A Distributed Search Service for Peer-to-Peer File Sharing in Mobile Applications

Christoph Lindemann and Oliver P. Waldhorst
University of Dortmund
Department of Computer Science
August-Schmidt-Str. 12
44227 Dortmund, Germany
http://www4.cs.uni-dortmund.de/~Lindemann/

Abstract

In this paper, we present the concept of Passive Distributed Indexing, a general-purpose distributed search service for mobile file sharing applications, which is based on peer-to-peer technology. The service enables resource-effective searching for files distributed across mobile devices based on simple queries. Building blocks of PDI constitute local broadcast transmission of query- and response messages, together with caching of query results at every device participating in PDI. Based on these building blocks, the need for flooding the entire network with query messages can be eliminated for most application. In extensive simulation studies, we demonstrate the performance of PDI. Because the requirements of a typical mobile file sharing application are not known – or even do not exist at all – we study the performance of PDI for different system environments and application requirements. We show that due to the flexible design PDI can be employed for several kinds of applications.

1. Introduction

The technical improvements in microelectronics increase the availability of computers in every part of human live. Low cost Laptops and Notebooks as well as powerful Personal Digital Assistants (PDAs), and even watches running fully featured operating systems [11] enable access to complex information at any time and from any place. Sophisticated wireless access technologies, e.g. UMTS, enable Internet access from almost everywhere in the world. However, even today 55% of all digital information resist not on public World Wide Web servers, but on personal computers. Equipped with interfaces for short- to medium-range wireless communication, e.g. based on the IEEE 802.11 [5] or Bluetooth [1] standards, mobile devices can form spontaneous self-organizing communication structures, so called Mobile Ad-Hoc Networks (MANETs [3]). These structures can be used to share data in common and innovative mobile applications. For example, data on PDAs can be shared by many persons based on full text or keyword search. Furthermore, communication-enabled MP3-Players can be used to exchange music in the MP3-format directly between devices. Growing spread of mobile DVD players and enhanced throughput of wireless communication may soon enable similar scenarios for digital movies. Perhaps one day electronic books will be able to search for related literature by title, keyword of even full text. This kind of innovative applications can take advantage of the bulk of information scattered across mobile devices. To enable effective data exchange in such environments, mechanisms are needed to discover a document of interest on remote devices. Document searching constitutes a building block of each file sharing system.

A promising approach for file sharing in wireline networks is based on peer-to-peer technology [13]. A peer-to-peer file sharing system is distributed, self-organizing and more or less decentralized. Several solutions for file sharing in wireline networks have been proposed. Today, the most popular solutions include the system formerly known as *Napster* [10], and the *Gnutella protocol* [4]. These approaches heavily depend on connectivity among peers and other network components, such as dedicated index servers, to provide searching capabilities. A first approach to port peer-to-peer technology to MANETs is *7DS* [12], which is designed to share documents among peers to enable Web-browsing without connection to the Internet.

In this paper, we introduce *Passive Distributed Indexing* (*PDI*), a general-purpose distributed search service for document exchange in mobile applications, which is based on peer-to-peer technology. PDI defines a set of simple messages for transmission of queries and responses. All messages are exchanged using local broadcast transmission. PDI supports forwarding of messages over several hops similar to techniques

known form routing in mobile ad-hoc networks [3]. Beside, PDI eliminates the need for flooding the entire network with query messages by maintaining an index cache at every device. This index cache is used to store pairs of keywords and document identifiers investigated in recently received reports. The index cache can be used to answer popular queries without forwarding them to a device that actually stores a matching document.

In extensive simulation studies, we investigate the performance of PDI for a set of different mobile computing environments and requirements of mobile applications. The computing environment defines system parameters such as the number of mobile devices participating in a distributed search service, their mobility model, and the transmission range of their interfaces for wireless communications. Application requirements specify the parameters of the workload generated by the applications running on top of PDI, i.e., the number of shared documents, the number of keywords occurring in queries and the time elapsing between two successive queries by the same device. PDI is designed to provide a general-purpose search service for mobile applications; therefore, we investigate the performance of PDI for different computing environments and application requirements.

This paper is organized as follows. Section 2 gives an overview over systems for wireline and wireless peer-to-peer file sharing. Section 3 describes the design and illustrates the basic operation of PDI. In Section 4, we present an in-depth performance study demonstrating the applicability of the introduced concepts. Finally, concluding remarks are given.

