably as old as distributed computing itself, different system requirements and conditions of current large-scale applications have led to a flurry of different approaches to the problem. Thus, the problem of searching for information in P2P networks can be treated in different ways, ranging from centralized indexing schemes, such as Napster [4], to decentralized mechanisms that navigate the underlying network without knowledge of its global structure.

Decentralized search algorithms have been studied both for *unstructured* and *structured* systems. In the former case, there is not any precise control over the network topology or data placement. A client seeking information searches across scattered collections stored at numerous member nodes by forwarding queries to one's neighbors until the target is found. Moreover, both search by identifier and by content is supported. Gnutella [5] uses flood routing to broadcast queries, generating a large amount of unnecessary traffic. There have been continuous efforts to improve the naïve search algorithm based on flooding. The authors of [6] proposed *Random Walk*, which forwards a query to a randomly chosen neighbor at each step and *Expanding Ring*, which performs successive floods with increasing time-to-live (TTL), until the target is found. It was shown through extensive experiments that a 32-walker Random Walk reduces the amount of network traffic by two orders of magnitude at the expense of slight decrease in search speed and generally outperforms Expanding Ring as well. Similar strategies, including Expanding Ring and a variation of Random Walks were examined in [7, 8].

In [9], hybrid search schemes that combine Flooding and Random Walks were proposed, which rectify the performance of Flooding in the case of a sparse network with a few vertices of large degrees. Moreover, it was shown that *Random Walk with*

*Local Flooding*, which performs shallow floodings on each step of the random walk, is more preferable than simple random walk on regular graphs since it achieves savings in response time, while the savings are much sharper if the graph has supernodes. In [10] a probabilistic message dissemination method was developed. By altering the probability of routing search request messages, it varies the probability that the search is successful. Contrary to the above works that propose alternatives to Flooding based mainly on specific properties of the network topology, in the current paper we expect to reduce network traffic without retarding search, by taking into account the locality of reference properties that are likely to be exhibited by the monitored context usage patterns.

In structured systems, objects are placed not at random nodes but at specified locations that will make subsequent queries easier to satisfy. Such systems (e.g. [11, 12, 13, 14, 15]) implement Distributed Hash Tables (DHTs), and search is performed by looking up the DHTs. In more recent works, techniques based on DHTs have been proposed for multiple-keyword search [16] and full-text search [17, 18]. They differ from our approach since we are interested in finding sources that are managed by peers, which have complete autonomy over their location.

## 3 Problem Description

We consider a set of Context Brokers forming an overlay network. Each Broker manages a set of local information sources registered to it via a registration protocol and thus it maintains the necessary interfaces for interacting with its local sources. Peers can use an enquiry protocol to query other peers in order to discover sources. Once a Broker receives a search request coming from either another peer or a local consumer (e.g. a CAS), it first looks up the request in its local information. If a matching source is not found, the Broker forwards the request to different peers until the target source is located. Precluding dependence from central control, the aim is to design efficient mechanisms for discovering and retrieving data. Due to the volatile nature of context, we deal only with dynamic information sources and thus, there is no caching support for actual data.

Each peer maintains a local directory, the *Local Sources Directory* (LSD), with entries to the sources it manages. Note that sources cannot be replicated. Each peer maintains additionally the *Remote Sources Directory* (RSD), which caches directory entries for sources maintained by other peers. An entry in the RSD is a pair (*source_info*, *loc*), where *source_info* provides a description of the information produced by a source and *loc* is the network address of the peer that is presumed to manage the given source.

Each peer *n* has a local neighborhood, denoted by *N(n)* and defined as the set of peers that are close (e.g., at one hop distance or within the same local area network) to that peer. Finally, global network topology is unknown and a peer only contacts peers in its neighborhood, as well as peers indicated in its RSD.

## 4 Solution

Based on the distributed computing environment provided by Active Networks, we propose specific algorithmic solutions built on top of the P2P communication paradigm,

that aim to enhance the system's efficiency and scalability for the provision of time-critical CASs. The relevant algorithms avoid the inefficiency of flood broadcasting methods, and a source exhibiting a locality of reference property can be easily located by applying an appropriate limited flooding algorithm, according to the type of locality.

