Design and Quantitative Analysis of Protocols for Epidemic Information Dissemination in Mobile Ad Hoc Networks

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Universität Dortmund
am Fachbereich Informatik

von

Oliver P. Waldhorst

Dortmund
2005

Tag der mündlichen Prüfung:	24.11.2005
Dekan:	Prof. Dr. Bernhard Steffen
Gutachter:	Prof. Dr.-Ing. Christoph Lindemann,
	Prof. Dr. Peter Buchholz

Abstract

This thesis explores the field of protocols for epidemic information dissemination (EID) that constitute a novel way to implement peer-to-peer communication in Mobile Ad Hoc Networks (MANET). Such protocols exploit device mobility and transmit information when mobile devices encounter, similar to the spread of an infectious disease among individuals. The contribution of this thesis is three-fold. As first contribution, a general-purpose distributed lookup service for MANET is presented. The lookup service, denoted as Passive Distributed Indexing (PDI), builds upon epidemic dissemination of index information. It differs from previously proposed EID systems in three major aspects: (1) PDI can be employed for key-to-value lookups in arbitrary mobile applications. (2) PDI explicitly considers the limited availability of memory resources on mobile devices by using buffers of finite capacity and employing Least Recently Used (LRU) replacement when the buffer capacity is exceeded. (3) PDI provides consistency mechanisms for ensuring coherence of disseminated information. A comprehensive simulation study illustrates that PDI is well-suited to implement consistent lookup operations in various mobile applications.

As second contribution, this thesis presents a design study of mechanisms for disseminating frequently changing presence information in a mobile instant messaging system. The study extends previous work in three major aspects: (1) It introduces sustained consistency as a novel measure for the coherence of the disseminated presence information. (2) It evaluates different alternative approaches for disseminating presence information with respect to sustained consistency and control traffic and identifies an epidemic approach as the method of choice. (3) It proposes the System for Presence information Exchange by Epidemic Dissemination (SPEED). As illustrated in a comprehensive performance study, SPEED outperforms an approach based on optimized flooding of presence information with respect to sustained consistency and control traffic.

As third contribution, this thesis presents a novel stochastic modeling approach for EID systems in MANET that differs from previous approaches in three major aspects: (1) It explicitly considers the spread of multiple data items as well as buffers with finite capacity and LRU replacement. (2) The approach conducts steady-state analysis rather than transient analysis. (3) The approach does not require offline simulations to determine model parameters. It is shown that an EID system with finite buffer can be modeled by discrete-time Markov chains with a state space that grows exponentially with the number of nodes and data items in the EID system. Since a numerical steady-state solution with state-of-the-art methods is computationally infeasible, an approximate solution approach is presented and applied for modeling the Seven Degrees of Separation (7DS) EID system. A comparison of the approximate results to the results of a detailed simulation study shows an excellent agreement. Furthermore, a comprehensive performance study provides key insight into the steady-state behavior of 7DS.

Acknowledgment

At this place, I would like to thank all people that contributed to the completion of this thesis by ideas, assistance, and support. First of all, I would like to thank my thesis advisor Prof. Dr.-Ing. Christoph Lindemann, for introducing me to scientific work and teaching me the art of writing papers. For their technical support, a special thanks goes to my college Christian Lambert for his work on the optimization of the SPEED system, the master's student Dennis Bölting for his support with the simulation of the PDI system, and all other master's students for the fruitful discussions regarding simulation models and prototypes of systems presented in this thesis. A special thanks goes to all current and former colleagues of the *Computer Systems and Performance Evaluation* group for a relaxed atmosphere that made working most of the time a pleasure.

Last but not least, I would like to thank my parents Brigitte and Jürgen, to whom I dedicate this thesis. Without there support in hard times, regardless if professional or private, all this would not have been possible.

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die durch Ideen, Hilfe und Unterstützung zur Fertigstellung dieser Arbeit beigetragen haben. Zuerst danke ich meinem Doktorvater Prof. Dr.-Ing. Christoph Lindemann, der mich an wissenschaftliche Arbeitsweisen heranführte und die Kunst des Verfassens von wissenschaftlichen Veröffentlichungen lehrte. Für ihre fachliche Unterstüzung geht mein spezieller Dank an meinen Kollegen Christian Lambert für seine Arbeiten zur Optimierung des SPEED Systems, an meinen Diplomanden Dennis Bölting für seine Unterstützung bei den Simulationen des PDI Systems und an alle anderen Diplomanden für die angeregten Diskussionen im Zusammenhang mit Simulationen und Prototypen zu in dieser Arbeit vorgestellten Systemen. Ein spezieller Dank geht an alle derzeitigen und ehemaligen Kollegen des Fachgebiets *Rechnersysteme und Leistungsbewertung* für eine entspannte Atmosphäre, die das Arbeiten zur meisten Zeit zu einem Vergnügen machte.

Zu guterletzt möche ich meinen Eltern Brigitte und Jürgen danken, denen ich diese Arbeit widme. Ohne ihre Unterstüzung in schwierigen Zeiten, egal ob beruflich oder privat, wäre all dies nicht möglich gewesen.

Contents

Chapter 1

Introduction

1.1 Motivation

In recent years, the continued miniaturization of mobile computing devices driven by technical achievements in microelectronics has rapidly increased the popularity of mobile computing. Low cost notebooks as well as powerful Personal Digital Assistants (PDAs), smart phones, and even watches running fully featured operating systems [NKR+02] enable pervasive processing of complex information. Sophisticated wireless access technologies, e.g., the Universal Mobile Telecommunication System (UMTS, [KAL+05]), provide Internet access from almost any place. However, a huge share of all digital information is stored not on public World Wide Web servers, but on personal computing devices.

Equipped with interfaces for short- to medium-range wireless communication, e.g., based on the IEEE 802.11 [IEE97] or Bluetooth [Bis01] standards, mobile devices can form spontaneous self-organizing communication structures, so called Mobile Ad Hoc Networks (MANET [MAN]). In MANET, all mobile devices may act as routers and forward packets on behalf of other devices. Thus, even if devices that are not located in each others transmission range they can communicate using a multi-hop route of intermediate devices. Some efforts have been made to support the "ad hoc" deployment of an Internet-like routing infrastructure in MANET, see, e.g., [PRD03], [JMH04], [CJ03]. Building upon such technologies, MANET can be used to share data in innovative mobile applications. For example, data on PDAs can be shared based on full text or keyword search. Furthermore, communication-enabled MP3-Players can be used to exchange music in the MP3-format directly from device to device. The increasing popularity of mobile DVD players and enhanced throughput of wireless communication technologies may soon enable similar scenarios for

digital videos. Perhaps one day electronic books will be able to search for related literature by title, keyword or even full text. Such innovative applications can make use of the bulk of information scattered across mobile devices. However, such applications require mechanisms for exchanging information among mobile devices that are tailored to MANET.

Since MANET lack devices with extraordinary computational resources that can accomplish server-like tasks, classical distributed Internet applications based on the client/server paradigm cannot be deployed in MANET. In recent years, the Peer-to-Peer (P2P) paradigm has gained increasing popularity as an approach for designing distributed applications [PSW01]. P2P systems are composed of peers with similar capabilities that act as both client and server. Typical P2P system are inherently scalable, since they exhibit no central entity that may become a bottleneck. Furthermore, similar to MANET, P2P systems are self-organizing due to the lack of a central control instance. This common property makes the P2P paradigm appealing as a design principle for MANET applications.

Most P2P systems for the Internet maintain logical connections between peers on the application layer for transferring data and control messages [LCP+05]. The logical application layer network defined by these connections is denoted as *overlay network* or simply *overlay*. We recall the functionality of overlay-based P2P systems in Section 2.1. Since bandwidth does not constitute a bottleneck in the Internet, algorithms for constructing P2P overlays in general do not consider the underlying topology of the physical network, e.g., as shown in [RIF02]. However, when a P2P system is deployed in a MANET, a mismatch between the overlay and the physical topology leads to an inefficient usage of scarce bandwidth resources. Furthermore, device mobility may frequently change the physical routes that are used by an overlay connection. In the worst case, intermittent connectivity may cause the disconnection of peers in the overlay network. We show in Section 2.2 that constructing and maintaining the network routes used by the overlay connections generates considerable overhead due to activities by the MANET routing protocol.

Cross-layer information transfer is an approach to solve the problem of a poor matching between the physical topology of the network and the overlay topology maintained by the P2P system. With cross-layer information transfer, information is exchanged between protocols and applications that run on different layers of the TCP/IP protocol stack. For example, the MANET routing protocol running on the Internet layer might pass information about the hop distance of other peers that run a certain P2P application to the instance of this P2P application running on the application layer of the local peer. Subsequently, the P2P application can adopt the

overlay connections to the current state of the physical network. Recent activities try to standardize mechanisms and interfaces for cross-layer information transfer in MANET, e.g., [CGMT03], [CMTG04]. Furthermore, P2P systems that employ cross-layer information transfer for deploying optimized file sharing systems in MANET have been proposed, e.g., [KLW03], [TCG05], or even provide mechanisms to construct application-aware networks routes that can be used by general-purpose overlays, e.g., [SGN03]. We recall the functionality of these systems in Section 2.3. P2P systems using cross-layer information transfer exhibit best performance in MANET with a high device density and low device mobility. In such networks, stable multi-hop routes exist that can be used by overlay connections with tolerable overhead. However, in networks with a low device density and high device mobility, intermittent connectivity or even network partition occur frequently and hinder the establishment of overlay connections even if they closely match the physical network topology.

A key observation in MANET is that mobility does not necessarily hinder communication, but may support cost-effective information exchange. In fact, Grossglauser and Tse showed that mobility increases the capacity in MANET for delay-tolerant applications [GT02]. This result uses intermediate devices as messengers to transfer a data packet from a source to a destination. Due to device mobility the messenger will encounter the destination on the long run and can subsequently deliver the packet. This concept can be generalized to information dissemination by transferring arbitrary data between devices on each encounter. Algorithms that implement such transfers disseminate information similar to the spread of an infectious disease among individuals and are therefore denoted as *epidemic algorithms*.

Many systems that implement information dissemination by epidemic algorithms, or for short *epidemic information dissemination* (*EID*), have been proposed in scientific literature. A first example constitutes the *Seven Degrees of Separation* (*7DS*) system proposed by Papadopouli and Schulzrinne, which implements Web browsing by Web document sharing between mobile devices [PS01]. The concept of 7DS was enhanced for more efficient document sharing by Goel, Singh, Xu, and Li [GSXL02]. Small and Haas proposed an epidemic algorithm denoted as *Shared Wireless Infostation Model* (*SWIM*) for collecting biological information in a hybrid network consisting of mobile nodes (i.e., whales) and fixed infostations (i.e., buoys) [SH03]. Häner, Becker, and Rothermel present the *Negotiation-based Ad Hoc Data Dissemination Protocol* (*NADD*) for disseminating sensor data between mobile devices, e.g., carried by firefighters [HBR03]. Hanna, Levine, and Mamatha proposed a fault-tolerant distributed *P2P information retrieval* (*P2P-IR*) system for document sharing in

MANET [HLM03]. Recently, Luo, Eugster, and Hubaux introduced the *Probabilistic Lightweight Group Communication System for Ad Hoc Networks* (*PILOT*), a protocol family for reliable multicasting and data sharing [LEH04]. We recall the functionality of these systems in Section 2.4.2.