2. Related Work in Peer-to-Peer File Sharing

Most solutions for peer-to-peer file sharing support searching for documents by keywords. To provide this feature, an index is needed to map a keyword on a set of matching documents. Today's peer-to-peer file sharing systems use different approaches for providing such an index, which differ in the degree of centralization. A system in which all peers are equal and no searching functionality is centralized is called a *pure* peer-to-peer system, whereas systems using special devices for providing index functionality are called *hybrid* peer-to-peer systems [16].

As an example for a hybrid peer-to-peer system, *Napster* [10] employs a centralized index hosted by an index server at the Napster site. The index reports hits for documents that are stored at devices, which are currently connected to the index server. Centralized approaches such as Napster yield very fast queries, given a sufficient capacity of the central site. The fault tolerance and scalability of a centralized system is limited, because the operation of the entire system depends on connectivity and operation of the central index server. In particular, search operations can only be performed by devices currently connected to the index server. Furthermore, documents can only be found, if they are located on devices that are connected to the index server. Approaches used by hybrid peer-to-peer systems as Napster are not well suited for environments consisting of mobile devices equipped with interfaces for short- to medium-range wireless communication for two reasons. First, in general there are no devices providing extraordinary computing power and storage capacity, which is suitable to provide centralized index functionality. Second, even if there is a centralized index server, short-range wireless communication cannot guarantee that it is always reachable from all mobile devices. Opposed to Napster, the search service presented in this paper does depend neither on existence of nor on connectivity to a centralized index server.

As an example for a pure peer-to-peer system, the *Gnutella protocol* [4] implements fully distributed searching. Queries are broadcasted to all peers by a multi-hop flood algorithm that transmits them from device to device. Decentralized systems as Gnutella offer a high degree of fault tolerance, because the operation of the system does not depend on the connection to a dedicated device. However, the flood algorithm employed for searching requires each device to maintain a connection to at least one peer. Furthermore, flooding the entire network with query messages limits Gnutella's scalability [15]. Opposed to Gnutella, the approach presented in this paper does not need to discover and maintain connections between participating devices. Additionally, there is no requirement for flooding the whole network with query messages in most application scenarios, as shown in Section 4.

Due to growing interest in Napster and Gnutella, solutions for peer-to-peer file sharing achieve increasing importance in wireline networks. Intuitively, it is obvious to transfer the successful peer-to-peer technology to the context of self-organizing mobile systems (MANETs). However, the impact on file sharing technologies in mobile environments is still limited. A first approach to port peer-to-peer technology to mobile environments is *7DS* [12]. 7DS supports Web browsing by on-the-fly file sharing among peers that are not necessarily connected to the Internet. For queries, 7DS implements a multi-hop flooding algorithm combined with multicast delivery of queries. If a device holds a document matching a query, a report is transmitted to the host that has send out the query. Opposed to 7DS, the approach presented in this paper provides searching capabilities for documents outside the wireless transmission range without flooding the entire network with query messages. Instead, query results are locally broadcasted, cached at

several devices, and can be delivered on behalf of the device, which actually stores a matching document. Multi hop forwarding of queries as well as multi hop forwarding of query results may additionally be enabled. However, as Section 4 shows, forwarding messages for more than two hops is not necessary for most applications.

Several methods for reducing the network load generated by flooding a mobile ad hoc network with messages have been proposed. Most approaches are related to MANET routing issues, e.g. tree based approaches [9] or clustering algorithms [8]. Recent work transfers the ideas of clustering algorithms to peer-to-peer systems for mobile ad hoc networks [7]. Opposed to this previous work, our approach does not present a solution for implementing efficient network-wide flooding of messages, but eliminates the need for flooding in many application scenarios, as shown in Section 4.

3. Passive Distributed Indexing

We introduce *Passive Distributed Indexing* (*PDI*), a simple approach for file searching in mobile environments. PDI is intended to provide a general-purpose file search service, which can be used by several kinds of mobile applications running on top of it. Note that PDI does not specify how the transmission of a located document is performed. E.g. documents may be transmitted using sophisticated ad-hoc routing mechanisms, or even employing a wireless WAN, if available.

To implement PDI, each mobile device maintains a local *repository*, consisting of a set of files stored in the local file system. PDI provides search services for all documents in the repository. Each document can uniquely be identified by the path in the local file system together with a unique device identifier, e.g. the IP or MAC address of the mobile device. We refer to a unique pair of path and device identifier as *document identifier*.