### 4.1 Locality of Reference Properties

We define the following notation and terminology: Given a set of nodes ( $n \in N$ ) and a set of objects ( $o \in O$ ), we denote by $r(o,n)$ a request ( $r \in R$ ), where *o* ( $o \in O$ ) is the requested object and *n* ( $n \in N$ ) the node the given request originated from, henceforth called *initiatory-node* of the request. Moreover, we will use the term *home-node* of an object to refer to the node that hosts the source of the given object. Note that in the following text, the words "source" and "object" are used interchangeably.

**Temporal Locality.** It implies that an object frequently accessed in the past, namely a popular object, is likely to be accessed in the future [19]. We define the popularity f of an object o as:

$$f_o = \sum_{n \in N} \left| R_n^{\,o} \right| \ . \qquad \textbf{(1)}$$

where N is the set of nodes, and $R_n^{\,o}$ the set of requests for object o initiated from node $n \in N$ , namely: $R_n^{\,o} = \left\{ \forall\, r(o',n') \in R \;\middle|\; o' = o, \quad n' = n \right\}$.

Observing the context usage patterns in our real test scenarios conducted under the IST project Context [20] we concluded that temporal locality was evident since a few objects were most popular and thus were repeatedly requested from the majority of peers. Such objects usually refer to some elementary types of information that constitute basic components of many complex types and are thus repeatedly requested. For example, a source that provides pure location information of mobile users for a wide geographical area is most likely to be accessed repeatedly, since location information is a basic component of many complex and more specialized types of information.

**Geographical Locality.** It accounts for the location of the nodes from which a repeated request originates and implies that an object accessed by a client is likely to be accessed again in the future by "nearby" clients [19]. We first define the set $N(n_o, l) \subseteq N$ , which consists of the nodes included in a geographical area centered at $n_o \in N$ and extended at distance *l*, as:

$$N(n_o, l) = \left\{ \forall n \in N \;\middle|\; dist(n, n_0) < l \right\} . \qquad \textbf{(2)}$$

where *dist(n, n0)* represents the path length between nodes *n*, $n_o$. An object *o* exhibits geographical locality with radius *L*, if a significant number of repeated requests originate from the same geographical area, which is extended around the home-node of *o*, namely:

$$\sum_{n \in N} \frac{\left|R_n^{\,o}\right| \cdot x(n, N(H(o), L))}{f_o} > T_{geo} \ . \qquad (3)$$

where $f_o$ represents the popularity of *o*, *H(o)* its home-node, $R_n^{\,o}$ the set of repeated requests initiated from node $n \in N$, $T_{geo}$ a certain threshold and *x(n,N)* equals to 1 (0) if $n \in N$ ($n \notin N$).

Geographical locality is most likely to be exhibited by context usage patterns, since CASs are usually developed and designated to be provided at specific areas, usually near the sources of information they utilize. For instance, in a university campus, a source that provides academic-related context information is meaningful for and thus used from relevant services offered in the campus, which constitutes a case of geographical locality.

**Spatial Locality.** It is looking for dependencies among the requested objects and implies that objects neighbouring an object frequently accessed in the past are likely to be accessed in the future. Defining a traversal stride to be a sequence of requests where the time between successive requests is less that StrideTimeout seconds [19], an object *o* exhibits spatial locality, if there exists a traversal stride $s_o$ starting with object *o*, such that:

$$\frac{f_{s_o}}{f_o} > T_{spat} \ . \qquad (4)$$

where $f_o$ is the popularity of object *o*, $f_{s_o}$ the popularity of the traversal stride $s_o$ and $T_{spat}$ a given threshold.

Spatial locality of reference is likely to be found in a context access pattern due to the modular design based on which context objects are constructed. In particular, a context object may be composed of other simpler objects in a cascaded way, requiring a cascaded assignment of values to each object in turn.

