26. Epidemic Data Dissemination for Mobile Peer-to-Peer Lookup Services

Christoph Lindemann, Oliver P. Waldhorst (University of Dortmund)

26.1 Motivation and Background

Building efficient lookup services for the Internet constitutes an active area of research. Recent issues concentrate on building Internet-scale distributed hash tables as building block of Peer-to-Peer systems, see e.g., [505], [575]. Castro et al. proposed the VIA protocol, which enables location of application data across multiple service discovery domains, using a self-organizing hierarchy [111]. Recently, Sun and Garcia-Molina introduced a partial lookup service, exploiting the fact that for many applications it is sufficient to resolve a key to a subset of all matching values [581]. The paper discusses various design alternatives for a partial lookup service in the Internet. However, none of these papers consider distributed lookup services for mobile ad-hoc networks.

In MANET, lookup services can be implemented using either unstructured or structured Peer-to-Peer networks as described in Chapters 24.4.1 and 24.4.2, respectively. However, such approaches put some requirements on the MANET environment: (1) The MANET must provide a high degree of connectivity, such that a given node can contact each other node at any time with high probability. (2) The nodes in the MANET must exhibit low mobility in order to minimize the required number of updates of routing tables and other structures. Typically, both structured and unstructured approaches will perform poorly in scenarios with low connectivity and high mobility. This chapter descibes an approach for building a Peer-to-Peer lookup service that can cope with intermittent connectivity and high mobility. The approach builds upon the observation by Grossglauser and Tse, that mobility does not necessarily hinder communication in MANET, but may support cost-effective information exchange by epidemic dissemination [262].

As a first approach to epidemic information dissemination in mobile environments, Papadopouli and Schulzrinne introduced Seven Degrees of Separation (7DS), a system for mobile Internet access based on Web document dissemination between mobile users [470]. To locate a Web document, a 7DS node broadcasts a query message to all mobile nodes currently located inside its radio coverage. Recipients of the query send response messages containing file descriptors of matching Web documents stored in their local file caches. Subsequently, such documents can be downloaded with HTTP by the inquiring mobile node. Downloaded Web documents may be distributed to other

R. Steinmetz and K. Wehrle (Eds.): P2P Systems and Applications, LNCS 3485, pp. 435-455, 2005.

nodes that move into radio coverage, implementing an epidemic dissemination of information.

Using a related approach, Goel, Singh, Xu and Li proposed broadcasting segments of shared files using redundant tornado encoding [253]. Their approach enables nodes to restore a file, if a sufficient number of different segments have been received from one or more sources. In [351], Khelil, Becker, Tian, and Rothermel presented an analytical model for a simple epidemic information diffusion algorithm inspired by the SPIN-1 protocol [290]. Both systems implement a push model for information dissemination. That is, shared data is advertised or even actively broadcasted without a node requesting it. Hanna, Levine, and Mamatha proposed a fault-tolerant distributed information retrieval system for Peer-to-Peer document sharing in mobile ad hoc networks [275]. Their approach distributes the index of a new document to a random set of nodes when the document is added to the system. The complete index of a document, i.e., all keywords matching it, constitutes the smallest unit of disseminated information. Recently, Small and Haas proposed an epidemic approach for collecting information in a hybrid network consisting of mobile nodes and fixed infostations [568]. Their architecture, denoted as Shared Wireless Infostation Model (SWIM), actively transfers information among wireless nodes on each contact, until information is unloaded to one of the infostations.

All approaches [470], [253], [351], [275], and [568] are tailored to specific applications like file-sharing and collecting information. Recall that global lookup operations are a building block of many distributed applications. Thus, such applications require a general-purpose lookup service. In the remainder of this chapter, we present a general-purpose distributed lookup service for mobile applications that uses epidemic information dissemination.

26.2 Passive Distributed Indexing

26.2.1 Overview

In this section, we describe the concept of a lookup service denoted Passive Distributed Indexing (PDI, [389]), that supports resolution of application-specific keys to application-specific values. As building block, PDI stores index entries in index caches maintained by each mobile device. Index entries are propagated by epidemic dissemination, i.e., they are exchanged between devices that get in direct contact, similar to the spread of an infections disease. By exploiting node mobility, such contacts occur randomly between arbitrary devices. Using the information disseminated in this way, PDI can resolve most queries locally without sending messages outside the radio coverage of the inquiring node.

By local resolution of queries, PDI effectively reduces network traffic for the resolution of keys to values for applications possessing a sufficiently high

degree of temporal locality in their query streams. Thus, deployment of PDI is attractive for various mobile applications. For example, since queries in a Peer-to-Peer (P2P) file sharing system follow a Zipf-like distribution [570], [355], PDI can implement a distributed search algorithm for such application, that can be complemented by an effective transport protocol for subsequent transfers of located files, e.g., [189]. Moreover, since it has been shown that user queries for Internet search engines possess high temporal locality [631], PDI can support searching in Web-based information portals or even the World Wide Web without connection to an Internet search engine. Further mobile applications effectively supported by PDI include instant messaging (IM) and mobile city guides. In an IM application a lookup service resolves user identifier to the terminal the user is currently logged on and the current presence state of a user, e.g., available, busy, or away. In a city guide a user queries for names and locations of places of interest, e.g., sights, hotels, or restaurants. Both mobile applications are likely to exhibit high locality in the query behavior, too.

In the remainder of this chapter, we consider a MANET consisting of several mobile nodes, e.g. mobile users equipped with notebooks or PDAs and wireless network interfaces as illustrated in Figure 26.1. All mobile nodes collaborate in a shared application that uses a distributed lookup service. Radio coverage is small compared to the area covered by all nodes, e.g., less than 5% of the covered area. Subsequently, we assume IEEE 802.11x in the ad hoc mode as underlying radio technology [315]. However, we would like to point out that that the approaches described in this chapter could be employed on any radio technology that enables broadcast transmissions inside a nodeÂs radio coverage.