Besides the repository, each mobile device implementing PDI maintains a local *index cache*. An index cache is a data structure that stores pairs of keywords and document identifiers for documents matching these keywords. PDI defines a mechanism to fill the index cache with pairs of keywords and document identifiers for documents located at remote devices. The index cache can be used by a mobile device to generate answers for queries, even if it does not store any matching document.

Building blocks of PDI constitute local broadcast transmission of query- and response-messages, together with caching all received query results in the local index cache. A query is defined by a query string, consisting of several keywords. Currently, we assume that a document matches a query only if it matches all keywords in the query, i.e. the keywords are connected by a Boolean AND operation. Query-messages are transmitted using local broadcast. To overcome the shortcomings of low wireless transmission range of the communication interfaces, queries may be forwarded for a predefined number of hops. Note that for many applications, forwarding messages by more than two hops is not necessary, as shown in Section 4. In the same way, response-messages are transmitted and may be forwarded from device to device in parts or as a whole. All mobile devices listen for broadcasted responses, even if they did not actually issue a query. When a response is received, all reported references to matching documents are added to the local index cache for all keywords contained in the original query. We limit the maximum number of entries in the index cache and replace old index cache entries by new ones in a least-recently-used fashion.

Passive distributed indexing implicitly replicates results for popular queries in index caches at several mobile devices, making heavy usage of locality investigated in the query stream [15]. Thus, in most cases popular queries can be answered by several mobile devices, even if they do not actually store any matching documents. To ensure index consistency, we expire documents after a certain timeout. When this timeout has elapsed, we remove all references relating keywords to this document from the index cache.

PDI defines two types of messages. A message of type QUE is used to broadcast queries. A QUE message contains a query string, which consists of one or more keywords. A message of type REP reports query results in response to a QUE message. It contains the query string from the QUE message as well as the unique identifiers of one or more documents matching the query. To enable message forwarding, each PDI message contains a field SRC, which contains the unique identifier of the mobile device that generated the original message, a field SEQ, containing a unique and strictly increasing sequence number for each mobile device, and a field TTL, indicating the number of hops a message may be forwarded. When a mobile device forwards a message as described below, TTL is decremented by one, while SRC and SEQ are preserved. By this way, forwarding a message more than one time can be prevented by storing the highest SEQ entry received from each source device, while limiting the total number of hops a packet may be forwarded.

When a device receives a QUE message, at first it searches for documents matching the query string in the local document repository. PDI does not specify how documents are matched against the query string. E.g., it is possible to match document titles, meta-descriptions stored with the documents or even perform a full text search. The matching process yields a set D_l of document identifiers for matching local documents.

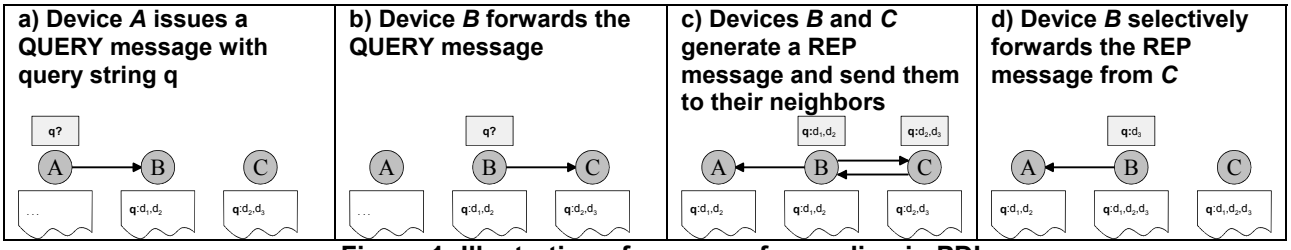


Figure 1: Illustration of message forwarding in PDI

After matching local documents, the device searches for entries matching the query string in the local index cache. Matching in the index cache is performed by matching each keyword contained in the query string against the keywords in the keyword / document identifier pairs stored in the index cache. Note that a remote document matches a query only if a keyword / document identifier pair is found for all keywords in the query in order to implement the AND operation. This matching process yields a set D_r of unique identifiers for matching remote documents. The joint set of local and remote document identifiers $D = D_l \cup D_r$ is broadcasted in a REP message, which contains the query string Q and all document identifier $d \in D$.