### 4.2 Active Networks Distributed Computing

Active network developers envisage transforming IP packets into encapsulated fragments of executable code that traverse the network and execute in limited environments at intermediate nodes. P2P networking, on the other hand, was devised as a lightweight, primitive, networking concept that achieves adequate results at minimum cost without sophisticated network protocols. We could thus easily think of active capsules that carry P2P queries to locate particular information, or code for determining

traffic and usage patterns for individual sources at each node and dynamically make decisions on redistributing information across the P2P network [3].

Because P2P queries are lightweight, the mobile code would pose a minimal computational burden and impose minimal network overhead, leading to a highly efficient self-sustaining and self-maintaining P2P system. In this framework, *passive monitoring* of context usage patterns is supported, which substantially reduces traffic by piggybacking the relative information on existing active packets traversing the network, as opposed to *active monitoring*, in which the measurements are done by sending additional control messages [21].

### 4.3 Usage-aware Search Mechanisms

The proposed mechanisms are based on a limited version of the flooding algorithm that takes advantage of the locality of reference properties exhibited by the monitored context usage patterns. We assume that appropriate Tables maintained by all the peers provide information on the objects exhibiting locality properties: *T-Table* for temporal locality, *G-Table* for geographical locality and *S-Table* for spatial locality. When a peer initiates a request, it first searches the local copies of the Tables to find potential matching sources. In case of a hit, it applies an appropriate limited flooding algorithm, depending on the type of property. Otherwise, it resorts to pure Flooding. In each case, once the requested object is found, the response message follows the reverse path to reach the initiatory-node and a new entry is created in the RSD of the initiatory-node. Note that the Tables are consulted once for each request, namely at the initiatory-node of the request, whereas the RSDs at each visited peer.

The rationale behind the proposed approach is based on two observations: first, only a small percentage of all the available objects are likely to exhibit a locality of reference property. Second, most requests refer to such objects. Therefore, we expect that the majority of requests can be satisfied by looking up the Tables and applying limited flooding, whereas the locality information requires limited storage space and thus can be retrieved quickly with small overhead.

**Range-limited Flooding.** It proposes to limit the range of flooding to an extent determined by the popularity of the object requested and in a way to save as much bandwidth as possible without increasing the time to locate the appropriate source. Thus, the more popular the object, the lower the number of neighbours $K$ to which the request should be forwarded in each step, since popularity provides a clue about the number of previous repeated requests for the object, which should have thus located the corresponding source. Supposing that peer $i$ receives a request $r(o,n)$, $K$ is given by:

$$K \propto \frac{1}{f_o} \cdot d_i \ . \qquad (5)$$

where $f_o$ is the popularity of $o$ and $d_i$ the degree of $i$.

We assume that the T-Table hosts popularity information of the most popular objects. An entry ($o$, $f_o$) indicates the popularity $f_o$ of an object $o$, as measured by its home-node,

which keeps logs of the requests it has serviced. An object *o* is registered with T-Table only if its popularity is greater than a given threshold *a*, while an update operation is performed only if a noticeable change, which is determined by a given percentage threshold *b*, is recorded by its home-node. In case a peer decides to perform a register or an update operation, the respective information will be encapsulated in a following request message initiated from that peer. In this case, the respective active packet has to load a modified code that executes both the query and the register/update operation in each visited peer, and thus updates all the copies of T-Table.

**Depth-limited Flooding.** It is proposed as an alternative that aims to save bandwidth by limiting the depth (TTL) of flooding. The success of this algorithm when searching for an object is based on a strong indication that the given object is most likely to be located following a limited number of hops. Therefore, if an object proves to exhibit geographical locality of reference with radius *L*, it is likely that the same object will be requested from nearby nodes in the future. These nearby nodes could limit the depth of flooding in following repeated requests initiated from them to the value *D=L*, with the expectation that either the source itself will be located nearby or the location of the source will be found cached in the RSD of a nearby node.

The existence of geographical locality of reference for a given object is verified by its home-node. Each peer maintains information related to its broader neighborhood, namely the nodes located in close geographic proximity, and periodically examines the request history of each source it owns, so as to correlate the initiatory nodes of the relevant requests. It thus registers every object that proves to exhibit geographical locality, based on inequality (3), with the G-Table maintained by each node in its broader neighborhood. A register/delete operation is performed by encapsulating the respective information in a following request message initiated from this peer.