26.2.2 Basic Concept

PDI implements a general-purpose lookup service for mobile applications. In general, PDI stores index entries in the form of pairs (k, v). Keys k and values v are both defined by the mobile application. For example, in case of a file sharing application, keys are given by keywords derived from the file name or associated meta-data. Values are given by references to files in form of URIs. Opposed to Distributed Hash Table systems like [505], [575], PDI does neither limit the number of keys matching a value nor the number of values matched by a key. However, some mechanisms implemented in PDI require that a value is unique in the system. That is, it is only added to the system by a single node. This can be easily achieved by extending the application specific value v by a unique node identifier for a node n. For example, the node identifier i_n may be derived from the node's IP address or the MAC address of the radio interface. For ease of exposition, we will abbreviate the unique value given by (v, i_n) pairs just by v.

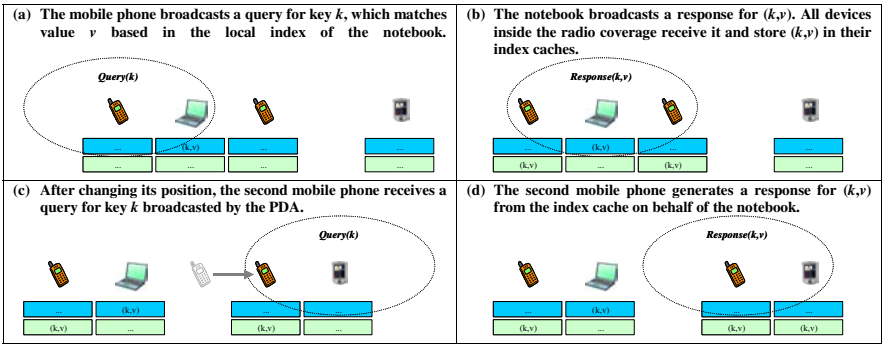


Fig. 26.1: Illustration of epidemic information dissemination with PDI

A node n may contribute index entries of the form (k, v) to the system by inserting them in a local index. In Figure 26.1, the local index is drawn as the first box below each mobile device. We refer to such an index entry as *supplied*. The node n is called the *origin node* of an index entry. For example, the notebook shown in Figure 26.1 is the origin node of the index entry (k, v). A key k matches a value v, if (k, v) is currently supplied to the PDI system. Each node in the system may issue queries in order to resolve a key k to all matching values v_i (see Figure 26.1a). A node issuing a query is denoted as *inquiring node*.

Query messages are sent to the IP limited broadcast address 255.255.255.255 and a well-defined port, using the User Datagram Protocol UDP. Using the IEEE 801.11 ad hoc mode, all nodes located inside the radio coverage of the inquiring node receive a query message. Each of these nodes may generate a response message. A response message contains the key from the query and all matching values from either the local index or a second data structure called index cache. To enable epidemic data dissemination, PDI response messages are sent to the IP limited broadcast address 255.255.255.255 and a well-defined port, too. Thus, all mobile nodes within the radio coverage of the responding node will overhear the message (Figure 26.1b). Not only the inquiring node but also all other mobile nodes that receive a response message extract all index entries and store them in the index cache (see Figure 26.1b). In Figure 26.1, index caches are drawn as the second box below mobile devices. Index entries from the index cache are used to resolve queries locally, if the origin nodes of matching values reside outside the radio coverage of the inquiring node (see Figures 26.1c and 26.1d). Obviously, the index cache size is limited to a maximum number of entries adjusted to the capabilities of the mobile device. The replacement policy least-recently-used (LRU) is employed if a mobile device runs out of index cache space. By generating responses from index caches, information is disseminated to all other nodes that are in direct contact, similar to the

spread of an infectious disease. By exploiting node mobility, index entries are disseminated within the network without costly global communication. However, information is only transferred when actively requested by a node. In fact, PDI builds and maintains an index distributed among mobile node of the MANET in a passive way.

26.2.3 Selective Forwarding for Extending Radio Coverage

Recall that all PDI messages are send to the limited broadcast address and received by all nodes located inside the radio coverage of the sender. Depending on the transmission range of the wireless network interfaces, this may considerably limit the number of nodes that receive a message. PDI includes a flooding mechanism that controls forwarding based on the content of a message. The mechanism is illustrated in Figure 26.2. Query messages are flooded with a TTL with value TTL_{query}, which is specified by the inquiring node. For detecting duplicate messages, each massage is tagged with a unique source ID and a sequence number as described above. We will show in Section 26.4 that $TTL_{query} \leq 2$ yields a sufficient performance in most scenarios. Thus, PDI communication remains localized despite of the flooding mechanism.

Similarly to query messages, response messages are forwarded with time-to-live TTL_{query}. Recall that the payload of query messages consists of a few keys. Thus, query messages are small and may be flooded without significantly increasing network load (see Figure 26.2a and 26.2b). In contrast, response messages can contain numerous values that may each have a considerable size, depending on the application using PDI. Therefore, flooding of complete response messages will significantly increase network load, even if the scope of flooding is limited to two hops. For the cost-efficient flooding of response messages, PDI incorporates a concept called selective forwarding. That is each node that receives a response message will search the index cache for each index entry contained in the message (see Figure 26.2d). If an entry is found, the node itself has already sent a response for this query with high probability (e.g., as shown in Figure 26.2c). Therefore, forwarding this index entry constitutes redundant information. Using selective forwarding, each relay node removes all index entries found in its local index cache from the response message, before the message is forwarded (see Figure 26.2e).