All of the proposals for EID systems mentioned above suffer from at least one of the following three drawbacks. First, all systems with exception of PILOT have been designed for a specific application and do not provide generic mechanisms for data sharing in MANET. Second, none of the EID systems explicitly considers the limited availability of memory resources on mobile devices, since all proposals use a buffer of infinite size for storing data items during the dissemination process. Third, none of the EID systems provides mechanisms to ensure coherence of the disseminated data. We present a general purpose EID system that uses buffers with finite capacity and comprises implicit and explicit mechanisms for providing data coherence in Chapter 3. Furthermore, we present a special-purpose system for the dissemination of frequently changing information and evaluate the data coherence provided by this system in Chapter 4.

Designing EID systems certainly requires methodology and tools for the performance analysis of different design alternatives. Most proposals of EID systems conduct such performance analysis by simulation, e.g., [PS01], [GSXL02], [HLM03]. Other proposals solely use analytical performance models, or extend simulation results by analytical models, e.g., [PS01], [SH03], [LEH04], or [KBTR02] that provides the motivation for [HBR03]. We recall the analytical modeling approaches in Section 2.4.3. Similar to the proposed EID systems, most analytical performance models suffer from three drawbacks. First, all performance models consider the dissemination of a single data item in isolation. Therefore, none considers the impact of the competition for limited resources, e.g., memory capacity in finite buffers that is induced by the concurrent dissemination of multiple data items. Second, all performance models analyze the transient behavior of the EID system for determining the fraction of mobile devices that receive a data item in a finite time horizon. Third, many of the performance models require offline simulations to determine model parameters, e.g., the contact rate between mobile devices. We present a steady-state modeling approach that considers the spread of multiple data items, incorporates finite buffers with LRU replacement, and does not require offline simulations to determine model parameters in Chapter 5.

1.2 Contributions of this Thesis

The contribution of this thesis is three-fold. It can be divided into the fields of epidemic dissemination of index information, epidemic dissemination of presence information, and stochastic modeling of EID systems. This section summarizes the main results in each of the fields.

Epidemic Dissemination of Index Information

As a first contribution of this thesis we introduce *Passive Distributed Indexing* (*PDI*), a general-purpose distributed lookup service for mobile applications. PDI maintains a distributed index based on epidemic dissemination of index entries and supports the resolution of application-specific keys to application-specific values. As building block, PDI stores index entries in form of (key, value) pairs in buffers denoted as *index caches*, which are maintained by each mobile device. Opposed to all previously proposed EID systems, index caches have a finite size and Least Recently Used (LRU) replacement is employed when the capacity of an index cache is exceeded. Using index entries from the index caches of nearby devices, PDI can resolve most queries locally without forwarding messages to devices located outside the transmission range of the inquiring node. To foster information dissemination for devices with limited transmission range, PDI employs a message relaying mechanism denoted as *selective forwarding*. Selective forwarding deletes duplicate results from response messages before relaying them to achieve high bandwidth efficiency. Beyond all previous system proposals, PDI introduces two novel consistency mechanisms for keeping index caches coherent: (1) Inconsistencies in index caches due to intermittent connectivity or node failures are handled by configurable *value timeouts*, implementing implicit invalidation. (2) *Lazy invalidation caches* reduce the fraction of stale index entries due to data modifications, implementing explicit invalidation. Similar to the epidemic distribution of index entries, timeout values are propagated and invalidation caches are filled by epidemic dissemination of control information. In a comprehensive simulation study based on the network simulator ns-2 [Fe05] and using a workload inspired by a mobile P2P file sharing application, we illustrate the performance of PDI. The results show that configuring PDI for 2-hop forwarding increases hit rate by almost 40% in systems with low device density or limited transmission range. Furthermore, increasing the maximum device velocity from pedestrian speed to vehicular speed in an urban environment does not increase the hit rate, but reduces the message volume by up to 26% due to the impact of selective forwarding. Configuring index cache sizes and selective forwarding appropriate can increase the hit rate by 30%

and more for applications with a high amount of data or low locality in the request stream. Putting it all together, the suitable configuration of index cache size and selective forwarding PDI achieves a hit rate of more than 90%. Furthermore, with the combination of both value timeouts and lazy invalidation caches, more than 95% of the results delivered from the index caches are up-to-date. Performance experiments for other mobile applications confirm the results obtained for P2P file sharing and provide further key insight into the behavior of PDI. To complete the presentation of PDI, implementation and security issues are discussed.

Epidemic Dissemination of Presence Information

Although performance results show that the PDI system can be well applied to disseminate presence information in mobile instant messaging (IM) systems, the discussion indicates that neither the mechanisms provided by PDI nor the performance measures used for their evaluation are tailored to such application. Thus, as a second contribution of this thesis, we analyze various approaches for the proactive dissemination of presence information in highly partitioned MANET. Since presence information are time-critical and subject to frequent changes, we introduce *sustained consistency* as a novel performance measure that quantifies the fraction of time in which the IM system displays a coherent view of the current presence states. As further performance measure of interest, we evaluate the control traffic generated by the dissemination of the presence states. To compare different design alternatives for presence information dissemination, we conduct discrete-event simulation based on a high-level stochastic model of the IM system and derive upper bounds for sustained consistency as well as lower bounds for the control traffic. These bounds show that an approach that implements epidemic dissemination of presence information by using both (1) one-time pushing of state information to all reachable devices and (2) successive periodical polling for up-to-date state information constitutes the method of choice. Building upon these results, we propose *SPEED*, the *System for Presence information Exchange by Epidemic Dissemination* in MANET. SPEED constitutes a hybrid approach combining a push and a pull model and incorporating controlled flooding, caching, and epidemic dissemination of presence information. To optimize performance, SPEED estimates the current node density in the system by counting the direct neighbors of each node and adaptively adjusting system parameters according to these estimates. To illustrate the effectiveness of SPEED, we conduct a detailed simulation study using ns-2 and compare the performance of SPEED to the performance bounds derived earlier and to an optimized approach based on periodical flooding of presence information. The simulation study reveals that for low node

density SPEED achieves up to 20% higher sustained consistency than the flooding-based approach due to epidemic information dissemination. For higher node density, epidemic information dissemination saves up to 48% control traffic compared to the flooding-based approach while providing equal sustained consistency. Compared to the theoretical bounds the presented configuration of SPEED achieves high traffic efficiency at the expense of reduced sustained consistency. However, sensitivity studies show that SPEED can be easily configured to achieve higher sustained consistency at the expense of increased control traffic.

Stochastic Modeling of Epidemic Information Dissemination

As a third contribution of this thesis we present a stochastic modeling approach for EID systems in MANET. Beyond previous work, the introduced approach explicitly considers the concurrent dissemination of multiple data items, finite buffer capacity at mobile devices and an LRU buffer replacement scheme. Furthermore, the model can be employed for steady-state analysis rather than transient analysis and does not require off-line simulation for determining model parameters. We show how EID systems with finite buffers can be modeled by a discrete-time Markov chain (DTMC) and derive customized DTMC models for the well-known EID systems 7DS and PDI. Unfortunately, the state space of the DTMC models grows exponentially in the number of nodes and the number of data items in the system. Thus, for relevant system sizes a numerical computation of steady-state performance measures based on the DTMC models is computationally infeasible. However, extending the approach for modeling LRU buffer management proposed by Dan and Towsley [DT90], [DDY90] we introduce an approximate modeling approach for the numerical computation of steady-state performance measures. As major difference to [DT90], [DDY90], in an EID system updates of the LRU stack may result from hits in both the local buffer and the buffers of remote devices. Using this approximate modeling approach, we analyze the performance of four variants of 7DS. In particular, we perform a comparative evaluation of systems comprising mobile devices running 7DS with and without power conservation as well as with and without support of fixed infostations. A validation of the results derived from the approximate modeling approach against the results of a detailed simulation study shows an excellent agreement. In fact, the approximate results almost always lie in the 99% confidence interval of the corresponding simulation results. The performance study of 7DS shows that neither the transmission range nor the selected variant of 7DS has a significant impact on the fraction of dataholders in the long run. However, for high transmission ranges, the selected variant of 7DS has a significant impact on the hit rate. Depending on the

7DS variant and the buffer size, hit rates between 0.48 and 0.92 can be achieved. For low transmission ranges, the hit rate lies between 0.37 and 0.77 regardless of the 7DS variant. Enabling aggressive power conservation in scenarios with high transmission ranges significantly degrades the hit rate. Furthermore, performance results indicate that a reduced transmission range of $115m$ yields higher hit rates than using power conservation at $230m$ transmission range and that query repetition is most effective for systems with small buffers and small transmission ranges.

1.3 Publications Making up this Thesis

This thesis is based on the following publications that are listed in the order of their appearance. All publications were co-authored with Christoph Lindemann, who contributed to their completion by visions, guidance, and advice. Furthermore, some of the publications are based on joint work with other Ph.D. students. In the later cases, the individual contributions of the author of this thesis are outlined below.

- The performance study of Gnutella in MANET presented in Chapter 2 has been published in [KLW03] and appeared in a modified version in [KLW04a]. Both papers have been co-authored with Alexander Klemm. In [KLW03], the author identified the shortcomings of off-the-shelf P2P systems and designed the search algorithm of the ORION system while Alexander Klemm developed the transport protocol of the ORION system. In [KLW04a], again, the author discussed the shortcomings of off-the-shelf P2P systems and outlined PDI as an example for a P2P system based on epidemic information dissemination, while Alexander Klemm outlined ORION as an example for a P2P system based on cross-layer information transfer.

- The basic concept of the PDI protocol presented in Chapter 3 has been proposed in [LW02a]. [LW03] introduces the consistency mechanisms for PDI and has received the *best paper award* at *ACM MobiDE 2003*. [LW04a] is an extended version of [LW02a] and [LW03] that presents a comprehensive overview over the PDI system. Some aspects of employing PDI for disseminating presence information in a mobile instant messaging system have been discussed in [LW05a].

- The discussion of mechanisms for disseminating presence information in MANET and the proposal of the SPEED system presented in Chapter 4 have been published as a technical report [LLW05]. The report has been co-authored with

Christian Lambert. In [LLW05], the author derived the performance bounds of mechanisms for disseminating presence information in MANET. Furthermore, he identified the major design principles of the SPEED system. Christian Lambert conducted the performance optimization of both the SPEED system and the flooding-based approach used for comparison.

- The approximate steady-state modeling approach for EID systems with finite buffers presented in Chapter 5 has been published in [LW05b].