To compensate the shortcomings of the low transmission range introduced by interfaces for short- to medium-range wireless communication, PDI-enabled mobile devices can be configured to forward QUE messages, given that the TTL field of the received message does not equal zero. Messages of type QUE consist only of a query string containing a few keywords and therefore are small and simple. Thus, they can be forwarded without introducing unbearable load to the wireless network.

Opposed to QUE messages, a REP message contains a list of document identifiers, which may be of considerable size. Therefore, forwarding complete REP messages over several hops will significantly increase network load. To keep load low, PDI-enabled devices *selectively forward* REP messages. When a device receives a REP message containing a query string Q and a set of documents D, it searches the local index cache for all pairs (q,d) of keywords $q \in Q$ and document-identifiers $d \in D$. The index cache is updated for all pairs (q,d), which are not found in the cache. If the index cache reaches maximum size, old entries are replaced by new ones in a least-recently-used fashion. If the TTL field of the original message is not equal 0, a modified REP message is generated, which contains exactly the set of documents $D' = \{d'\}$, for which (q,d') was not found in the index cache. The REP message is broadcasted with decremented TTL field, as well as SRC and SEQ fields preserved from the original message. Selectively forwarding REP messages significantly reduces network traffic. The intuition behind selective forwarding is that if a part of a REP message matches a local cache entry, the message must have been received from a different direction than the original REP message that generated the entry in the index cache. Therefore, the result is already known roughly in the area in which the message will be forwarded, thus, the transmission is redundant and can be avoided.

Message forwarding in PDI is illustrated in Figure 1. We assume an environment with three mobile devices, referred to as device *A*, *B*, and *C*. All devices are drawn as filled gray circles. PDI messages are drawn as light-gray rectangles together with arrows indicating the direction of the wireless communication. White rectangles show a fraction of the local index cache at each device. In our example, we assume that the index caches of devices *B* and *C* each hold two document identifiers matching a query *q*. The document identifier d_2 is stored in both caches, the document identifiers d_1 and d_3 are only stored at device *B* and *C*, respectively. We assume that device *A* issues a QUE message for query *q* (Figure 1a). The message is transmitted using local broadcast, and therefore is only received by device *B*. *B* forwards the message using local broadcast again, so the query finally reaches device *C* (Figure 1b). As response to the query, each *B* and *C* generates a REP message, containing the entries of the local index cache that match *q* (Figure 1c). Similar to QUE messages, the REP messages are transmitted using local broadcast, so the message from *B* reaches *A* as well as *C*. All devices update the local index cache with the data contained in *B*´s message. The REP message from *C* cannot reach *A* directly and must be forwarded by *B*. To avoid unnecessary network traffic, *B* selectively forwards only the entries in the message, which are not already stored in its local index cache (Figure 1d). Not that at the end of the transmission, the content of all index caches holds identical entries for query *q* (not shown).

4. Performance Results

The performance of passive distributed indexing is affected by a wide range of parameters. We divide these parameters into *system parameters*, *application parameters*, and *protocol parameters*. System parameters describe the environment used for running PDI from the technical point of view. The system parameters include the number of mobile devices, the

transmission range of the wireless communication interfaces, and the mobility model, describing the movement of the mobile devices. Application parameters are parameters specific to the mobile application running on top of PDI. These parameters include the number of documents in each local repository, the number of keywords of interest in the application domain, and the distribution of keywords in the queries as well as the distribution of time elapsing between two successive queries by the same mobile device.

To keep PDI simple, we define only three parameters, which control the protocols operation. The first parameter is the *index cache size*, i.e., the storage contributed to PDI by each mobile device. Note that this parameter is related to the system running PDI, for the maximum index cache size cannot exceed the memory available at a mobile device. The second parameter is the *maximum time-to-live* value used for packet forwarding, i.e. the initial value of the TTL field in a PDI message at its original source. By modifying the maximum time-to-live, PDI can be configured to run in different forwarding modes. The third parameter is the *document timeout*, which specifies how long entries for a particular document should be kept in the index cache. In the remainder of this paper, we assume that the time between changes to the document repositories is much bigger than simulation time. Therefore, we neglect the impact of document modifications and index cache consistency.

PDI has been designed as a general-purpose search service for usage in a set of different system environments and application domains; therefore, we have performed extensive simulation studies to evaluate the performance of PDI in different scenarios. In the remainder of this section, we first describe the experimental setup used to model these scenarios. After that, we illustrate how the protocol parameters can be used to compensate the impact of system and application parameters on protocol performance. At last, we illustrate the performance of PDI in the initial warm-up phase.