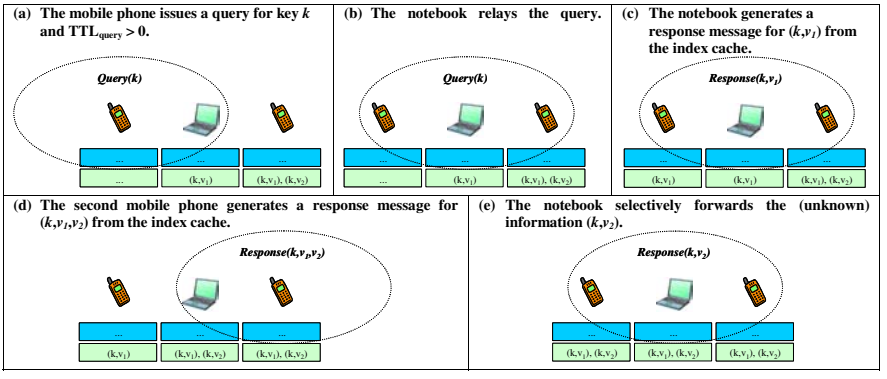


Fig. 26.2: Message forwarding in PDI

26.3 Consistency Issues

26.3.1 Configurable Value Timeouts for Dealing with Weak Connectivity and Node Failures

The basic concept of PDI as described in Sections 26.2.2 and 26.2.3 does not take into account intermittent connectivity and spontaneous departures of nodes; circumstances under which all information previously supplied by a node expire. Examples of these cases include node failure or nodes leaving the area covered by the system. In such cases, an implicit invalidation mechanism can achieve cache coherency. Timeouts constitute a common concept to implement implicit invalidation in several distributed applications, as they can assure cache consistency without the need to contact the source of the cached information. PDI defines the concept of value timeouts to approximate the most recent information about the state of an index entry at the origin node. Value timeouts limit the time any index entry (k, v) with a given value v will be stored in an index cache. By receiving a response from the origin node of (k, v), the corresponding value timeout will be reset. Let $age\,((k, v))$ be the time elapsed since (k, v) has been extracted from a response message generated by its origin node. We define the age a_v of value v as $a_v = min_k\,(age\,((k, v)))$, i.e., the time elapsed since the most recent response message of this kind was received. If at a node holds $a_v > T$ for the given timeout value T, all pairs (k, v) are removed from its index cache. PDI implements only one timeout per value v rather than an individual timeout for each index entry (k, v). This is motivated by the observation that in most applications modification of an index entry (k, v) for a given v indicates a substantial change of the value. Subsequently, all other index entries (k', v) are likely to be influenced. For example, in a file sharing system a pair $(keyword_i, URI)$ is removed when the file specified by URI is withdrawn

from the system. Thus, all other pairs $(keyword_j, URI)$ also become stale. Note that depending on the application the concept of value timeouts can be easily extended to incorporate individual timeout durations T_v for each value v. Such duration may be included in a response message generated by the origin node. For ease of exposition, we assume in the remainder of this paper a global timeout value T for all values in the system.

To determine the current age of a value, an age field is included in the response message for each value. This age field is set to zero in each response from the origin node. When receiving a response message, a node n extracts the age of each value and calculates the supply time s_v. That is the time of generating a response for this value by the origin node. Assume that the response message contains age a_v, then s_v is determined by $s_v = c_n - a_v$, where c_n denotes the local time of node n. s_v is stored in the index cache together with v. Note that v might already be present in the index cache with supply time s'_v. The copy the index cache might result from a more recent response by the origin node, i.e., $s_v < s'_v$. Thus, in order to relate the age of a value to the most current response from the origin node, the supply time is updated only if $s_v > s'_v$. When a node generates a response for a cached index entry (k, v), it sets the age field for each value v to $a_v = c_n - s_v$. Note that only time differences are transmitted in PDI messages, eliminating the need for synchronizing clocks of all participating devices.

26.3.2 Lazy Invalidation Caches for Dealing with Data Modification at the Origin Node

Additional to the scenarios described above, a node produces stale index entries by modifying information. That is the case when an index entry is removed from the local index. One way to handle such modification of information is to wait until the timeouts of the values in the stale index entries elapse. Depending on the application and the timeout duration T, this straightforward solution may cause severe inconsistency, especially if T is large. A more effective way to handle information modification in distributed applications constitutes the explicit invalidation by control messages. Examples of explicit invalidation schemes include the invalidation of cached memory blocks in distributed shared memory (DSM) systems, or the invalidation of documents in web caches. To achieve consistency, the origin node of an item sends invalidation messages to exactly those nodes that are caching this item. In DSM systems, the origin node of a shared page sends invalidation messages to all nodes sharing this page. In web caching systems, the origin server of a web document sends invalidation messages to each web cache that holds a copy of the document. Note that both mechanisms require that the origin node knows of all copies of an item and is connected to all sharers. Unfortunately, in a mobile environment consisting of nodes with limited resources, connectivity

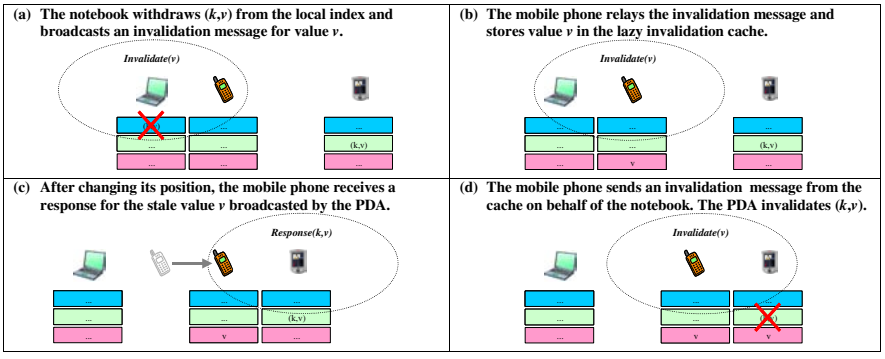


Fig. 26.3: Epidemic dissemination of invalidation messages using lazy invalidation caches

of nodes cannot be guaranteed nor directories for all cached copies of a shared item can be maintained. To address these constraints in mobile systems, PDI defines the concept of lazy invalidation caches implementing explicit invalidation of values by epidemic dissemination of invalidation messages. As basic idea of PDIÂs explicit invalidation mechanism, a node removes all index entries (k, v) from the index cache when it receives an invalidation message for value v. Flooding with a TTL with value TTL_{inv} is a straightforward way to propagate invalidation messages. Unfortunately, in mobile systems even a multi-hop connection between two nodes frequently does not exist. Subsequently, stale index entries remain in the index caches of nodes that are not reached by the invalidation message. Note that these index entries will be redistributed in the system due to the epidemic dissemination. We have shown that even repeated flooding of invalidation messages does not significantly reduce the number of hits for stale index entries [389].