The author also co-authored several publications which are not part of this thesis. In [LW01], the author presented a comprehensive performance study of different protocols for cooperative web caching in the backbone of the German gigabit research network G-WIN. The performance of Web-cache replacement schemes under current and future workloads has been analyzed by the author in [LW02b]. Both [LW01] and [LW02b] are related to Web caching and therefore beyond the scope of this thesis. However, some ideas for buffer management were inspired by the author's experiences with caching algorithms. In two papers co-authored with Alexander Klemm the author characterized the workload of P2P file sharing systems based on measurements in the Gnutella P2P network [KLW04b], [KLVW04]. In these publications, the author supported the development of an analytical framework to characterize user behavior and conducted experiments to confirm the representativeness of the measured data. Both publications [KLW04b], [KLVW04] are part of the Ph.D. thesis of Alexander Klemm [Kle05]. The publications [LTK+00] and [LTK+02] focus on the performance analysis of time-enhanced UML diagrams. In these publications, the author supported the development of a novel method for analyzing a special class of general semi Markov processes and derived some of the performance curves. Both publications [LTK+00], [LTK+02] are part of the Ph.D. thesis of Axel Thümmler [Thu03].

1.4 Thesis Outline

This thesis is organized as follows. Chapter 2 discusses the deployment of P2P systems in MANET. It recalls the architecture of state-of-the-art P2P systems for the Internet in Section 2.1 and identifies shortcomings for such systems in MANET in Section 2.2. As approaches to cope with these shortcomings it introduces P2P systems based on cross-layer information transfer and P2P systems based on epidemic information dissemination in Sections 2.3 and 2.4, respectively.

Chapter 3 proposes the PDI system for the epidemic dissemination of index information to implement a distributed lookup service in MANET. After discussing the requirements of lookup services in Section 3.1, it introduces PDI and presents mechanism for message forwarding and consistency maintenance in Section 3.2. A generic workload model for characterizing mobile applications is introduced in Section 3.3. The workload model is employed in a comprehensive simulation study that analyzes the performance of PDI in Section 3.4. Completing the design of PDI, Section 3.5 discusses implementation and security issues.

The dissemination of frequently changing presence information in MANET is subject of Chapter 4. This chapter presents sustained consistency as a novel performance measure for quantifying the coherence of the presence information displayed by a mobile instant messaging system in Section 4.1. Section 4.2 uses both sustained consistency and control traffic to derive performance bounds for different approaches for presence information dissemination. Based on the performance results, SPEED is presented in Section 4.3. Section 4.4 compares the performance of SPEED to the performance of an optimized flooding approach for presence information dissemination.

An approximate modeling approach for EID systems with finite buffers is presented in Chapter 5. The chapter derives an approach for modeling EID systems as DTMC, shows how to customize this approach for the 7DS and PDI systems, and validates the DTMC models by discrete-event simulation in Section 5.1. An approximate analysis approach for deriving steady state performance measures is presented and applied to the 7DS System in Section 5.2. The approach is validated and employed for a comprehensive performance study of 7DS in 5.3.

Finally, Chapter 6 summarizes the results of this thesis and gives some concluding remarks.

Chapter 2

Peer-to-Peer Systems in Mobile Ad Hoc Networks

MANET and P2P systems are both infrastructure-less and self-organizing. Due to this common properties, P2P systems seem to be naturally attractive for the deployment for MANET. However, although similar solutions can be applied to enable self-organization and communication in both P2P systems and MANET, straightforward combination of both approaches multiplies the performance problems of each individual approach. This chapter recalls the technology of P2P systems for the Internet and identifies shortcomings that hinder the straight-forward deployment of such systems in MANET. Based on the observations, it identifies two different solutions for the deployment of P2P systems in MANET: cross-layer information transfer and epidemic information dissemination. While cross-layer information transfer is beyond the scope of this thesis and is therefor only briefly reviewed, the main focus of this thesis is on epidemic information dissemination in MANET. Hence, this chapter discusses both system proposals based on epidemic information dissemination and analytical modeling approaches for such systems. Some results on the shortcomings of P2P systems have been discovered in a performance study that has been published in the proceedings of the *IEEE Semiannual Vehicular Technology Conference* (*VTC-Fall 2003*) [KLW03]. A survey on solution approaches for these shortcomings has been published in *Lecture Nodes in Computer Science 2965* [KLW04a].

2.1 Peer-to-Peer Systems for the Internet

The operation of P2P systems for the Internet is typically based on the construction of a logical structure on top of the physical structure of the underlying network [LCP+05]. This logical structure, which is denoted as *overlay network* or simply *overlay*, connects peers using direct application-layer connections. The most common functionality provided by an overlay is key-based searching for data items. Overlay-based P2P systems can be subdivided into unstructured and structured systems, based on the way data is distributed among the peers and searching is performed. In unstructured systems, data items are not placed at particular peers, but each data item is stored by the peer that contributes this item to the system. Subsequently, in the worst case all peers must be contacted to locate a particular data item. Most structured overlay systems implement *distributed hash tables* (*DHTs*). That is, each data item is placed at a peer that is determined by applying a hash function on the data item. To search for a data item by a key, the key is also mapped onto a pseudo-unique identifier using a hash function. The pseudo-unique identifier is used to route a query message across the overlay to the peer that stores the data item matching the key. Note that DHT systems require a one-to-one matching between keys and data items.

Examples for unstructured P2P systems constitute Gnutella [Cli01], [For] and KaZaA [KaZ]. Since we use Gnutella to illustrate the shortcomings of P2P systems designed for the Internet in a mobile environment in the next section, we recall its basic operation. Gnutella constitutes a distributed platform for file sharing. In contrast to previous file sharing systems such as Napster [Nap] that employ a centralized server for searching and use P2P communication only for file transfer, Gnutella implements an entirely distributed search mechanism. Gnutella clients construct an overlay that is used to route messages for searching among the peers. To establish overlay connections to other peers, a peer sends a connection request using the Hypertext Transfer Protocol HTTP. For initial connections, a list of initial peers is retrieved from a well-known server. Furthermore, the Gnutella protocol specifies four other types of messages. Messages of types Ping and Pong are used to maintain overlay connectivity and obtain information about other peers. Messages of type Query contain a set of keywords to specify the files a user is searching for. Searching is performed by controlled flooding of the overlay network. That is, a Query message is sent to the neighboring peers in the overlay network. These peers forward the message to their neighbors and so on. Since the Gnutella overlay network may have arbitrary loops, a flooded Query message may reach the same peer multiple

times. Forwarding a QUERY message more than once is prevented by storing the global unique identifier (GUID) of the Query message in a routing table, along with the identity of the directly connected peer that the query is initially received from. The maximum number of overlay hops that a Query message may travel is specified by a time-to-live (TTL) field, which is set to 7 when the message is generated. The field is decremented each time the message is forwarded, and the message is not forwarded if TTL is equal to zero. Each peer that stores a file matching the keyword in a Query message responds with a message of type QueryHit. The QueryHit message is transferred to the inquiring peer on the reverse overlay path on which the Query message was routed to the responding peer, using the GUID-based routing table.

Note that searching the network by flooding generates a high amount of search messages, leading to poor performance and limited scalability of Gnutella. Several recent approaches try to improve the scalability of the search process by reducing the number of messages generated to resolve a user query. For example, the current Gnutella protocol specification includes an option to form an overlay of *ultra peers* that are powerful nodes with high-bandwidth network connections. Ultra peers are used to route Query messages on behalf of weaker peers denoted as *leafs*. However, searching is performed by flooding the overlay maintained by the ultra peers. Recent academical approaches try to improve the scalability and searching performance of unstructured overlays by using more sophisticated search techniques than plain flooding. For example, [LCC+02] proposes to traverse the overlay by a random walk instead of flooding it. Furthermore, it explores how proactive file replication improves the performance of such searching approaches.

Structured overlays constitute another approach for improving the searching performance. As an example for structured approaches, Aberer, Punceva, Hauswirth, and Schmidt propose P-Grid, a virtual binary search tree that is used to route query messages to a number of nodes which are responsible to answer these queries [APHS02]. Each peer in a P-Grid has to maintain application-layer connections to two other peers. In DHT systems such as CAN [RFH+01] and Chord [SMK+01], a query can be resolved involving $O(log\, n)$ (or $O(n^c)$ for $c < 1$) overlay hops for systems with n peers. Each node in CAN has to maintain connections to $2d$ neighbors, where d is a configuration parameter. Chord maintains connections to $O(log\, n)$ neighbors.

We conclude from this overview over P2P systems for the Internet that all state-of-the-art P2P systems are based on overlay networks with static connections between participating peers. Furthermore, several proposals try to improve the search performance of state-of-the-art P2P systems. In the next section, we show that not

the search algorithm but rather the algorithm for constructing and maintaining the overlay network constitutes the major shortcomings for the deployment of P2P systems in MANET.

2.2 Deploying Peer-to-Peer Systems in Mobile Ad Hoc Networks

This thesis focuses on networks of mobile devices such as laptop computers, PDAs or mobile phones. Such devices are typically equipped with interfaces that enable short- to medium range wireless communication, e.g., IEEE 802.11 wireless LAN adapters [IEE97] or Bluetooth adapters [Bis01]. We assume that several devices use a MANET to cooperate in a distributed application. We denote the entity of the device hardware, the application software, and the data stored by the application as *node* in the remainder of this thesis.

Since both MANET and P2P systems are infrastructure-less and self-organizing, it seems to be attractive to use P2P technology as a building block for MANET applications. Deploying an off-the-shelf P2P system on top of a MANET routing protocol seems to be a straightforward approach. However, though attractive because of its off-the-shelf nature, this approach reveals several shortcomings due to limited bandwidth availability and node mobility. Recall that each peer participating in an overlay-based P2P system has to maintain overlay connections to other peers. The maintenance of an overlay connection between two arbitrary peers in a MANET imposes several problems, as illustrated in Figure 2.1. In this figure, mobile nodes are drawn as circles named by capital letters A to G. If two nodes are located in each others transmission range, a thin line connects the corresponding circles. E.g., direct communication between nodes D and G is possible, while it is not possible between nodes A and G. We refer to a direct wireless connection as *physical connection*, while a sequence of physical connections as provided by a routing protocol is denoted as *network-layer connection* or *network route*. The sum of all physical connections constitutes the *physical topology* of the MANET. Note that the physical topology may frequently change due to node mobility.

Overlay connections are shown as thick lines in Figure 2.1. Note that an overlay connection can be established across several hops, i.e., spanning several physical connections. For example, Nodes A and G are connected by an overlay connection which uses the route spanning the physical connections A–D–G.

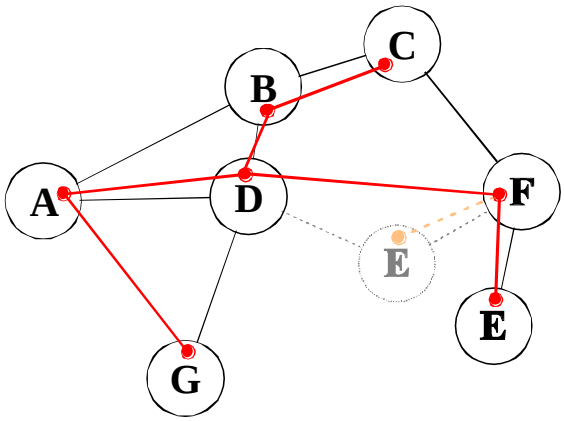

Figure 2.1. Link-layer versus application-layer connections

By analyzing scenarios as shown in Figure 2.1 we identify the following shortcomings of static overlay systems for MANET:

Inefficient usage of physical connections: Most overlay construction mechanisms are unaware of the physical topology. Thus, multiple overlay connections may use the same physical connection and a message sent from one node to another in the overlay may traverse the same physical connection several times. For example, consider a message sent from Node C to Node F in Figure 2.1. The message is forwarded using the overlay connections C–B, B–D and D–F. Considering the routes defined by the physical topology, it is easy to see that the message traverses the physical connections D–B and B–C twice. This phenomenon that can also be observed for connections between fixed hosts in the Internet [RIF02], has a dramatic impact in wireless environments due to scarce radio bandwidth and energy resources. Later in this section we show that inefficient usage of link-layer connections result in significantly increasing transmitted message volume.