4.1. Experimental Setup

To evaluate the performance of PDI in different system environments and for different application requirements, we conduct simulation studies based on the *Network Simulator ns-2* [6] with mobile and wireless extensions. ns-2 can be used to simulate mobile devices and wireless communication using a protocol stack including UDP, IP an MAC layer. We developed an ns-2 application implementing PDI as described in Section 3. The application can be configured to support different values for index cache size and maximum time-to-live. An instance of the PDI application is attached to each simulated mobile device, using the UDP/IP protocol stack and a MAC layer according to the IEEE 802.11 standard for wireless communication [5]. We run the wireless communication interface in the ad-hoc mode using unreliable local broadcast. As radio-propagation model, we chose two-ray ground propagation [14]. Different transmission powers of the communication interface were selected to model different transmission ranges.

Besides a PDI-enabled application and a protocol stack for wireless communication, we need to define a *mobility model* to capture the movements of the mobile devices as well as a *workload model* to capture the queries generated by the application running on top of PDI. For the mobility model, we assume an area of 1000 m × 1000 m. N_{Nodes} mobile devices move in this area according to the *random waypoint mobility model* [3], which is commonly used to model the movement of individual pedestrians. According to this mobility model, each device starts at a location chosen uniformly at random inside the considered area and moves to another randomly chosen location. The speed of the device is chosen uniformly at random form $[0, s_{\max}]$, where the top speed s_{max} may be different in different experiments. When a device reaches its destination, it rests for a period T_{hold}, before it continues its movement to the next randomly chosen destination at randomly chosen speed.

For the workload model, we assume that every device participating in PDI provides search functionality on N_{Docs} documents stored in the local repository. Each document d_i, $0 \le i < N_{Docs}$, matches a randomly chosen set $K_i \subset \{0, N_{Keywords} - 1\}$ of keywords, where $N_{Keywords} > N_{Docs}$ is the number of keywords of interest in the application domain. For simplicity, we assume the cardinality $|K_i|$, which denotes the number of keywords matching document d_i, to be calculated by $K_i = \lceil N_{Keywords} - i / N_{Keywords} \rceil$, i.e., there is a linear correlation between the document number and the number of matching keywords. Each mobile device sends out queries for a randomly chosen keyword in randomly chosen intervals using QUE messages as described in Section 3. The time between

Table 1. Parameters used in experiments.

Parameter	Value
Total simulation time	6 h
Simulation area	1000 m × 1000 m
Maximum speed s_{max}	1.5 m/sec
Holding time T_{hold}	50 sec
Number of devices N_{Nodes}	64
Transmission Power	17.6125 mW
Inter-request time T_{Think}	120 sec
Number of keywords $N_{Keywords}$	512
Slope of query distribution β	0.9
Number of documents N_{Docs}	16

two successive queries is chosen randomly according to an exponential distribution with mean T_{Think}. For simplicity, we restrict our experiments to queries consisting of a single keyword. The keyword contained in a query is chosen randomly according to a Zipf-like distribution [2], i.e., for the probability for choosing keyword k_j holds $\Pr(k = k_j) \cong j^{-\alpha}$, for $0 \le \alpha \le 1$. Note that depending on the value of α, the query stream can bear significant locality.

In various simulation runs, we found that the performance of PDI is rather independent of the system parameters defined by the mobility model, i.e. maximum speed s_{max} and the holding time T_{hold} as defined by the random waypoint model. Furthermore, it is independent of application parameters modeling the mean time between two successive queries, T_{Think}, and the number of keywords used in queries, $N_{Keywords}$. Therefore, we fix these parameters to the values shown in Table 1. All other parameters may be variable in some experiments. If not stated otherwise, these parameters are set to default values, which are also shown in Table 1.

As primary performance measure in our simulation studies, we define *query hit rate*. To calculate query hit rate, we count the number N_{rep} of unique document identifiers in REP messages received by the device that issued a particular query. Query hit rate is calculated as the quotient N_{rep}/N_{all}, where N_{all} denotes the number of all documents matching a query. Note that query hit rate differs from hit rate considered in studies of Web caching systems, which measures the number of documents that can be found in the cache when requested by a unique identifier. In general, there are many documents matching a particular query, therefore query hit rate might satisfy a user even if it is much lower than 50%. In future work, we will investigate other performance measures, e.g. system response time.