**Prefetch-limited Flooding.** It exploits the idea of prefetching within flooding, namely search for more than one object within a single flooding, in anticipation of future requests. In particular, when a node initiates a request for an object that proves to exhibit spatial locality of reference, it piggybacks its request with a set of queries for the dependent objects that it speculates will be requested by the client in the near future, in order to resolve all of them during the same flooding search.

Contrary to the other properties that are server-detected, a thorough analysis of the clients' access patterns is required in order to identify the spatial locality of reference that may exist among the various objects. Assuming that each peer keeps logs of the requests it has initiated, it is capable of testing for the existence of the property locally for every object that is frequently requested, based on the inequality (4). Note that each entry ($o$, $s_o$) of the S-Table indicates the object *o* that exhibits spatial locality and the respective stride $s_o$. Every peer that frequently initiates requests for an object registered in the S-Table is responsible for verifying that this object indeed exhibits spatial locality. In the opposite case, it should initiate a delete operation in order to remove the object from all the S-Tables. The register/delete operation is performed with the same piggybacking mechanism.

## 5 Simulation Results

We have implemented an event-based multithreaded simulation in *Java* to test our algorithms. As a reference for comparison we use the pure Flooding, which achieves the best search time at the cost of intensive network use, and the Random Walk with Local Flooding of TTL=1, which was proposed recently as an alternative to the pure Random Walk that improves its search time. Henceforth, the latter one will be called Random Walk for the sake of brevity. We assume constant peer participation, no failures, and that the sources do not migrate during the simulation. The simulated requests are simple and there is exactly one matching source to a request. Moreover, we assume that the RSD size is infinite. The initial topology of the overlay network, as formed by the neighbors' connections, is modeled as a random graph that is generated based on the *Waxman model* [22]. We follow the *analytic workload generation* method, which starts with models for various workload characteristics i.e., the locality of reference properties in our case, and then generates outputs that adhere to these models [23]. Since each property is proposed to be exploited in a different way, we generate distinct synthetic traces for each property in order to experiment exclusively with the potential benefits achieved in each case. Zip's law [24] is used to model the distribution of context requests and the default TTL value is set to the graph diameter. As a metric for the time to locate a source we use the *average path-length per request* (defined as the ratio of the total number of hops incurred across all requests to the total number of requests), while the bandwidth consumption is well captured by the *average number of messages per request* (defined as the ratio of the total number of messages sent across all requests to the number of all requests).

### 5.1 Test Case 1: Temporal Locality

The network size equals to 200 nodes and each peer node controls one source, for a total of 200 sources. Each peer monitors the popularity of the local sources and updates the T-Tables accordingly. For simplicity we assume that the T-Tables are updated at constant periods, namely every time the overlay has executed 100 new requests, for a total of 1000 requests. Fig.1 and Fig.2 depict the evolution of the average path-length and the average number of messages per request, over time, under Range-limited Flooding (R-F), Flooding (F) and Random Walk (RW), for two different values of the Zipf parameter *a*, namely  =0.7 and  =0.95, respectively. The time evolution is given in terms of consecutive time periods, each corresponding to the execution of 100 requests.

Clearly, the average path-length per request decreases over time and R-F achieves to follow F very closely. Similarly, the average number of messages per request decreases with time, but in this case R-F clearly outperforms F, achieving lower bandwidth waste. Moreover, the bandwidth savings achieved by R-F becomes more intense with time. Comparing R-F with RW, it becomes obvious that the latter one achieves high bandwidth savings, but at the cost of a noticeable increase in the average path-length per request. It is also remarkable that the performance of RW improves with time. In particular, the bandwidth savings it achieves with regard to R-F increases from 53% to 78%, whereas the path-length savings of R-F with regard to RW decreases from 89% to 73%.