High Routing Overhead: Maintaining an overlay connection between two nodes in a MANET requires some efforts on the network-layer. Node movement forces the routing protocol to permanently adapt routes to the physical topology, generating a high amount of control messages. For example, consider the overlay connection between Nodes D and F in Figure 2.1, which has been established over the route D–E–F when Node E was located at the position indicated in gray. After Node E has moved to the final position, a route discovery must be initiated to maintain a network route for the application-layer connection between Nodes D and F. Note that network routes may also change in the Internet. However, this event is much more frequently in MANET due to node mobility. We show that routing overhead is a serious problem for overlay-based P2P systems later in this section.

Frequent loss of application-layer connections: In general, the ad hoc routing protocol may fail to maintain a route between two nodes due to node mobility and intermittent connectivity. In the worst case, this can lead to a partitioning of the overlay network or to the disconnection of nodes. For example, consider that the link-layer connections A–B and A–D in Figure 2.1 fail. In this case, G loses the overlay connection to A, and is disconnected from the overlay, although a physical connection to D still exists. Note that if a peer loses the connection to the overlay, it has to employ a bootstrap mechanism for establishing new connections. Peers may well lose connection to the application-layer network in P2P systems for the Internet; however, this scenario is much more frequent in MANET due to node mobility. Later in this section we show that traffic generated by initial peer discovery results in significant control overhead.

To illustrate the impact of each of the problems described above, we analyze the performance of an off-the-shelf approach using a P2P system from the Internet which utilizes a MANET routing protocol in a simulation study. Recall that most P2P systems are based on an overlay network. Thus, we consider Gnutella as an example for such P2P systems. To complete the off-the-shelf approach, we employ TCP-SACK as transport-layer protocol on top of the MANET routing protocol DSR. Performance results are obtained using the Network Simulator ns-2 [Fe05]. We use an IEEE 802.11 MAC and physical layer and the two-ray ground propagation as radio propagation model implemented in ns-2. The simulated mobile nodes have a transmission range of approximately $125m$.

We simulate $N_{Nodes} = 40$ mobile nodes moving in an area of $1000m \times 1000m$ according to the random waypoint mobility model [BMJ+98], which is commonly used to mimic the movement of pedestrians. According to this mobility model, each node starts at a location randomly chosen from the simulation area according to a uniform distribution. It moves to another randomly chosen location with a speed chosen uniformly from $[0, v_{max}]$. When a node reaches its destination, it rests for a period T_{hold}, before it continues its movement to the next randomly chosen destination with a randomly chosen speed. For this experiment, we use $v_{max} = 2m/s$ and $T_{hold} = 50s$.

In the simulations, we focus on the Gnutella search algorithm and do not consider file transfers, since file transfers are performed outside the overlay using direct TCP connections. To generate a workload for the search algorithm, we assume that each node shares N_{Docs} unique documents. We identify the documents by unique numbers d, $1 \leq d \leq N_{Nodes} \cdot N_{Docs}$. Each document d matches a randomly chosen set $\mathcal{K}_d \subset \mathbb{N}_0$ of keys. To determine $\mathcal{K}_d$ for each document d, we adopt the principle of *selection*

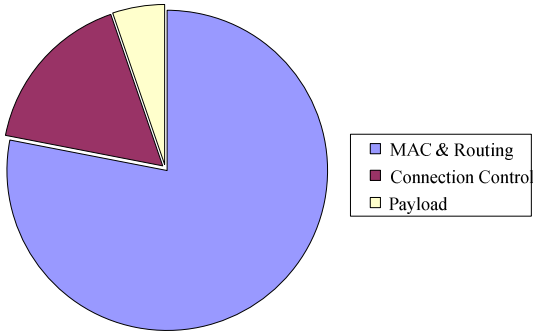

Figure 2.2. Breakdown of message volume for the Gnutella P2P system in a MANET

power [YGM01], which assumes that the probability that key k matches a document d is $f(k)$. Experiments have shown that $f(k) = (1/\lambda)e^{-(k/\lambda)}$ for $\lambda \approx 100$ provides a sufficient model for selection power. Note that $f(k)$ is independent of the document d, i.e., selection power depends only on the key k. Following [Sri01], [KLVW04], we assume a Zipf-like distribution [BCF+99] with $\alpha = 0.9$ for the probability $g(k)$ that a given query is for key k (i.e., $g(k) \cong k^{-0.9}$). Furthermore, we assume a correlation between query popularity and query selection power, i.e., the most popular key 1 with popularity $g(1)$ has the highest selection power $f(1)$ [YGM01]. Each node sends queries in intervals determined by a Poisson process with a mean interarrival time of $120s$. For simplicity, we use only a single key in each query, which is chosen randomly according to the popularity distribution $f(k)$.

During the simulation, we record the volume of all messages transmitted by the mobile nodes. We brake down the total message volume into three categories depending on the message types. Gnutella Query and QueryHit messages are classified as *payload*, since they constitute the only message types that are used for searching. Gnutella ConnectionRequest, Ping and Pong messages are used to establish and maintain overlay connections and are therefore classified as *connection control*. All messages used by DSR for establishment and maintenance of network routes as well as the request-to-send (RTS), clear-to-send (CTS) and acknowledgment (ACK) packets generated by the IEEE 802.11 medium access control mechanisms are classified as *MAC & routing*.

The breakdown of message volume is shown in Figure 2.2. Confirming the claims made earlier in this section, the figure shows that the payload constitutes only 5.2% of the transmitted message volume. Connection control traffic for maintenance of overlay connectivity constitutes 16.6% of the volume. The major share of 78.2% is consumed by MAC & routing traffic. Note that this experiment indicates that lim-

ited scalability due to the flooding-based search mechanism is not the key factor for limiting the applicability of Gnutella in MANET. In fact, applicability is significantly limited by the mismatch between the overlay and the physical topology as well as by node mobility. These results have been confirmed by a recent performance study of Gnutella in MANET [TCG05]. We conclude from Figure 2.2 that for the successful deployment of P2P systems in MANET more sophisticated approaches than the employment of off-the-shelf components are required. Examples for such approaches constitute P2P systems using *cross-layer information transfer* and P2P systems using *epidemic information dissemination*, which are discussed in the following sections.

2.3 Peer-to-Peer Systems using Cross-Layer Information Transfer

As discussed in Section 2.2, unawareness of the overlay construction algorithm for the physical topology is a major source of overhead when employing an off-the-shelf P2P system in MANET. As a consequence, efficient deployment of P2P systems in MANET requires close cooperation between protocols and applications running on different layers of the TCP/IP protocol stack. For example, it is easy to see that the application-layer can benefit from knowledge of network-layer routes by adjusting overlay routes accordingly.

Several recent system proposals try to exploit cross-layer information transfer for building P2P overlays in MANET. Conti, Giordano, Maselli, and Turi proposed a reference-architecture for cross-layer design of a MANET protocol stack [CGMT03], [CMTG04]. The architecture uses a shared data structure called *network status*, which serves as a repository of network information collected on all layers off the TCP/IP stack. Besides sharing information among network layers, the network status can be used by a protocol running on one layer to trigger an event on another layer. For example, the application can be notified by the link layer if the loss of a physical connection is detected. In recent work, Turi, Conti, and Gregori apply the network status to optimize the Gnutella overlay network for MANET [TCG05]. Their approach is build on an enhanced version of the OLSR routing protocol that passes the address and hop distance to other Gnutella peers to the application layer using the network status. For the initial establishment of overlay connections, peers are selected in ascending order of the hop distances. Furthermore, the routing protocol notifies the application layer if a new Gnutella peer is detected. If the new peer

is closer than any peer with an already established connection, the connection is terminated and a connection to the new peer is established.

In the context of P2P communication in MANET, Chen, Shah and Nahrstedt proposed a system for implementing data advertising, lookup and replication based on cross-layer information transfer [CSN02]. In their approach, the network-layer provides a *node profile* that stores information about a nodes position, velocity, and energy resources to a middleware-layer. Similarly, the middleware-layer provides a *data profile* that contains information on the quality-of-service requirements of each data flow to the network-layer. Node and data profiles are communicated via shared memory, requiring the employment of a proactive link-state routing protocol that is capable to communicate with the middleware-layer.

Schollmeier, Gruber, and Niethammer presented the Mobile Peer-to-Peer Protocol (MPP) suite that spans from the network to the application-layer [SGN03]. The suite tries to reuse existing protocols as far as possible. It uses an enhanced version of DSR called *EDSR*. P2P applications can register with EDSR and communicate information using a message passing mechanism called *Mobile Peer Control Protocol* (*MPCP*). Information passed via MPCP are used to establish application-aware routes that can be employed for data transfers using standard mechanisms, e.g., the hypertext transfer protocol HTTP.

In a paper co-authored with Alexander Klemm, the author introduced the *Optimized Routing Independent Overlay Network* (*ORION*, [KLW03]). ORION comprises an algorithm for the construction and maintenance of an application-layer overlay network that enables routing of all types of messages required to operate a P2P file sharing system, i.e., queries, responses, and file transmissions. Overlay connections are set up on demand and maintained only as long as they are used, closely matching the current physical topology. ORION combines application-layer query processing with the network-layer process of route discovery, substantially reducing control overhead and increasing search accuracy compared to the off-the-shelf approach analyzed in 2.2. Additionally, the ORION overlay network enables low overhead file transfer, as well as an increasing probability for successful file transfers compared to the off-the-shelf approach.

Note that all approaches for building P2P overlays by cross-layer information transfer require the existence of routes between all nodes that participate in the P2P system. Thus, such approaches require sufficient node density to provide a high connectivity of the physical topology. In the remainder of this thesis we focus on scenarios with low node density, high node mobility, and frequently occurring network partitions.

2.4 Peer-to-Peer Systems using Epidemic Information Dissemination

2.4.1 An Introduction to Epidemic Information Dissemination

As discussed in Section 2.2, node mobility causes route changes, intermittent connectivity or even disconnected operation, and thus, hampers the deployment of overlay-based P2P systems in MANET. However, as shown by Grossglauser and Tse, node mobility does not only hinder communication in MANET, but also supports cost-effective information exchange [GT02]. They demonstrated a capacity/mobility trade-off by showing that the average throughput per source-destination pair of nodes can be kept constant for increasing node density in a multihop wireless network with mobile nodes. Moreover, they presented a formal proof and simulation results showing that mobility increases the per-session throughput in a MANET for delay-insensitive applications. As major concept, [GT02] does not route messages directly from the source node to the destination node, but hands the message over from the source node to an intermediate node. With the assumption that all nodes are mobile, the intermediate node will encounter the destination node in the long run. Subsequently, the message can be transfered from the intermediate node to the destination node.