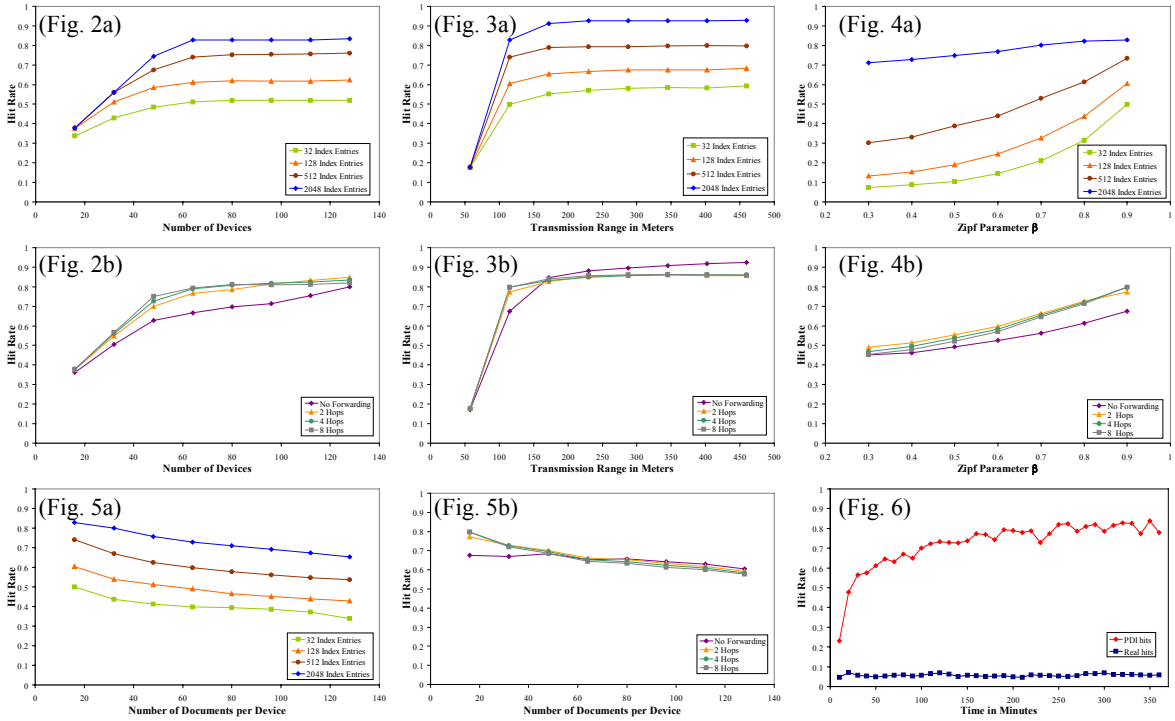
4.2. Sensitivity to System Parameters

In a first experiment, we investigate the sensitivity of PDI to the number of mobile devices participating in the searching application. The results of this performance study are illustrated in Figure 2a. We found that for a small number of devices participating in the system, the size of the local index caches has only limited impact on the performance of PDI. Furthermore, we found that for small index cache sizes, the impact of a growing number of mobile devices on the performance of PDI is rather limited. This can be explained by the fact that every device contributes new documents to the search service, so that small index caches cannot hold entries for all documents matching a query. For growing index cache sizes, the performance of PDI increases with a growing number of devices. We conclude that index caches may be small if the number of devices participating in the system is limited. To gain the largest possible benefit of the variety of documents contributed to the search service by a large number of devices, sufficient index cache size should be provided.

Similar to the impact of index cache size, the impact of message forwarding is limited in systems with low density of mobile devices, as shown in Figure 2b. Forwarding messages for more than eight hops gains only marginal improvements in query hit rate for such systems. However, for a growing number of mobile devices, query hit rate grows faster in systems with message forwarding enabled than in non-forwarding systems. In environments with about 64 mobile devices per square kilometer, configuring the system for packet forwarding can improve query hit rate by almost 20%, because with high probability a forwarded message will reach several devices. This benefit vanished when the number of devices grows further, because a high number of devices carry information of interest around in the local index cache, which replaces message-forwarding adequately. We conclude from Figure 2b that message forwarding is useful in system environments showing a medium density of mobile devices, while in systems with extreme device densities message forwarding should be disabled to avoid unnecessary network traffic.