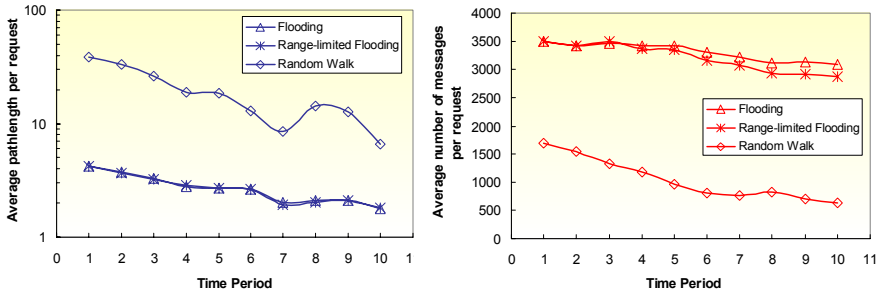


**Fig 1.** Performance of R-F versus F and RW, for =0.7

Moreover, the value of *a* affects the bandwidth savings achieved by R-F with reference to F. In particular, usage patterns with a higher skew in popularity distribution seem to take greater advantage of the savings achieved by R-F (Fig.2). On the other hand, the percentage of bandwidth savings and path-length waste achieved by RW with regard to R-F seem not to be affected from the value of .

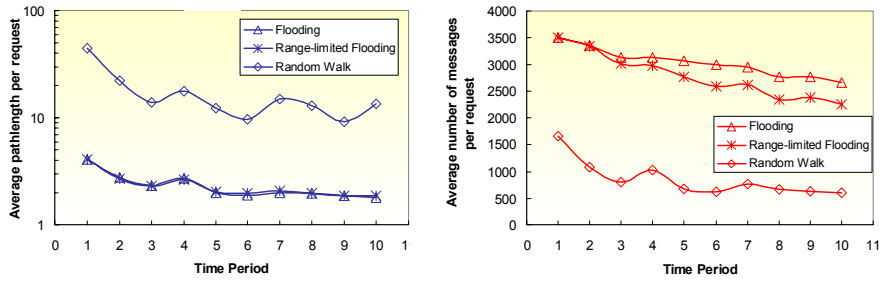


**Fig. 2.** Performance of R-F versus F and RW, for =0.95

### 5.2 Test Case 2: Geographical Locality

We first consider an overlay graph of 500 nodes and density equal to 0.035. Each peer maintains 10 sources. We experiment with synthetic traces of 400 requests produced according to Zipf distribution (with $a$=0.7), that exhibit geographical locality at different degrees. The degree, namely the fraction of the requests executed in the whole trace that exhibit geographical locality, has been set to 0.2, 0.4, 0.6, and 0.8, with 0.48%, 1.28%, 2.72% and 4.32% of the most popular objects exhibiting the property, respectively. We evaluate the performance of pure Flooding (F), Random Walk (RW) and Depth-limited Flooding (D-F), assuming that the last one has set the radius of geographical locality to 3 and then we repeat the experiment on a graph of density 0.07. The results are depicted in Fig.3 and Fig.4, respectively.

As the degree of geographical locality increases, the average path-length drops and D-F achieves to maintain the path-length at the same level with F. On the other hand, while

the average number of messages per request remains almost constant under F as the degree increases, D-F achieves growing savings. Comparing the performance of RW to the one of D-F, we observe that the bandwidth savings achieved by the former with regard to the latter is on the order of 50%, whereas the path-length savings of the latter with regard to the former is 90%. Moreover, both values are not affected by the variation of the degree of geographical locality.

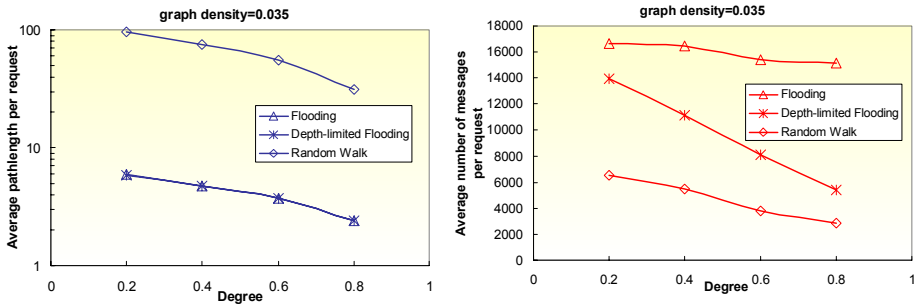


**Fig. 3.** Performance of D-F versus F and RW for graph density equal to 0.035

As far as the graph density is concerned, the sparser the graph, the more profitable the application of D-F compared to F would be. On the other hand, the performance of D-F compared to RW seems not to be affected by the graph density.