Direct applications of the concept of using intermediate nodes for message transmission include efficient route discovery by exploiting encounter ages [DFGV03] and approximating node positions for position-based routing [GV03]. However, this concept can be straightforwardly extended to an effective way for disseminating information from peer to peer in a MANET. This extension is denoted as *epidemic algorithm* for information dissemination or simply *epidemic information dissemination* (*EID*). With epidemic information dissemination, information is transmitted on node encounters, similar to the transmission of an infectious disease between individuals. In the Internet, epidemic algorithms are easy to deploy, robust, and highly resilient to failures [EGKM04]. Furthermore, mathematical models for the spread of infectious diseases have been widely studied, see, e.g., [Bai75], [BCC01], so that performance and reliability can be easily analyzed for a wide class of epidemic algorithms. There exist applications of epidemic algorithms in various fields of computer science, e.g., for the maintenance of replicated databases [DGH$^+$87].

For recalling the functionality of epidemic algorithms for the Internet [EGKM04], assume that N nodes participate in the dissemination process. For ease of exposition,

we denote the nodes by natural numbers $n \in 1, 2, \ldots, N$ and the set of all nodes as $\mathcal{N} = \{1, 2, \ldots, N\}$. For implementing an epidemic algorithm, every node $n \in \mathcal{N}$ buffers messages from other nodes in a finite buffer of size B. Each message is forwarded a limited number of times T. When forwarding the message for time t, $1 \leq t \leq T$, the message is sent to a set $\mathcal{S}_t \subset \mathcal{N}$. The cardinality $F = |\mathcal{S}_t|$ is denoted as *fan out* of the epidemics. The nodes $m \in \mathcal{S}_t$ are selected from $\mathcal{N}$ according to a uniform distribution. The epidemics stops when each node that has received a message has forwarded it T times. In the optimal case, each node $n \in \mathcal{N}$ should have received a message when the epidemics stops. The probability that all nodes are reached by the message is denoted as the *reliability* of the epidemic algorithm. Note that the reliability depends on the choice of the parameters B, T, and F. In particular, higher values for T and F increase the reliability, since forwarding a message many times to a large set of nodes obviously reduces the probability for missing a node. Nevertheless, high values for these parameters also increase the *load* imposed by the epidemic algorithm, i.e, the number of forwarded messages. Thus, designing epidemic algorithms for the Internet requires careful choices of the parameter depending on the application.

In general, it is not possible to transfer an epidemic algorithm straightforwardly from the Internet to MANET for two reasons. First, the selection of target nodes during each forwarding route requires that a node $n \in \mathcal{N}$ knows all nodes $m \in \mathcal{N}$, $m \neq n$. Second, all nodes $n \in \mathcal{S}_t$ selected as targets in round t, $1 \leq t \leq T$, must be reachable during the round. Obviously, due to node arrivals and departures, node mobility, and intermittent connectivity, these requirements cannot be easily fulfilled. That is, the fan F of the epidemic algorithm does heavily depend on the current physical topology of the MANET. Thus, more sophisticated approaches for epidemic algorithms must be used to implement information dissemination in a MANET.

In the remainder of this section, we discuss proposals for EID systems in MANET. Furthermore, we recall approaches for the performance and reliability analysis of these systems.

2.4.2 System Proposals

Seven Degrees of Separation

As first proposal of a system that uses 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 nodes [PS01]. An application scenario for 7DS constitute people with

notebooks and IEEE 802.11 WLAN interfaces that access the World Wide Web from a crowded area such as a subway platform. Typically, direct access to the Internet is not possible in such environment and Web pages must be exchanged from node to node.

Each mobile node running 7DS maintains a local buffer, i.e., a directory on the local disk, for storing Web pages. To search for Web pages in the buffer of remote nodes, 7DS defines two message types, i.e., Query messages and Report messages. A node that searches for a Web page sends a Query message. The message contains the URL of the Web page and the address of the inquiring node. A Query message is sent to all nodes in the transmission range using single hop multicast in the IEEE 802.11 ad hoc mode. When a node receives a Query message, it searches its buffer for a Web page with the URL specified in the message. If the Web page is found, the node returns a Report message to the inquiring node using a unicast transmission. The inquiring node selects the most appropriate download source from all report messages and transfers the Web page using a direct HTTP connection. Subsequently, this Web page is available to other mobile nodes implementing epidemic information dissemination.

Besides this P2P data sharing among mobile devices, 7DS supports *server-client* (*S-C*) data delivery using fixed or mobile infostations as well as hybrid forms comprising of P2P data sharing and additional S-C data delivery. As additional feature the 7DS system allows mobile nodes to switch the wireless interface on and off for power conservation. With power conservation, the mobile nodes divide the operation cycle in ON periods of duration P_{ON} and OFF periods of duration P_{OFF}. During an OFF period, the mobile node switches off the wireless communication interface and does not respond to queries. Due to the fact that mobile devices hardly have synchronized clocks, the ON and OFF periods are not synchronized among the nodes. Thus, the length P_{ON} has to be sufficient large compared to P_{OFF} to enable successful message exchange between two nodes. [PS01] proposes several design variants for 7DS that combine the features mentioned above. In particular, 7DS systems in P2P mode with and without power conservation (denoted as P and NP, respectively) as well as systems using fixed infostations with and without power conservation (denoted as FIS-P and FIS-NP, respectively) have been introduced.

In contrast to the approach taken by epidemic algorithms for the Internet 7DS nodes do not transmit data items actively but only when it is requested by another node. Thus, the epidemic dissemination is driven by a pull approach rather than by a push approach, and the fan out of the epidemic is $F = 1$. Nodes that can provide a data item are identified by Query and Report message when a transmission is

required, eliminating the requirement that a single node must know all other nodes that participate in the system. As further contrast to epidemic algorithms for the Internet, a data item is sent by a node on each request by remote nodes rather than only for a limited number of times. Note that the dissemination of a data item with 7DS does only stop when all nodes have received the a item, i.e, the reliability of the approach is 1. Approaches for the analysis of the speed of the epidemic dissemination are recalled in Section 2.4.3.

7DS constitutes a special-purpose system for sharing Web documents. The system does not provide mechanisms for resolving consistency issues, e.g., between different revisions of the same web document. Furthermore, 7DS assumes that a buffer of infinite size is available for storing web documents on the mobile nodes and does not consider the impact of a limited buffer on the system performance.

The Shared Wireless Infostation Model

Small and Haas proposed an epidemic algorithm for collecting information in a hybrid network consisting of mobile nodes (i.e., whales) and fixed infostations (i.e., buoys) [SH03]. Their architecture, denoted as *Shared Wireless Infostation Model* (*SWIM*), is a marriage of the infostation concept [GBMY97] with the ad hoc networking model. SWIM assumes that whales are implanted with radio frequency (RF) devices that collect biological data, e.g., heart rate, body temperature, speed, and diving depth of the whale. Such data is divided in packets that store data for a particular period, e.g., one packet a day. Each packet has a unique identifier given by a unique identifier for the whale and a sequence number. With SWIM, data packets are unloaded from the RF devices when whales surface near SWIM infostations mounted on buoys. Data can be unloaded to the infostations using high bandwidth wireless connections in contrast to low bandwidth satellite connections used in conventional systems for collecting biological information. This technology guarantees that all information can be unloaded while a whale surfaces.

To maximize the probability that a data is unloaded at an infostation in a reasonable time, the RF device of a whale also stores information from other whales. Subsequently, all data packets stored by the RF devices of two whales is exchanged when the wales are located in each others transmission range, implementing epidemic information dissemination. To limit the amount of memory consumed by the dissemination process, an RF device will never accept a packet again after it has unloaded this packet to an infostation. Furthermore, each packet is assigned with a time-to-live (TTL) value. After the expiration of the TTL the packet is removed from the memory of all RF devices. The TTL must be carefully chosen to maximize

the probability that the packet is unloaded to an infostation before expiration as shown in Section 2.4.3.

With SWIM, information is actively transmitted from one node to other nodes similar to epidemic algorithms for the Internet. However, the fan out F of the epidemics is not fixed but depends on the number of contacts a whale makes over time. Furthermore, the epidemics does not stop after a predefine number of transmissions T but after the TTL has expired for a packet.

SWIM constitutes a special purpose system for collecting biological information. Due to the semantics of the generation of data packets, SWIM does not have to deal with consistency issues. Furthermore, the design of SWIM is aware of the limited memory resources available at the RF devices and uses a TTL to limit memory consumption. However, the impact of a limited data buffer with a fixed size on the performance of SWIM is not considered in [SH03].

The Probabilistic Lightweight Group Communication System

Luo, Eugster, and Hubaux introduced the *Probabilistic Lightweight Group Communication System for Ad Hoc Networks* (*PILOT*, [LEH04]). PILOT consists of a set of protocols for reliable multicasting and data sharing in MANET. Application examples for PILOT include tracking of node positions for mobility management and storing certificates and cryptographic keys for securing MANET applications. Opposed to 7DS and SWIM, PILOT requires a reactive MANET routing protocol, e.g., DSR as used in [LEH04].

The design of PILOT is build upon a layered architecture with two protocol layers. As basic protocol on the lower layer PILOT uses *Route Driven Gossip* (*RDG*), an epidemic protocol for probabilistic multicast of single packets. Upon RDG, PILOT builds the *Reliable Route Driven Gossip* (R^2DG) protocol for reliable multicast streaming and the *Probabilistic Quorum System for Ad Hoc Networks* (*PAN*) for implementing a reliable distributed storage system. RDG is the only component of PILOT that implements an epidemic algorithm. R^2DG and PAN constitute applications of RDG that piggyback negative acknowledgments or multicast write updates to a set of storage nodes, respectively. Thus, we recall only the functionality of RDG.

RDG applies an approach that is very similar to the operation of epidemic algorithms in the Internet as recalled in Section 2.4.1. The main difference is that each node maintains only a partial view of the group of nodes that participate in the epidemic dissemination process. RDG defines multiple multicast groups, where epidemic information dissemination is performed independently for each group. A node that wants to join a multicast group broadcasts a Join message. Other group

members that receive the Join message reply with a GroupReply message with probability P_{reply}. The probability P_{reply} is set individually by each node based on an current estimate of the size of the multicast group. The GroupReply message is transmitted to the joining node using the reactive MANET routing protocol. Thus, a route is established between the joining node and the responding node. The joining node adds all responding nodes to a list $AView$ of active group members for which a valid route is known. Due to node mobility and nodes leaving the multicast group, nodes must be removed from $AView$ from time to time. If the size of $AView$ falls below a threshold, a new join process must be started.