In a second experiment, we investigate the sensitivity of PDI to the transmission range of the wireless communication interfaces used by the mobile devices. The results of this study are shown in Figure 3a. For a transmission range below 100 Meters, PDI does not gain sufficient query hit rates despite of the size of the individual index caches. This can be explained by the fact that in most cases broadcasted QUE messages are received only by a small number of devices. With a transmission range around 115m, PDI gains sufficient query hit rates, given that a reasonable size of the index cache is provided. We conclude that for short-range communication devices, e.g. Bluetooth [1], the number of participating devices must be high to enable the effective employment of PDI.

As another interesting result, we found that for high communication range PDI with message forwarding disabled gains best performance, see Figure 3b. This investigation can be explained by the fact that forwarding responses to uncommon queries over great distances will fill index caches with junk entries, replacing entries for popular queries. We conclude that in environments with medium wireless transmission ranges, message forwarding should be enabled to benefit from PDI. When transmission range is high, message forwarding should be disabled to avoid spamming index caches with junk entries.



Figures 2 - 6. Performance results for PDI for different system and application scenarios

4.3. Sensitivity to Application Parameters

As shown in Figure 4a PDI is extremely sensitive to locality in the request stream for small sizes of the index cache. For large cache sizes, e.g. 2048 entries, PDI can achieve a hit rate of more than 70% despite of the locality. We conclude that for applications offering no significant locality in the request stream, the sizes of the index cache must be chosen adequate. Figure 4b shows the impact of query locality on different options for message forwarding. We find that for low-locality request streams, all forwarding options achieve performance within a 5% interval. For higher locality, PDI gains significant performance improvements from packet forwarding. Note that in most cases 2-hop forwarding performs equal to or even better than forwarding messages over four or even more hops. This indicates that PDI in most cases is able to resolve a query from inside the direct neighborhood of a device. We conclude that 2-hop message forwarding should be enabled in applications offering a high degree of locality in the request stream.

In a further experiment, we investigated the sensitivity of PDI to the number of documents stored at each device. We investigate in Figure 5a that the hit rate of PDI decreases linearly with a growing number of documents for a given index cache size. Furthermore, we conclude from this figure, that the hit rate of PDI increases with index size in a log-like fashion, for increasing the size of the index cache by factor four only gains a linear increase in hit rate. The same investigation has been made for Web caching systems with regard to cache size and cache hit rate, respectively. It has been shown that this behavior can be explained if a Zipf-like request distribution is assumed [2].

These investigations may lead to the conclusion that more sophisticated forwarding strategies rather than increasing index cache sizes should be employed to improve the hit rate of PDI. We investigated the impact of message forwarding in a further experiment. The result is shown in Figure 5b. We notice that the impact of document forwarding increases PDI Hit rate by more than 10% if only a small number of documents is contributed to the search service by each device. As an interesting fact, again PDI configured for 2-hop forwarding gains comparable performance to forwarding over four or even more hops. However, with growing number of documents, more sophisticated forwarding strategies loose ground in comparison to no forwarding at all. Again, this behavior can be explained by the fact that broadcasting query responses over many hops will fill caches with junk entries. We conclude that for applications, in which only a few documents are contributed by each device, performance of PDI can be improved by configuring the system for 2-hop forwarding.

4.4. Transient behavior

As a last experiment, we investigated the transient behavior of PDI for a single application scenario. In Figure 6, we plot the cumulative query hit rate achieved by PDI in a 10-minute period over total

simulation time. For comparison, we also illustrate the rate of hits reported form devices, which actually hold a matching document. This hit rate, denoted as *real hit rate*, indicates the performance of a query scheme using single-hop point-to-point communication without caching results in local index caches. We investigate that even ten minutes after simulation start, the hit rate of PDI is about 10% higher than the real hit rate. The PDI hit rate steadily increases until it reaches an average of about 75%. We conclude form this performance study that there is no need for a sophisticated mechanism to perform an initial warm-up. After short time, PDI will fill local index caches in a self-organizing manner, and therefore will be able to answer queries providing hit rates as stated above.