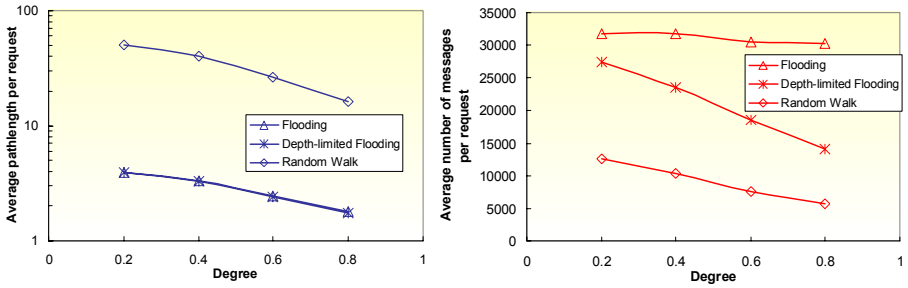


**Fig. 4.** Performance of D-F versus F and RW for graph density equal to 0.07

### 5.3 Test Case 3: Spatial Locality

The goal of the third experiment is to quantify the cost of Prefetch-oriented Flooding (P-F) in terms of the degree of spatial locality exhibited by a trace. The network graph consists of 200 nodes, each maintaining 1 source. We generated 4 different synthetic traces according to Zipf distribution ($a$=0.7), each consisting of 1000 requests. The degree of spatial locality in each of them is adjusted to 0.2, 0.4, 0.6 and 0.8, respectively, with 2.5%, 10%, 26% and 54.5% of the most popular objects exhibiting the property, respectively. The traversal strides consist of 2 objects. Measuring the same metrics, the graphs that result are depicted in Fig.5.

Clearly, while F is not affected notably by the degree variation, P-F achieves significant savings in terms of the average path-length and the number of messages per request, which become more evident as the degree increases.

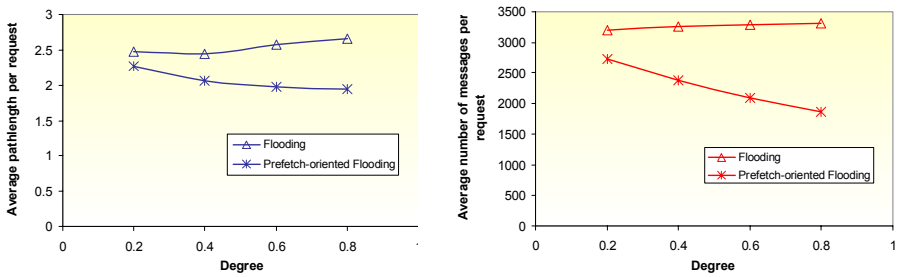


**Fig. 5.** Performance of P-F versus F

## 6 Conclusions

While the majority of mechanisms proposed in literature are based on network topology properties, the current work shows the potential of exploiting usage-awareness toward improving search efficiency. The simulation results show that the new mechanisms maintain the good search time of Flooding, while they achieve to reduce the bandwidth consumption that tends to cripple such systems. Moreover, the degree of bandwidth savings is tightly connected to the degree of the locality of reference properties exhibited. In the proposed framework, the overhead imposed by usage-awareness is kept low because the locality-Tables are of small-size, with light registries, and no additional traffic is needed to maintain them. We thus believe that the current work could be considered as a first step toward proposing even more powerful mechanisms that combine both approaches. It therefore contributes to a growing development toward economic activity in decentralized search mechanisms.