This observation is consistent with [162], which reports that deleted database items ÂresurrectÂ in a replicated database environment due to epidemic data dissemination. In [162], a solution is proposed that uses a special message to testify the deletion of an item, denoted as death certificate. Death certificates are actively disseminated along with ordinary data and deleted after a certain time. In contrast, we propose a more or less passive (or ÂlazyÂ) approach for the epidemic dissemination of invalidation messages, which is illustrated in Figure 26.3. For the initial propagation of an invalidation message by the origin node, we rely on flooding as described above (Figure 26.3a). Each node maintains a data structure called lazy invalidation cache, which is drawn as a third box below the mobile devices in Figure 26.3. When a node receives an invalidation message for a value v it does not only relay it, but stores v in the invalidation cache (Figure 26.3b). Note that an entry for v is stored in the invalidation cache, regardless if the node stores any index entry (k, v) for v in the index cache. Thus, every node will contribute to the

propagation of invalidation messages, so that distribution of information and invalidation messages is separated. To enable the epidemic dissemination of the invalidation message, a node scans the invalidation cache for all values contained in an overheard response message (Figure 26.3c). If a value v is found, the node will generate an invalidation message, because the hit in the invalidation cache indicates that the index cache of a nearby node contains a stale entry (Figure 26.3d). The invalidation message is not flooded through the entire network, but only with a certain scope TTL_{inv} similar to forwarding query and response messages as described in Section 26.2.3. A node that receives a cached invalidation message for value v will store v in the invalidation cache, and remove all index entries (k, v) from the index cache. Additionally, the node checks whether it has recently received hits for v in response to an own query, which must also be invalidated and may not be passed to the application running on top of PDI.

As the index cache size, the invalidation cache size is limited to a fixed number of values and LRU replacement is employed. We have shown in [389] that setting the invalidation cache size to a fraction below 20% of the index cache size achieves a sufficient reduction of false hits assuming a reasonable rate of data modification. Note that LRU replacement does neither guarantee that an invalidation cache entry is kept until all stale index entries are invalidated, nor that it is removed after a certain time, inhibiting a node indefinitely from restoring a value it has invalidated once. Increasing the invalidation cache size solves the first problem, though, doing so amplifies the second problem. To avoid this tradeoff, maintaining the supply time of invalidation messages similar to the supply time of values as described by Section 26.3.1 yields an efficient mechanism to decide whether a result for a value is more recent than an invalidation message.

26.4 Performance Studies

26.4.1 Simulation Environment

To evaluate the performance of the PDI and the proposed consistency mechanisms, we conduct simulation experiments using the network simulator ns-2 [198]. We developed an ns-2 application implementing the basic concepts of PDI, selective forwarding, value timeouts, and lazy invalidation caches as described in Sections 26.2 and 26.3. An instance of the PDI application is attached to each simulated mobile node, using the UDP/IP protocol stack and a MAC layer according to the IEEE 802.11 standard for wireless communication. Recall that PDI can be configured by the four parameters shown in Table 26.2. As goal of our simulation studies, we will show that PDI can be configured to the demands of different applications by adjusting these parameters. Therefore, we have to define detailed models of the system in which PDI is deployed.

The performance of PDI is affected by several characteristics of the mobile environment. We separate these characteristics into system characteristics and application characteristics. System characteristics describe the mobile environment. These characteristics include the density of mobile nodes as well as their arrivals and departures, the transmission range of the wireless communication interfaces, and the mobility model, describing the movement of the mobile nodes. Application characteristics are specific to the mobile application using PDI. These characteristics include the number of values supplied by each mobile node, the matching between keys and values, the distribution of keys in the queries as, the distribution of pause times between two successive queries by the same mobile node, and the validity of a value.

For modeling system characteristics, we assume that N mobile users equipped with mobile devices participate in a mobile application. Assuming the two-ray ground reflection model, we configure the transmission power of the wireless interface to provide a radio-coverage with a radius R. We assume that the mobile nodes move in an area of size A according to the random waypoint (RWP) mobility model [92], which is commonly used to mimic the movement of pedestrians. According to this mobility model, each node starts at a location chosen uniformly at random inside the simulation area and moves to another randomly chosen location. The speed of the node is chosen uniformly from $(0, v_{max}]$, where the maximum speed v_{max} may be different in different experiments. When a node 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. Note that the assumption of movement at pedestrian speed might constitute a conservative assumption for disseminating information by exploiting mobility.

In the remainder of this paper, we will use a workload model inspired by a Peer-to-Peer file sharing application. To capture the characteristics of this application, we assume that the application defines a set of keys $\mathcal{K}$ with cardinality $K = |\mathcal{K}|$. We associate each key with its popularity rank, i.e., $\mathcal{K} = 1, \ldots, K$, where key 1 is the most frequently requested key. Additionally, the application defines a set of values $\mathcal{K}$ with cardinality $V = |\mathcal{V}|$. We use an arbitrary numbering of values, i.e., $\mathcal{V} = 1, \ldots, V$. The values are equally distributed among the mobiles nodes, i.e., each node contributes the same number of values. To determine the matching between keys and values we define a selection function. The selection function $w_{select}(k)$ denotes the probability that a key k matches a given value v for $k = 1, \ldots, K$ and $v = 1, \ldots, V$. Note that the selection function is independent of v. Following [634], the selection function can be well represented by an exponential distribution with mean $1/\alpha$:

$$w_{select}(k) = \alpha e^{-\alpha k}$$

To determine which keys are used in queries, we define a query function with probability density function (pdf) $w_{query}(k)$ that denotes the probability

that a query is for a given key k. As shown in [570], [355], query popularity in Peer-to-Peer file sharing systems follows a Zipf-like distribution, i.e., the query function for $k = 1, \ldots, K$ is given by:

$$w_{query}(k) \sim k^{-\beta}$$

To determine the timestamps of queries issued by the mobile users, we define a pause function $w_{pause}(t)$ that denotes the probability that a node pauses for the time t between two successive queries, $0 \leq t \leq \infty$. We assume that pause times are exponentially distributed with mean $1/\lambda$, i.e., the pause function is given by:

$$w_{pause}(t) = \lambda e^{-\lambda t}$$

Up to this point, the model of system and application characteristics is still not suitable for evaluating the consistency mechanisms presented in Section 26.3, since it does not incorporate node departures and data modifications. To take node departures into account, we assume that the time t between two successive arrivals or departures, respectively, follows an exponential distribution with pdf $w_{arrival}(t)$, $0 \leq t \leq \infty$. We adjust the arrival and departure rates such that approximately dN for $0 \leq d \leq 1$ nodes will arrive or depart during the considered simulation time T, respectively. Thus, we define $\mu = dN/T$ and assume that he arrival function for $0 \leq t \leq \infty$ is given by a is given by an exponential distribution with mean $1/\mu$:

$$w_{arrival}(t) = \mu e^{-\mu t}$$

Then, the number of arrivals or departures in simulation time T is Poisson distributed with a parameter given by $\mu T = dN$ (i.e., it has mean dN), matching our assumption. Arriving nodes enter the system with empty index and invalidation caches and contribute values that are not used by any node that is already in the system. Departing nodes do not send invalidation messages for supplied data, i.e., all index entries supplied by a departing node expire.

To consider value modifications, we assume that each value expires exactly once during a simulation of length T. Thus, we define an expiration function with pdf $w_{expire}(t)$ that denotes the probability that a value expires after a time t, $0 \leq t \leq T$, as continuous uniform distribution:

$$w_{expire}(t) = \frac{1}{T}$$

If not stated otherwise, all parameters defining system and application characteristics are chosen as shown in Table 26.1. The default settings of the PDI protocol parameters are shown in Table 26.2.

We choose performance measures to evaluate the recall and the coherence of the results delivered by PDI. Recall is measured by the hit rate HR, i.e., $HR = H_F/K_F$ for H_F denoting the number of up-to-date hits and K_F

Parameter	Value
Total simulation time T	7200 s
Simulation area A	1000 m × 1000 m
Number of devices N	64 or 80
Maximum speed v_{max}	1.5 m/s
Rest Period T_{hold}	50 s
Transmission range R	115 m
Number of keys K	512
Number of values V	16
Parameter of selection function α	1/100
Parameter of query function β	0.9
Parameter of pause function λ	1/120
Fraction of arriving / departing nodes d	0.3

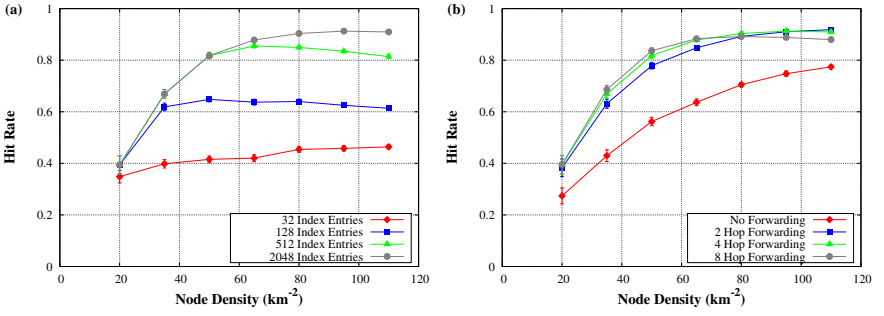
Table 26.1: Default values for simulation parameters

the total number of all up-to-date matching values currently in the system. Coherence is measured by the stale hit rate SHR, i.e., $SHR = H_S/(H_S + H_F)$, where H_S denotes the number of stale hits returned on a query. Note that stale hit rate is related to the information retrieval measure *precision* by $precision = 1 - SHR$.

In all experiments, we conduct transient simulations starting with initially empty caches. For each run, the total simulation time is set to 2 hours. To avoid inaccuracy due to initial warm-ups, we reset all statistic counters after a warm-up period of 10 min. simulation time. Furthermore, we initialize positions and speed of all nodes according to the steady state distribution determined by the random waypoint mobility model [85] to avoid initial tran-

PDI Protocol Parameter	Default Value
Index cache size	2048 entries
Selective forwarding TTL	4 hops
Value timeout	1000 s
Invalidation cache size	128 entries

Table 26.2: Default values for PDI configuration parameters



(a) (b)

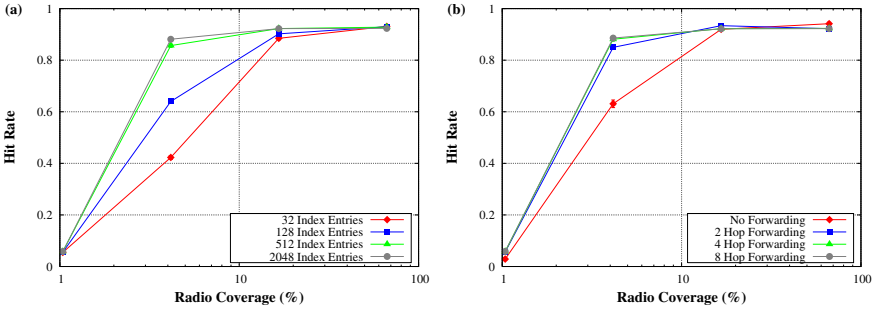
Fig. 26.4: Recall vs. system size for (a) different index cache sizes and (b) different numbers of forwarding hops

sients. For each point in all performance curves, we performed 100 independent simulation runs and calculated corresponding performance measures at the end of the simulation. In all curves 99% confidence intervals determined by independent replicates are included.

26.4.2 Sensitivity to System Characteristics

To evaluate the performance of PDI with respect to the system characteristics, we do not consider consistency aspects in our first experiments. That is, we do not model value expiration, assuming that all values are valid during a simulation run. Furthermore, we do not consider node arrivals and departures, assuming that the community of PDI users is static during the simulation time. The impact of consistency issues is evaluated in Section 26.4.4.