5. Conclusion and Future Work

We presented the concept of Passive Distributed Indexing, a general-purpose distributed document search service for mobile file sharing applications. PDI is based on peer-to-peer technology, i.e., PDI does not require any centralized infrastructure for providing searching capabilities. Building blocks of PDI constitute local broadcast transmissions of query- and response-messages together with caching results of popular queries at every device participating in PDI. Using these building blocks, PDI eliminates the need of flooding the whole network with query messages. Furthermore, it can be configured to support different system environments and application requirements by adjusting three parameters, namely the size of the local index caches, the maximum number of hops that a message will be forwarded, and the document timeout used for expiry of cache entries.

In extensive simulation studies, we demonstrated the usefulness of PDI for different systems and applications. We conclude from this performance studies, that for systems with a high density of mobile devices and applications bearing low locality in the query stream, sufficient index cache sizes should be provided. In systems with a medium density of mobile devices and medium wireless transmission ranges, 2-hop packet forwarding should be enabled, whereas packet forwarding should be disabled if either the number of mobile devices or their transmission range is high. A large number of documents contributed to PDI by each mobile device can be handled by providing sufficient index size. Finally, PDI provides an initial filling of index caches in very short time, eliminating the need of sophisticated warm-up mechanisms.

Future work on PDI will lead in three main directions. First, we will investigate the impact of document modifications on the performance of PDI and design appropriate mechanisms for providing index caches consistency. Second, we will evaluate the performance of PDI considering more sophisticated workload models, e.g. workloads consisting of location depended queries. Third, we will develop a prototype implementation of PDI and test it in a mobile e-learning environment at University of Dortmund as a prove-of-concept.

References

[1] C. Bisdikian, An Overview of the Bluetooth Wireless Technology. IEEE Communications 39(12), 86-94, 2001.

[2] L.Breslau, P. Cao, L. Fan, G. Phillips, and S. Schenker, Web-Caching and Zipf-like Distributions: Evidence and Implications. *Proc. 18th Annual Joint Conference of the IEEE Computer and Communications Societies* (*IEEE INFOCOM 99*), New York, NY, 1999.

[3] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva, A Performance Comparison of Multi-Hop Wireless Ad-Hoc Network Routing Protocols. *Proc. 6th ACM/IEEE Int. Conf. on Mobile Computing and Networking* (*MobiCom 98*), Dallas, Texas, 1998.

[4] clip2, The Gnutella Protocol Specification v0.4, 2001. Available from http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

[5] I. S. Department. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Standard 802.11-1997*, 1994

[6] K. Fall and K. Varadhan, The ns-2 manual. Technical Report, Berkeley University, 2002. Available from http://www.isi.edu/nsnam/ns/ns-documentation.html

[7] R. Gold and C. Mascolo, Use of Context-Awareness in Mobile Peer-to-Peer Networks. *Proc. 8th IEEE Workshop on Future Trends of Distributed Computing Systems* (*FTDCS 01*), Bologna, Italy, 2001.

[8] C. Lin and M. Gerla, Adaptive Clustering for Mobile Wireless Networks. *IEEE Journal on Selected Areas in Communications* **15**(7), 1265-1275, 1997.

[9] H. Lim and C. Kim, Flooding in Wireless Ad Hoc Networks, *Computer Communications* **24**(3-4), 353-363, 2001.

[10] Napster Hompage. http://www.napster.com

[11] C. Narayanaswani, N. Kamijoh, M. Raghunath, T. Inoue, T. Cipolla, J. Sanford, E. Schlig, S. Venkitewaran, D. Guniguntala, V. Kulkarani, and K. Yamazaki, IBM's Linux Watch: The Challenge of Miniaturization. *IEEE Computer* **35**(1), 33-41, 2002.

[12] M. Papadopouli and H. Schulzrinne, Effects of Power Conservation, Wireless Coverage and Cooperation on Data Dissemination among Mobile Devices. *Proc. ACM Symp. on Mobile Ad Hoc Networking & Computing* (*MobiHoc 2001*), Long Beach, CA, 2001.

[13] M. Parameswaran, A. Susarla, and A. Whinston, P2P Networking: An Information-Sharing Alternative. *IEEE Computer* **34**(7), 31-38, 2001

[14] T. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 1996.

[15] K. Sripanidkulchai. The popularity of Gnutella queries and its implications on scalability. *Proc. O'Reilly Peer-to-Peer and Web Services Conference*, 2001

[16] B. Yang and H. Garcia-Molina, Comparing Hybrid Peer-to-Peer Systems. *Proc. 27th Int. Conf. on Very Large Data Bases* (*VLDB 2001*), Rome, Italy, 561-570, 2000