Based on the partial view of the multicast group provided by $AView$, PILOT operates as described in Section 2.4.1. That is, a message that is received by a nodes is forwarded in $T = \tau_q$ rounds, where τ_q is denoted as the *quiescence threshold*. In each round F, destination nodes are randomly chosen from $AView$ for a fixed fan out F. An additional *age threshold* τ_a limits the overall number of times a packet is forwarded. Beside this basic forwarding mechanism, RDG supports additional operations, e.g., leaving a multicast group, by piggybacking control information on the disseminated data messages.

PILOT constitutes a general purpose system for group communication in MANET. Unfortunately, it requires a reactive routing protocol and is therefore designed for scenarios with high node density, where network partitions are uncommon. The PAN protocol supports reliable data sharing according to the *shared private* consistency model, which requires that read and write requests on a data item are executed in first-in-first-out (FIFO) order. However, consistency maintenance is a feature of PAN, not of RDG, so that the basic epidemic algorithm does not provide means for consistency maintenance. Similar to epidemic algorithms for the Internet, PILOT uses a buffer at each node for storing data during the epidemic dissemination process. Nevertheless, the impact of a limited size of the buffer is not considered in the system proposal.

Other System Proposals

Besides 7DS, SWIM, and PILOT, some less prominent approaches for epidemic information dissemination have been proposed. Using an approach very similar to 7DS, Goel, Singh, Xu and Li proposed spitting a shared file into segments and broadcasting these segments using a redundant tornado encoding [GSXL02]. Their approach enables nodes to restore a file in case a sufficient number of different segments have been received from one or more sources. Thus, dissemination of large files is performed more efficiently and reliably as with 7DS. In [KBTR02], Khelil, Becker, Tian,

and Rothermel presented and analyzed a simple epidemic information dissemination algorithm inspired by the SPIN-1 protocol [HKB99]. Very similar to SWIM, the algorithm exchanges information at each contact between mobile nodes. Building upon these results, Häner, Becker, and Rothermel present the *Negotiation-based Ad Hoc Data Dissemination Protocol* (*NADD*) [HBR03], which uses active data advertisements to negotiate data transmissions between neighboring nodes. Similar to SWIM, NADD implements a push model for information dissemination. Hanna, Levine, and Mamatha proposed a fault-tolerant distributed *P2P information retrieval* (*P2P-IR*) system for document sharing in MANET [HLM03]. Their approach distributes the index of a new document to a random set of nodes that are neighbors in the physical topology 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. All these proposals of special-purpose systems do not consider coherence of disseminated information and finite buffers at the mobile nodes.

2.4.3 Performance and Reliability Analysis

Analysis of 7DS

[PS01] uses a simulation study based on the network simulator ns-2 to evaluate different design variants by considering the percentage of dataholders and the delay for receiving a data item in a finite time horizon. They represent the popularity of this data item by varying the number of mobile nodes which are querying this item. Beside this simulation study, they presented a simple analytical model based on a diffusion-controlled chemical process [OTB89] for analyzing the transient behavior of the FIS variant of 7DS. This model is basically a trapping process that assumes that particles of type C move in a d-dimensional space according to a diffusion motion. Particle of type S, denoted as *traps*, are statically and uniformly distributed in the space. The particles of type C are absorbed by the traps when they step on them. It has been shown that for the survival probability ϕ_n of particles of type C after time n holds:

$$log(\phi_n) = -\alpha \left(log \left(\frac{1}{1-q} \right) \right)^{2/(d+2)} n^{d/(d+2)} \tag{2.1}$$

Here, α is a constant taken from the diffusion process, and q is the concentration of traps in the space.

Considering the dissemination of a single data item and assuming that a 7DS client node always acquires an item when it moves into the transmission range of an infostation, 7DS clients can be modeled by particles of type C. Fixed 7DS infostations

are modeled by traps. Then, for the constant q holds $q = \pi R^2/A$ for a simulation area of size A and a homogenous transmission range R. The fraction of nodes that have acquired an information after time n is given by $1-\phi_n$. For a constant α taken from literature, [PS01] shows that the analytical results closely matches simulation results for sufficient large transmission ranges R. However, the simulation study uses the diffusion mobility model assumed in [OTB89], whereas other simulation studies in [PS01] use the random waypoint mobility model, which is much more realistic. Note that the modeling approach is designed for a transient analysis and does only consider the dissemination of a single data item. The impact of the dissemination of multiple data items and a finite buffer size on the dissemination process is not considered.

Analysis of SWIM

To determine a TTL that provides a probabilistic guarantee for unloading a data packet a buoy, [SH03] models the dissemination of a single packet as the spread of an infectious disease. Applying the framework that is commonly used in this context [Bai75], [BCC01], a whale is *infected* if it stores the considered data packet in the memory of its RF device. A whale is *susceptible* to the infection if it does not store the item yet, and it is *recovered* if it has successfully unloaded the item at a buoy. The spread of the data item can be modeled by a continuous-time Markov chain (CTMC). The states of the CTMC are given by (S, I, R), where S, I, and R denote the number of susceptible, infected and recovered whales, respectively. At time t we have $S(t) + I(t) + R(t) = N$ for the total number of whales N, and $I(0) = 1$, $S(0) = N-1$, and $R(0) = 0$, since only one whale is infected with the packet at the moment of packet generation. The rate of a state transition $(S, I, R) \rightarrow (S-1, I+1, R)$ is given by βSI for a contact rate β between the whales. The rate of a state transitions $(S, I, R) \rightarrow (S, I-1, R+1)$ is given by γI for a contact rate γ between a whale and a buoy. [SH03] uses off-line simulation to determine the constants β and γ for different mobility models of the whales.

Goal of the analysis is to determine the time T required until the first whale is recovered, i.e., $R(t) = 0$ for $t < T$ and $R(T) = 1$. Note that T is stochastic, i.e., $T = \tau$ holds with a certain probability depending on τ. [SH03] provides an expression for the cumulative distribution function (CDF) $F(T)$. With this function, it is possible to select a TTL τ so that the probability for unloading a data packet at a buoy before expiration is $F(\tau)$.

For deriving an expression for $F(T)$, note that the CTMC defines the following

fist-order differential equations:

$$\frac{dS}{dt} = -\beta SI \tag{2.2}$$

$$\frac{dI}{dt} = \beta SI = \beta(N-I)I = \beta NI - \beta I^2 \tag{2.3}$$

Using the initial condition $I(0) = 1$, these equations can be solved for

$$I(t) = \frac{N}{1 + e^{-\beta Nt}(N-1)}. \tag{2.4}$$

This expression can be used to determine the CDF

$$F(T) = 1 - K\left(\frac{N-1}{e^{\beta NT} + N - 1}\right) \tag{2.5}$$

with a constant K depending on the mobility model and the distribution of buoys.

Similar to the analysis of 7DS, the analysis of SWIM focuses on the transient dissemination of a single data item. The impact of the dissemination of multiple data items and memory limitations in form of a finite buffer size are not considered. Furthermore, off-line simulation is required for determining model parameters.

Analysis of PILOT

[LEH04] presents an analysis of the RDG protocol. Performance measures of interest constitute the fraction of nodes reached by the epidemic dissemination of a message, denoted as the *reliability degree* R_{ds}, and the network load N_l measured by the number of unicast messages generated by the dissemination times the number of hops they are transmitted. The analysis is based on the observation that for the number of nodes S_r that are infected with a given data item after round r and $P(S_r = 0) = 1$ for $r < 0$, the sequence of random vectors $\{\mathbf{S}_r | r \geq 0\}$ with $\mathbf{S}_r = [S_r, S_{r-1}, \ldots, S_{r-\tau_q}]$ forms a discrete-time Markov chain (DTMC). The states of the DTMC are given by T-tuples $\{1, ..., n\}^T$, where n is the size of the multicast group and $T = \tau_q + 1$. Note that $\{S_r | r \geq 0\}$ does not form a DTMC, since the transition probability from S_r to S_{r+1} depends on the number of infectious nodes in the state S_r. Since this number is given by $S_r - S_{r-\tau_q}$, the transition probability is not independent from the history of states, i.e., the Markov property is violated.

The analysis uses the probability p that a node is infected by a particular forwarding message, and $q = 1-p$. The probability p can be determined by the probability to chose a node as target for a message and some observations on the physical topology of the network that is estimated by off-line simulations following [LEH03].

Assume that in round r the number of infected nodes is $i := S_r$, the number of infectious nodes is $k := S_r - S_{r-\tau_q}$, and the number of susceptible nodes is $n-i$. We define a binary random variable X_l for each susceptible member l, $1 \leq l \leq n-i$. With $P(X_l = 0) = q^k$, it is easy to see that $S_{r+1} - S_r = \sum_{l=1}^{n-i} X_l$ follows a binomial distribution. Thus, it holds

$$\begin{aligned} & P(S_{r+1} = j | S_r = i, S_r - S_{r-\tau_q} = k) \\ = & \begin{cases} \binom{n-i}{j-i}(1-q^k)^{j-i} q^{k(n-j)} & j \geq i \\ 0 & j < i \end{cases} \end{aligned} \tag{2.6}$$

Using Equation 2.6, the global balance equation of the DTMC for a state $\mathbf{s}_{r+1} = [j, i_0, i_1, \ldots, i_{\tau_q - 1}]$ and arbitrary states $\mathbf{s}_r = [i_0, i_1, \ldots, i_{\tau_q}]$ with $0 \leq i_{\tau_q} \leq i_{\tau_q - 1}$ and $i = i_0$ can be written as

$$P(\mathbf{S}_{r+1} = \mathbf{s}_{r+1}) = \sum_{i_{\tau_q}=0}^{i_{\tau_q-1}} \binom{n-i}{j-i} \left(1 - q^{i - i_{\tau_q}}\right)^{j-i} q^{(i-i_{\tau_q})(n-j)} P(\mathbf{S}_r = \mathbf{s}_r). \tag{2.7}$$

This equation can be used to derive the marginal distribution of S_r, i.e., a colum vector $\mathbf{v}_r$ with $\mathbf{v}_r(i) = P(S_r = i)$ by

$$v_r(i) = \sum_{i_1=0}^{i} \sum_{i_2=0}^{i_1} \cdots \sum_{i_{\tau_q}=0}^{i_{\tau_q-1}} P(\mathbf{S}_r = \mathbf{s}_r) \tag{2.8}$$

Using the marginal distribution of S_r after the final round $r = \tau_a$, the CDF of the reliability degree R_{ds} can be computed by

$$F_n(x) = \sum_{i=1}^{\lceil nx \rceil} v_{\tau_a}(i). \tag{2.9}$$

The network load N_l can be computed by

$$N_l = E[S_{\tau_a}] \cdot F \cdot \tau_q \cdot E[H], \tag{2.10}$$

where $E[S_{\tau_a}]$ denotes the expected number of infected nodes after round τ_a and $E[H]$ is the expected distance between two nodes in hops, estimated from the characteristics of the physical network [LEH03].

Note that the analysis of RDG, again, considers only the dissemination of a single data item. The impact of the spread of multiple data items an buffers with finite capacity is not considered in the analysis. Furthermore, off-line simulation is required for determining model parameters.