In a first experiment, we investigate the sensitivity of PDI to the node density, i.e., the number of mobile nodes moving in the simulation area and participating in the application that uses PDI. The results of this performance study are shown in Figure 26.4. Figure 26.4 (a) plots the hit rate as a function of node density for different sizes of the index cache. We find that for a small number of nodes, the size of the local index caches has only a limited impact on the performance of PDI. In these scenarios, a low node density limits the epidemic information dissemination due to a lack of contacts between the nodes. For an increasing node density, we observe three different effects. First, an increasing number of contacts between the nodes significantly foster epidemic information dissemination for sufficient large index caches (i.e., 128 index cache entries or more). Second, connectivity increases with node density. Thus, selective forwarding increases hit rate even for small index caches (i.e., 32 index entries). Third, the hit rate for large caches decreases

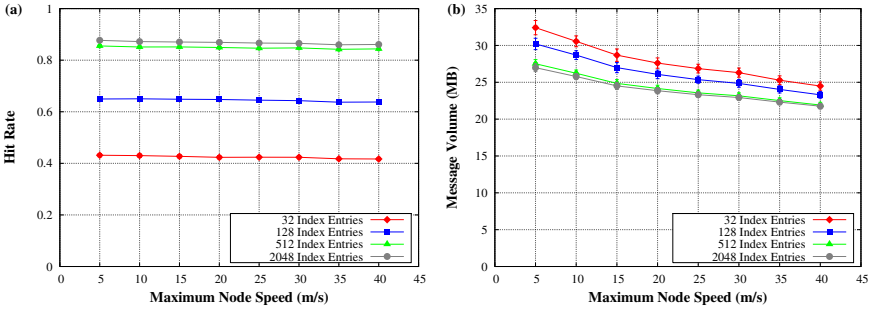


(a) (b)

Fig. 26.5: Recall vs. radio coverage for (a) different index cache sizes and (b) different numbers of forwarding hops

when the node density passes a certain number of nodes. To understand this effect, recall that the overall number of values in the system increases with node density because each node contributes additional values to the lookup service. Furthermore, the keys in queries are selected according to a Zipf-like selection function. Due to the heavy tailed nature of this function, responses to a large number of queries must be cached in order to achieve high hit rates. Thus, the hit rate decreases even for large caches when the number of index entries for popular queries exceeds cache capacity and the epidemic dissemination of data decreases. We conclude from Figure 26.4 (a) that epidemic data dissemination requires a sufficient node density. To gain most benefit of the variety of values contributed to the lookup service by a large number of nodes, sufficient index cache size should be provided. In properly configured systems with a reasonable node density, PDI achieves hit rates up to 0.9.

Similar to the impact of index cache size, the impact of message forwarding is limited in systems with a low node density, as shown in Figure 26.4 (b). Forwarding messages for more than four hops yields only marginal improvements in the hit rate due to limited connectivity. However, for an increasing node density, the hit rate grows faster in systems with message forwarding enabled than in non-forwarding systems, as increasing connectivity favors selective forwarding. In environments with about 64 nodes, configuring the system for packet forwarding can improve hit rate by almost 30%. Nevertheless, non-forwarding systems benefit from growing node density as it fosters epidemic information dissemination. Thus, the benefit of selective forwarding with $TTL_{query} \geq 4$ hops becomes negligible, if the number of mobile nodes becomes larger than 64. In these scenarios, forwarding messages over multiple hops will decrease the variety of information stored in the index caches, because forwarded results replace other results in a high number of caches. Thus, fewer different results are returned from the caches for succes-



(a) (b)

Fig. 26.6: (a) Recall vs. mobility for different index cache sizes and (b) message volume vs. mobility for different numbers of forwarding hops

sive queries. We conclude from Figure 26.4 (b) that message forwarding is beneficial in system environments showing a medium node density, while in systems with high node density message forwarding should be disabled.

In a second experiment, we investigate the sensitivity of PDI to the transmission range of the wireless communication interfaces used by the mobile nodes. The results of this study are shown in Figure 26.5. Figure 26.5 (a) shows hit rate as a function of transmission range for different cache sizes. For a transmission range below 100 meters, i.e., a radio coverage of about 1% of the simulation area, PDI does not gain sufficient hit rates despite index cache size. Here, in most cases broadcasted query messages are received only by a small number of nodes. Consistent with the results shown Figure 26.4 (b), given a reasonable size of the index cache and a transmission range of 115 meters (i.e., a radio coverage of about 4% of the considered area) PDI achieves sufficiently high hit rate for Peer-to-Peer search queries. For larger transmission ranges, most queries will reach the origin nodes of index entries (k, v) for a query v. Thus, PDI does not benefit from caching index entries. We conclude from Figure 26.5 (a) that for short-range communication devices the number of participating devices must be high to enable the effective employment of PDI, whereas for long-range communication, the system does not significantly benefit from PDI.

Figure 26.5 (b) shows hit rate as a function of transmission range for different values of TTL_{query}. We find that message forwarding has no impact in systems with small transmission ranges, while system with medium transmission ranges heavily benefit from forwarding. As another interesting result, we find that for high transmission ranges PDI with message forwarding disabled gains best performance. Here, unnecessary forwarding of PDI messages will result in a substantial number of collisions of wireless transmissions, as confirmed by the examination of ns-2 trace files. These collisions reduce the

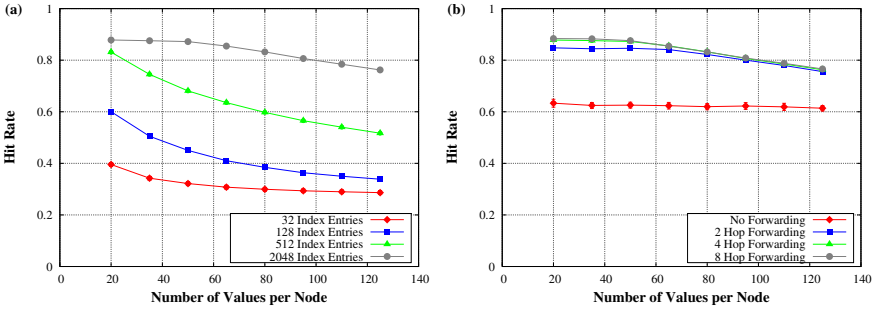
number of response messages that reach the enquiring node. We conclude from Figure 26.5 (b) that in environments with medium wireless transmission ranges, message forwarding should be enabled to benefit from PDI. If transmission range is high, message forwarding should be disabled to avoid congestion of the wireless medium.