## References

1. Satyanarayanan, M.: Challenges in Implementing a Context-Aware System. Editorial Introduction in IEEE Pervasive Computing, 2 (2002)
2. Xynogalas, S., Chantzara, M., Sygkouna, I., Vrontis, S., Roussaki, I., Anagnostou, M.: Context Management for the Provision of Adaptive Services to Roaming Users. IEEE Wireless Communications 11 (2), 40--47 (2004)
3. Parameswaran, M., Susarla, A., Whinston, A.B.: P2P Networking: An Information-Sharing Alternative. Computer 34 (7), 31--38 (2001)
4. Napster, http://www.napster.com
5. Kan, G.: Gnutella. In: Oram, A. (eds.) Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly (2001)
6. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and Replication in Unstructured Peer-to-peer Networks. In: International Conference on Supercomputing, pp. 84--95. ACM Press, New York, USA (2002)

7. Yang B., Garcia-Molina, H.: “Efficient Search in Peer-to-peer Networks”, In: IEEE ICDCS, IEEE Press, Vienna, Austria (2002)
8. Gkantsidis, C., Mihail, M., Saberi, A.: Random Walks in Peer-to-peer Networks. In: Infocom 2004, pp. 120--130. IEEE Press, Hong Kong, China, (2004)
9. Gkantsidis, C. Mihail, M., Saberi, A.: Hybrid Search Schemes for Unstructured Peer-to-Peer Networks. In: Infocom 2005, pp. 1526--1537. IEEE Press, Miami, Florida (2005)
10. Menascé, D.A., Kanchanapalli, L.: Probabilistic Scalable P2P Resource Location Services. ACM Sigmetrics Performance Evaluation Rev. 30 (2), 48--58 (2002)
11. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. In: Proc. ICSI Workshop on Design Issues in Anonymity and Unobservability. LNCS, vol. 2009, pp. 46--66. Springer-Verlag (2001)
12. Stoica, I., Morris, R., Karger, D., Kaashoek, M.E., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, In:ACM Sigcomm, San Deigo, (2001)
13. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content Addressable Network. In: ACM Sigcomm 2001, San Deigo, CA (2001)
14. Rowstron A., Druschel, P.: Pastry: Scalable, Distributed, Object Location and Routing for Large-scale Peer-to-Peer Systems. In: IFIP/ACM International Conference on Distributed System Platforms (Middleware), pp. 329--350, Heidelberg, Germany (2001)
[15] Zhao, Y., Kubiatowicz, J.D., Joseph, A.: Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical report, UCB/CSD-01-1141, Berkeley (2000)
16. Reynolds P., Vahdat, A.: Efficient Peer-to-peer Keyword Searching. In: International Middleware Conference, pp. 21--40, Rio de Janeiro (2003)
17. Tang, C., Xu, Z., Dwarkadas, S.: Peer-to-Peer Information Retrieval Using Self-organizing Semantic Overlay Networks, In: ACM SIGCOMM 2003, pp.175--186, Germany (2003)
18. Tang C., Dwarkadas, S.: Hybrid Global-local Indexing for Efficient Peer-to-peer Information Retrieval. In: Symposium on Networked Systems Design and Implementation (NSDI), pp. 211--224, San Francisco, California, (2004)
19. Bestavros, A.: Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time for Distributed Information Systems. In: International Conference on Data Engineering, pp. 180--187. New Orleans, Louisiana (1996)
20. IST-2001-38142-CONTEXT: http://context.upc.es
21. Caripe, W., Cybenko, G., Moizumi, K., Gray, R.: Network Awareness and Mobile Agent Systems. IEEE Communications Magazine 36 (7), 44--49 (1998)
22. Waxman, B.M.: Routing of Multipoint Connections. IEEE Journal on Selected Areas in Communications 6 (9), 1617--1622 (1988)
23. Barford P., Crovella, M.: Generating Representative Web Workloads for Network and Server Performance Evaluation. In: ACM SIGMETRICS 98, pp.151--160, Madison (1998)
24. Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web Caching and Zipf-like Distributions: Evidence and Implications. In:INFOCOM, pp.126--134, New York (1999)