2.5 Summary

In this chapter, we recalled the functionality of P2P systems for the Internet and pointed out that all state-of-the-art systems construct overlay networks that consist of application layer connections. Using the popular Gnutella file sharing system as an example, we illustrated that overlay-based P2P systems exhibit severe shortcomings when deployed in MANET by simply running them on top of a MANET routing protocol. The main reasons for these shortcomings are a mismatch of the overlay topology and the physical topology of the MANET, as well as a high routing overhead to cope with node mobility and intermittent connectivity. Solution concepts for implementing P2P systems in MANET can be subdivided into the domains of P2P systems using cross-layer information transfer and P2P systems using epidemic information dissemination.

While P2P systems based on cross-layer information transfer are beyond the scope of this thesis and, thus, have been only briefly recalled, we discussed the fundamentals of epidemic information dissemination in more detail. Based on a description of the functionality of epidemic algorithms in the Internet, we argued that such algorithms cannot be directly employed in MANET due to imperfect knowledge of all members of the system and intermittent connectivity among these members. These characteristics imply that the fan out of the epidemic algorithm heavily depends on the current physical topology of the MANET.

We discussed proposals for systems that implement epidemic information dissemination in MANET. Throughout the discussion, we made the following observations: First, most systems are designed to support information dissemination only for a specific application. Second, non of these systems explicitly considers the limited availability of memory resources at mobile devices by limiting the size of the buffers used for storing data items during the dissemination process. Third, non of the systems incorporates mechanism to provide coherence of the disseminated information. A discussion of approaches for analyzing the performance and reliability of EID systems furthermore revealed that all approaches perform only transient analysis, do only consider the dissemination of a single data item in isolation, and ignore the impact of finite buffers on the system performance. Furthermore, many approaches require off-line simulation to determine model parameters.

In the remainder of this thesis we (1) propose a general-purpose epidemic information dissemination system that features mechanism for maintaining the consistency of information, (2) propose a special-purpose system for communicating presence information in mobile instant messaging systems and analyze coherence of information

in this system, and (3) present an approach for the steady state performance analysis of EID systems with finite buffers that does not require off-line simulation.

Chapter 3

Epidemic Dissemination of Index Information

The distributed resolution of application specific identifiers denoted as *keys* to application specific data items denoted as *values* is one of the fundamental problems in the design of distributed applications. In the Internet the task of key-to-value lookups is performed based either on the client/server or the peer-to-peer paradigm. This chapter explores the task of implementing key-to-value lookups in MANET. It presents a general-purpose distributed lookup service denoted as *Passive Distributed Indexing* (*PDI*). Besides mechanisms for key-to-value lookups based on dissemination of index entries using a finite buffer at each node, PDI incorporates mechanisms for consistency maintenance. The performance of PDI is demonstrated for several mobile applications in a simulation study using the network simulator ns-2. Finally, implementation and security issues of PDI are discussed. The basic concept of PDI has been introduced in an article published in the proceedings of the *IEEE Conf. on Peer-to-Peer Computing* (*P2P 2002*) [LW02a]. The PDI consistency mechanisms have been presented at the *Int. ACM Workshop on Data Engineering for Wireless and Mobile Access* (*MobiDE 2003*) and has received the *best paper award* [LW03]. An extended version of the paper featuring a complete description of the PDI protocol has been published in a special issue of the *ACM SIGMOBILE Mobile Computing and Communication Review* (MC^2R) on mobile data management [LW04a]. Some aspects of employing PDI for disseminating presence information in a mobile instant messaging system have been discussed in a publication at the *14. GI / ITG Fachtagung Kommunikation in Verteilten Systemen* [LW05a].

3.1 General-purpose Lookup Services

Mobile computing devices such as Personal Digital Assistants (PDAs), mobile phones, laptops, and embedded sensing devices are typically managed using service discovery protocols such as the Service Location Protocol [GPVD99]. Many distributed applications running on mobile devices require a more general resolution of application-specific keys to application-specific values than just service discovery, a functionality provided by a *lookup service*. In a cellular mobile network, service discovery and lookup services can be implemented using a centralized index located at a well-known network address. In MANET, the lack of a fixed infrastructure, as well as intermittent connectivity due to node mobility hampers the employment of a centralized index for service location and lookup services.

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 P2P systems as described in Section 2.1, e.g., [RFH+01], [SMK+01]. Castro, Greenstein, Muntz, Biskidiuan, Kermani, and Papadopouli proposed the VIA protocol, which enables location of application data across multiple service discovery domains, using a self-organizing hierarchy [CGM+01]. Sun and Garcia-Molina recently 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 [SGM03]. The paper discusses various design alternatives for a partial lookup service in the Internet. However, none of these papers consider distributed lookup services for MANET. In the remainder of this section, we formally specify the functionality provided by a general-purpose lookup service.

Adopting the terminology introduced in [SGM03], a general purpose lookup service maintains the set $\mathcal{S} = \{(k, \mathcal{V}(k))\}$, where k denotes a key and $\mathcal{V}(k)$ denotes a

Algorithm 3.1 *place*$(k, \{v_1, \ldots, v_h\})$

if $k = k'$ for some $(k', \mathcal{V}(k')) \in \mathcal{S}$ **then**
  $\mathcal{V}(k') \leftarrow \mathcal{V}(k') \cup \{v_1, \ldots, v_h\}$
else
  $\mathcal{S} \leftarrow \mathcal{S} \cup (k, \{v_1, \ldots, v_h\})$

Algorithm 3.2 *lookup*(k)

if $k = k'$ for some $(k', \mathcal{V}(k')) \in \mathcal{S}$ **then**
  return $\mathcal{V} \subset \mathcal{V}(k)$
else
  return $\emptyset$

set of values matched by key k. The set $\mathcal{S}$ is denoted as *index* that is implemented in a distributed fashion. The pair (k, v) of a key k and a matching value $v \in \mathcal{V}(k)$ is denoted as *index entry*. Note that in general there exists a *many-to-many matching* between keys and values. That is, a key k may match multiple values, i.e., for the cardinality of the set $\mathcal{V}(k)$ holds $|\mathcal{V}(k)| \geq 1$, and a value might be matched by multiple keys k, l, that is, it may hold $\mathcal{V}(k) \cap \mathcal{V}(l) \neq \emptyset$ for keys $k \neq l$. A general-purpose lookup service should support the basic operations for inserting index entries into the distributed index, for looking up index entries by a key, and for deleting index entries from the distributed index. The semantic for inserting an index entry into the index $\mathcal{S}$ by the operation $place(k, \{v_1, \ldots, v_h\})$ is shown in Algorithm 3.1. Due to intermittent connectivity in a MANET, a lookup operation, $lookup(k)$, may not return all values $v \in \mathcal{V}(k)$. Following [SGM03], lookup services that do not return the entire set $\mathcal{V}(k)$ when queried for key k are denoted as *partial lookup services*. That is, they implement a partial lookup operation which guarantees that at least a certain minimum number of matching values is always returned. In MANET, even such a lower bound cannot be guaranteed. Hence, a general-purpose lookup service for MANET should use a best-effort approach to return the set of matching values $\mathcal{V}(k)$ where each value $v \in \mathcal{V}(k)$ is returned with a sufficient high probability. For ease of exposition, we denote this best-effort partial lookup operation as $lookup(k)$ with the semantic shown in Algorithm 3.2. As last operation, a general-purpose lookup service should be able to remove index entries from the distributed index. That is, it should implement an operation $delete(k, v)$ as shown in Algorithm 3.3. In the remainder of this chapter, we present a system that implements the operations *place*, *lookup*, and *delete*.

3.2 Passive Distributed Indexing

3.2.1 Rationale and Basic Functionality

We consider a system consisting of several mobile nodes, e.g., mobile users equipped with notebooks or PDAs and wireless network interfaces as illustrated in Figure 3.1.

Algorithm 3.3 $delete(k, v)$

```
if k = k' for some (k', V(k')) ∈ S and v ∈ V(k') then
    V(k) ← V(k) − v
else
    do nothing
```

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 total area. Thus, the nodes must cooperate and form a MANET to enable efficient communication. Subsequently, we assume IEEE 802.11 in the ad hoc mode as underlying radio technology. However, we point out that the approach presented here could be employed on any radio technology that enables broadcast transmissions inside a node's radio coverage.

We present the *Passive Distributed Indexing* (*PDI*) protocol that implements a general-purpose lookup service for mobile applications according to Section 3.1. 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 the 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 URLs. Opposed to DHT systems such as [RFH+01], [SMK+01], PDI does not require a one-to-one matching between keys and values. However, some mechanisms implemented in PDI require that a value is unique in the system. That is, the value 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 i_n for node n. For example, the node identifier i_n may be derived from the node's IP or MAC address. For ease of exposition, we abbreviate the unique value given by the pair (v, i_n) just by v.

A node n implements *place*$(k, \mathcal{V})$ by inserting index entries of the form (k, v) for $v \in \mathcal{V}$ into a data structure called *local index*. In Figure 3.1, 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 3.1 is the origin node of the index entry (k, v). A key k can be resolved to a value v, if (k, v) is currently supplied to the PDI system.

Nodes implement *lookup*(k) by sending Query messages (see Figure 3.1 (a)). A node that sends a Query message is denoted as *inquiring node*. Query messages contain the key to resolve and 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 transmission range of the inquiring node receive a Query message. Each of these nodes may generate a Response message. A Response message contains the key k extracted from the Query message and all matching values $v \in \mathcal{V}(k)$ that are currently stored by the responding node either the local index or a second data structure called *index cache*. To enable efficient epidemic dissemination of information, PDI Response messages are also sent to the IP limited broadcast address 255.255.255.255 and a well-defined

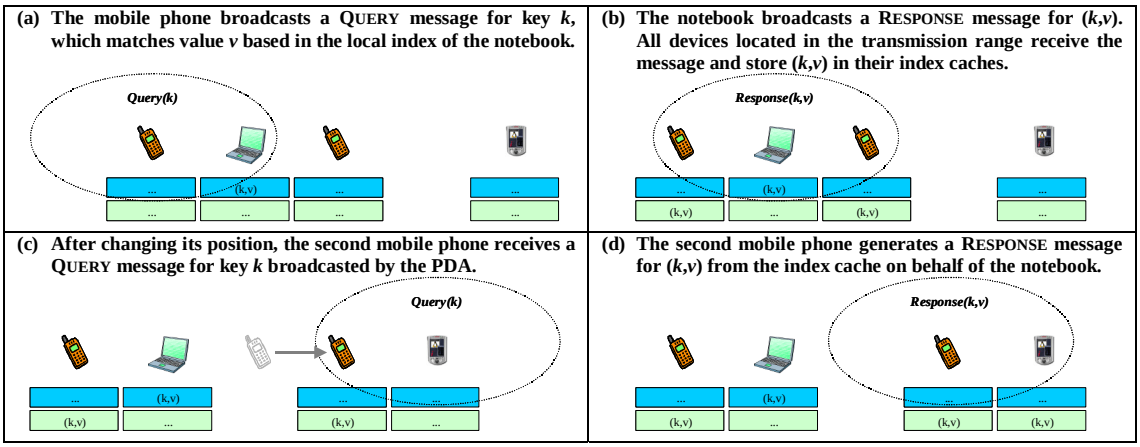

Figure 3.1. Illustration of epidemic information dissemination with PDI

port. Thus, all mobile nodes within the transmission range of the responding node overhear the message (Figure 3.1 (b)). All mobile nodes that receive a RESPONSE message including the inquiring node extract the key k and all matching values v and store the index entries (k, v) in the index cache (see Figure 3.1 (b)). In Figure 3.1, 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 transmission range of the inquiring node (see Figures 3.1 (c)) and 3.1 (d)).