As final system parameter, we investigate the sensitivity of PDI to node mobility in Figure 26.6. Figure 26.6 (a) plots the hit rate as a function of maximum node speed. Increasing device velocity from pedestrian speed to the speed of cars in an urban environment, we find that hit rate is almost independent of node speed for the considered application scenario. Recall that according to the pause function a node sends queries in an average interval of 120s, while the holding time defined by the random waypoint mobility model is 50s. That is, with high probability the node will change position between two successive queries, so that epidemic information dissemination is large despite of node mobility. Nevertheless, PDI benefits from increasing mobility in terms of total volume of transmitted PDI messages as shown in Figure 26.6 (b). As increasing mobility fosters epidemic information dissemination to some extend, message volume is reduced up to 25% depending on node mobility. As information is more equally distributed across the simulation area for high mobility scenarios, most queries can be resolved locally, i.e., from nearby nodes. Results from more distant nodes will be suppressed by the selective forwarding mechanism, resulting in substantial savings in message volume. We conclude from Figure 26.6 that PDI performs well for a reasonable degree of mobility, while high mobility increases the benefit of selective forwarding.

In an experiment not shown here, we employ the Reference Point Group Mobility Model (RPGM) [303], which organizes nodes into groups that move together across the simulation area. [303] argues that group mobility is more realistic than individual mobility in mobile scenarios. Obviously, group mobility reduces epidemic information dissemination, since nodes primarily get in touch with members of their own group. The experiments show that PDI performance is reduced by at most 10% for scenarios with few nodes (and few groups, respectively). However, when the number of nodes exceeds 80, simulation results converge against the results for the RWP mobility model, since contacts among groups frequently occur. We conclude from these experiments that PDI performs well even for more realistic mobility models if a reasonable node density is provided.

26.4.3 Sensitivity to Application Characteristics

To evaluate the performance of PDI with respect to the characteristics of the application runing on top of it, again, we do not consider consistency issues originating from node departures and data modifications. As first application

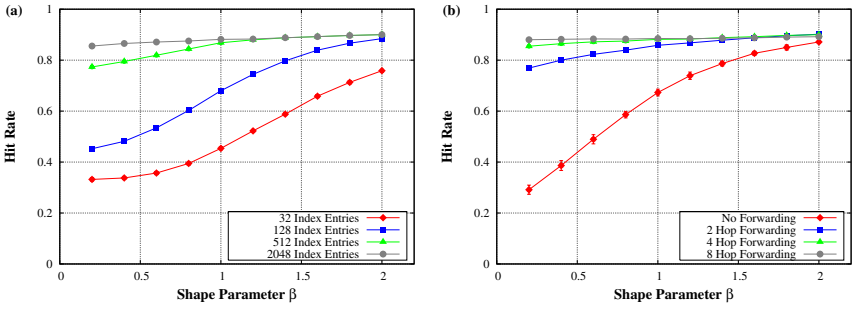


(a) (b)

Fig. 26.7: Recall vs. shared data for (a) different index cache sizes and (b) different numbers of forwarding hops

characteristics, we consider the overall data volume managed by PDI. Note that data volume depends on the number of values supplied by each node, which may be very different for different mobile applications. For example, for a Peer-to-Peer file sharing application, we assume that the number of values is up to 100, where each node in an instant messaging application only contributes a single value. For searching a Web portal or even the entire WWW, the overall number of values might be considerable larger, but does not depend on the number of nodes. To capture the characteristics of these different applications, we perform a sensitivity study with respect to the data volume managed by PDI.

Figure 26.7 plot hit rate as a function of the number of values contributed by each node. We investigate in Figure 26.7 (a) that the hit rate of PDI decreases with a growing number of values for all considered index cache sizes. Furthermore we observe that PDI is most sensitive to the number of values for medium index cache sizes (i.e., 128 and 512 index cache entries). As observed in Figure 26.4 (a), index caches of these sizes cannot provide high hit rates for an increasing amount of data, whereas a large cache can handle the data easily. Recall that the performance of small caches shows low sensitivity to the data provided to the system, as the hit rate is primarily determined by selective forwarding. To illustrate the impact of selective forwarding at larger index cache sizes, we investigate the performance of different forwarding options for an index cache size of 2048 index entries and an increasing number of contributed values in Figure 26.7 (b). We find that setting $TTL_{query} = 2$ improves hit rate by 40%, while setting $TTL_{query} \geq 4$ improves hit rate by less than 5% compared to $TTL_{query} = 2$. We conclude from Figure 26.7 that large index caches should be provided and selective forwarding enabled to handle large numbers of values.



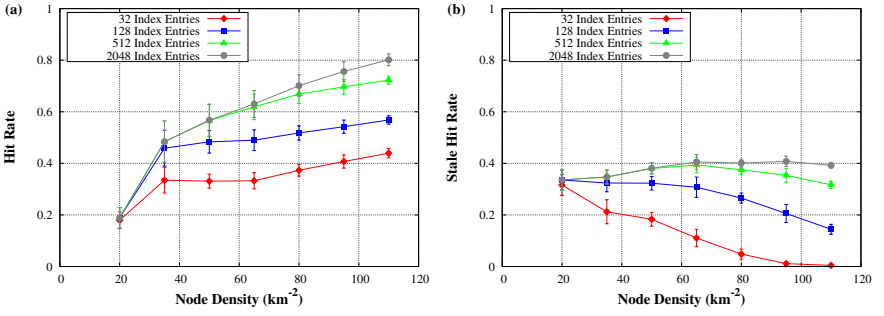
(a) (b)

Fig. 26.8: Recall vs. locality for (a) different forwarding options and (b) different numbers of forwarding hops

In an experiment not shown here, we found that PDI is almost insensitive to the shape parameter of the matching function α, which determines the distribution of the number of values matched by a key. As all caching-based mechanisms, PDI performance depends on locality in the query stream. Recall that the selection function is given by a Zipf like distribution with shape parameter *beta*. If $0 < \beta \leq 0.5$, the locality in the query stream is low, whereas $\beta > 1$ indicates high locality. Hit rate as a function of the shape parameter β is shown in Figure 26.8 for different index cache sizes and forwarding options, respectively. Figure 26.8 (a) reveals that PDI is extremely sensitive to locality in the query stream for small sizes of the index cache (i.e., 32 and 128 index cache entries). For large cache sizes (i.e., 512 and 2048 entries), PDI can achieve a hit rate of more than 0.75 despite of the locality. Figure 26.8 (b) shows that PDI is most sensitive to query locality if no selective forwarding is enabled. With $TTL_{query} \geq 2$, however, PDI can effectively combine results from several index caches and effectively cope with low locality. We conclude from Figure 26.8 that sufficient index cache size should be provided and 2 hop forwarding enabled if the application using PDI provides low locality.

26.4.4 Impact of Consistency Mechanisms

In the final simulation experiments, we investigate the coherence of index caches maintained by PDI with and without the invalidation mechanisms presented in Section 26.3. These performance curves are shown in Figures 26.9 and 26.10. Figure 26.9 (a) plots hit rates as a function of node density for different sizes of the index cache. Comparing Figure 26.9 (a) with Figure 26.4 reveals that the hit rate is reduced due to cache space occupied by stale index entries. Stale hit rates as a function of node density is plotted

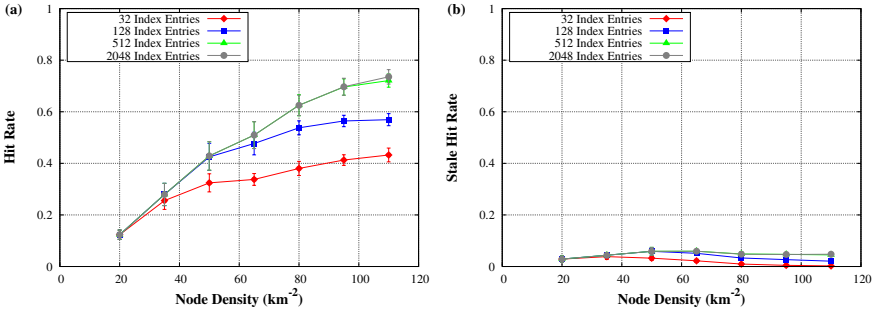


(a) (b)

Fig. 26.9: (a) Recall and (b) coherency vs. system size without invalidation

in Figure 26.9 (b). We find that without invalidation the stale hit rate may reach 0.4. For smaller index cache sizes, the stale hit rate decreases with node density. Jointly considering Figures 26.9 (a) and (b) reveals that for an increasing node density the stale hit rate drops rapidly at the point when the growth of the hit rate slows down. Looking closer at the index caches in these scenarios, we find that the content of the caches is highly variable. Thus, stale index entries are removed early from the caches. We conclude from Figure 26.9 (b) that large caches yield a high amount of stale hits if no invalidation mechanism is used. In contrary, small index caches naturally reduce stale hits, while they fail to provide high hit rates as shown in Figure 26.9 (a). This evidently illustrates the need for invalidation mechanisms in order to achieve both high hit rates and low stale hit rates.

In a last experiment, we investigate the performance of an integrated approach combining both value timeouts and lazy invalidation caches to take into account both weak connectivity and information modification. In further experiments presented in [389], we found that suitable configuration of value-timeouts reduces the stale hit rate due to intermittent connectivity and node failure by 75%. Furthermore, lazy invalidation caches of moderate size reduce stale results due to data modification by more than 50%. Thus, we fix the duration of the value timeout to 1000s and the invalidation cache size to 128 entries, since these parameters achieved best performance for the considered scenario [389]. Figure 26.10 (a) plots the hit rate versus node density. We find that hit rate is reduced mostly for small systems due to invalidations of up-to-date index entries by value timeouts. This leads to a decrease of at most 20%. The performance of index cache sizes of both 512 and 2048 is equal because a large cache cannot benefit from long-term correlations between requests due to the short timeout. For growing number of nodes, the hit rate converges towards results without an invalidation mechanism as shown in Figure 26.9 (a).



(a) (b)

Fig. 26.10: (a) Recall and (b) coherency vs. system size for hybrid invalidation

As settlement for the reduction of hit rate, the stale hit rate is significantly reduced compared to a system without invalidation. As shown in Figure 26.10 (b), the stale hit rate is below 5% for all considered index cache sizes. We conclude from Figure 26.10 that the integrated approach comprising of the introduced implicit and explicit invalidation mechanisms can effectively handle both spontaneous node departures and modification of information. In fact, for large index caches, the stale hit rate can be reduced by more than 85%. That is, more than 95% of the results delivered by PDI are up-to-date.

26.5 Summary

In this chapter, we described a distributed lookup service for mobile applications denoted Passive Distributed Indexing (PDI). As key concept, PDI employs epidemic dissemination of (key, value) pairs among mobile devices. To foster information dissemination for devices with limited radio transmission range, PDI incorporates a bandwidth-efficient message relaying mechanism denoted selective forwarding. To provide coherent results in environments with frequently changing data, PDI incorporates implicit invalidation by configurable value timeouts and explicit invalidation by lazy invalidation caches.

In an extensive simulation study, we illustrated the recall and the result coherence achieved by PDI for different system and workload characteristics. We found that with the suitable configuration of index cache size and selective forwarding PDI can achieve hit rates of more than 90% despite of system and application characteristics. Considering result coherence, we found that a combination of both invalidation mechanisms provides more than 95% up-to-date results. All presented simulation results are based on a workload model inspired by a Peer-to-Peer file sharing application. However, we have

shown that PDI can cope with different application characteristics using appropriate configurations. Thus, PDI can be employed for a large set of mobile applications that possess a sufficiently high degree of temporal locality in the request stream, including Web-portal and Web search without connection to the search server, instant messaging applications, and mobile city guides.

In recent work, we have developed a general-purpose analytical performance model for epidemic information dissemination in mobile and hoc networks [390]. Currently, we are employing this modeling framework to optimize PDI protocol parameters for selected mobile applications. Based on the results, we are adopting PDI for developing software prototypes of a mobile file sharing system, a mobile instant messaging application, and disconnected Web search.