The index cache size is limited to a maximum number of B_{index} entries adjusted to the capabilities of the mobile device. A replacement scheme is employed if a mobile device runs out of index cache space. It has been shown that the *Least Recently Used* (LRU) replacement scheme performs well for managing finite buffers in replication-based P2P systems for the Internet [TK05]. Thus, PDI uses LRU. LRU manages the cache as a stack. Upon access to an item in the buffer that is located at stack position j, the item is moved to the top of the stack, and all items on positions $1, \ldots, j-1$ are pushed down by one position. If an item that is not found in the buffer must be inserted, it is inserted at the top of the stack, and all items are pushed down by one position, where the item on position B (i.e., the least recently used item) is removed from the buffer.

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. Due to node movement and overhearing RESPONSE messages of neighboring nodes, index entries are disseminated within the network without costly global communication. However, opposed to the approaches taken by SWIM and PILOT as recalled in Section 2.4, information is only transferred when actively requested by a node. In

fact, PDI builds and maintains a distributed index in a passive way.

Note that PDI might return no results to a query for key k even in case an index entry (k, v) is currently supplied. This may occur if the corresponding query neither reaches the origin node of the index entry (k, v) nor another node storing (k, v) in its index cache. We refer to such an unresolved query as a *false miss*. For the application using PDI, there are two ways to resolve false misses: First, the application can tolerate and simply ignore them. Second, the application has to resort to a centralized lookup service via the cellular infrastructure of a mobile network as fallback. Since typical query streams for lookup services possess a high degree of temporal locality, e.g., as observed for P2P file sharing systems [Sri01], [KLVW04] and Internet search engines [XO02], PDI produces only few false misses for such applications. For further reduction of false misses, PDI implements a message forwarding mechanism to enable communication beyond a node's transmission. The mechanism is discussed in Section 3.2.2. Recall that responses derived from the index caches only contain index entries from remote hosts. Such responses may be out of date under certain circumstances. For instance, when a node stops supplying an index entry (i.e., the pair (k, v) is removed from the local index of the node), copies of the index entry remain in the index caches of other nodes and may be included in Response messages. We refer to an index entry contained in a Response message as *fresh* or *up-to-date*, if it is currently stored in the local index of the origin node. Otherwise, the index entry is denoted as *stale*. Stale index entries obviously yield incoherent search results and even disseminate in the system. As shown in Section 3.4.4, hits for stale index entries constitute a significant fraction of all results received in response to a query if no invalidation mechanism is used. Opposed to all previously proposed EID systems, PDI incorporates consistency mechanisms that can cope with both intermittent connectivity and information modification. These mechanisms are discussed in Sections 3.2.3 and 3.2.4, respectively. Since mobile applications exhibit both sources of inconsistency, PDI implements an hybrid approach that combines both invalidation mechanisms.

3.2.2 Selective Forwarding for Extending the Radio Coverage

Recall that all PDI messages are sent to the limited broadcast address and received only by the nodes located inside the transmission range 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. To overcome the limitations of small transmission ranges, packet forwarding is a common technique in MANET. That is,

packets are exchanged between source and destination using one ore more intermediate hops as relays. Several approaches for multi-hop communication in MANET have been proposed. Sophisticated protocols for unicast or multicast communication use control messages to set up sequences of nodes (or routes) that are used by subsequent data packets. Using such a stateful approach is out of scope for a lightweight distributed lookup service for two reasons. First, communication occurs infrequently, so that most routes will only be used for a single query. Second, the size of PDI messages is roughly equal to the size of control messages, resulting in a high relative overhead for route setup. A stateless approach to one-to-many communication constitutes flooding. For flooding a packet through the network, each node that receives the packet forwards it exactly once. Since every neighbor of the node receives the packet and forwards it, uncontrolled flooding can generate an infinite number of duplicate transmissions of the same packet. To avoid duplicate transmissions, a unique tag is included in every packet, e.g., given by a unique source identifier (ID) and a sequence number. Thus, a node can check on reception whether it has received this packet already and discard duplicates. Furthermore, the scope of flooding may be limited by including a *time-to-live* (TTL) field into each packet, which indicates the maximum number of hops the packet may be relayed. A node decrements the TTL field on packet reception and does not forward the packet, if TTL is equal to 0.

Based on the observation that blind flooding generates a high amount of redundant packets and increases the congestion of the wireless medium, several approaches for efficient flooding have been proposed and evaluated, e.g., [NTCS99], [TNS03], [YGK03]. Protocols for efficient flooding can be classified as heuristic-based, i.e., each node decides locally whether to forward a message based on simple metrics, or topology based, i.e., some nodes are selected as relays based on their position in the network topology. Opposed to these approaches, PDI includes a flooding mechanism that controls forwarding based on the content of a message. The mechanism is illustrated in Figure 3.2. Query messages are flooded with a time-to-live 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 show in Section 3.4 that $TTL_{query} = 2$ yields sufficient performance in many application 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 only of keys. Thus, Query messages are small and may be flooded without significantly increasing network load (see Figure 3.2 (a) and 3.2 (b)). In contrast, Response messages

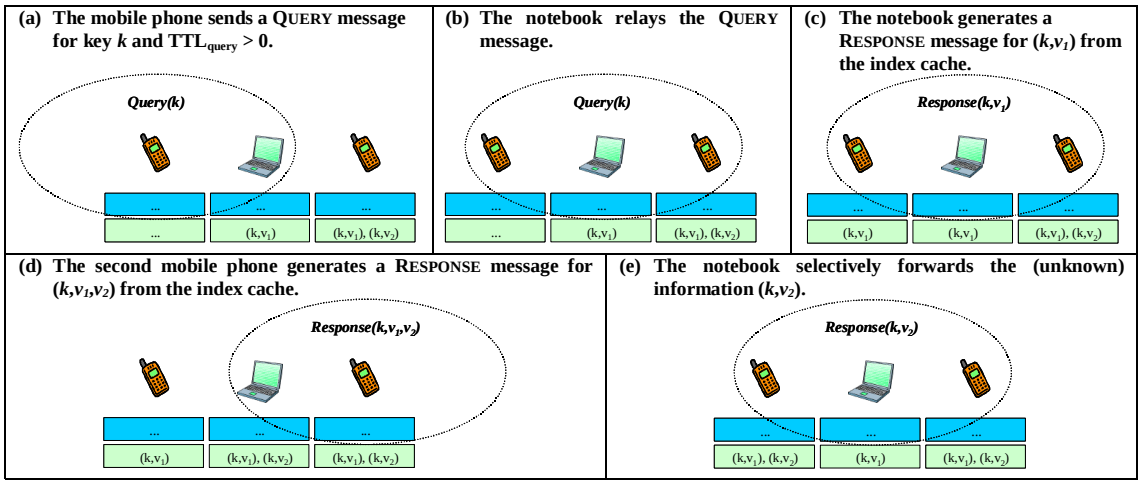

Figure 3.2. Message forwarding with PDI

can contain numerous values that may each have a considerable size, depending on the application using PDI. Therefore, flooding of complete RESPONSE messages significantly increases 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 denoted as *selective forwarding*. That is, each node that receives a RESPONSE message searches the index cache for each index entry contained in the message (see Figure 3.2 (d)). If an entry is found, in most cases the node has already generated a RESPONSE message for this index entry (e.g., as shown in Figure 3.2 (c)). 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 3.2 (e)).

3.2.3 Configurable Value Timeouts for Dealing with Intermittent Connectivity and Node Failures

The basic concept of PDI as described in Sections 3.2.1 and 3.2.2 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 such circumstances 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 coherency without the need to contact the source of the cached information. Examples include the invalidation of cached DNS records in the domain name system [Moc87] or the invalidation of cached

Web documents in WWW caches [CL98]. To achieve both maximum consistency and a sufficient number of cache hits, it is crucial for a timeout-based invalidation mechanism that the timeout durations are chosen appropriately. In DNS, the origin DNS server specifies the timeout duration for each entry. In web caching, the adaptive TTL algorithm calculates the timeout duration from the last modification time of a value at the origin server. Note that both approaches rely on information directly received from the origin server. In contrast, due to the epidemic dissemination, most index cache entries are extracted from responses comprising entries of other index caches, i.e., without direct contact to the origin node. Thus, PDI defines the concept of *value timeouts* that are disseminated epidemically to approximate the most recent information about the state of an index entry at the origin node.

Value timeouts limit the time for which any index entry (k, v) with a given value v is stored in an index cache. When receiving a Response from the origin node of (k, v), the corresponding value timeout is reset. Let $a_{(k,v)}$ be the time elapsed since (k, v) has been extracted from a Response message generated by its origin node. For all index entries (k, v) in the index cache of a node n, we define the age a_v of value v as $a_v = \min\left(a_{(k,v)}\right)$, i.e., the time elapsed since the most recent Response message of this kind has been received. If at a node holds $a_v > TO_{value}$ for the given timeout value TO_{value}, all pairs (k, v) are removed from the index cache. PDI implements only one timeout per value v rather than an individual timeout for each index entry (k, v). This is because in most applications the fact that one index entry (k, v) for a given v expires 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 $(keyword_i, URL)$ is removed when the file specified by the URL is withdrawn from the system. Thus, all other pairs $(keyword_j, URL)$ will also expire. For ease of exposition, we use a global timeout value TO_{value} for all values in the system in the remainder of this chapter. Note that depending on the application the concept of value timeouts can be easily extended to individual timeout durations for each value v. Such duration may be included in a Response message generated by the origin node.

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. This is the time at which a response for this value was generated 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 in 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 requirement for synchronizing clocks of all participating nodes.

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

In addition to the scenarios described above, a node produces stale index entries by modifying information. That is, it executes $delete(k, v)$ to remove an index entry (k, v) from the local index. In a worst-case scenario, a node leaves the system and all index entries supplied by the node expire at the same time. 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 value timeout duration TO_{value}, this straightforward solution may cause severe inconsistency, especially if TO_{value} 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 [LH86], or the invalidation of documents in web caches [LCD00]. 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 where all copies of an item reside and that all sharers are reachable. In contrast, in a mobile environment consisting of nodes with limited resources, neither connectivity of nodes cannot be guaranteed nor directories for all cached copies of a shared item can be maintained. To address these constraints, PDI defines the concept of *lazy invalidation caches* implementing an explicit invalidation of values by epidemic dissemination of Invalidation messages.

As basic idea of the explicit invalidation mechanism, a node removes all index entries (k, v) from the index cache when it receives an Invalidation message for the value v. Flooding with a time-to-live TTL_{inv} as described in Section 3.2.2 is a

straightforward way to propagate Invalidation messages. Unfortunately, network partitions may hinder that all nodes that store a copy of the value are reached by the Invalidation message. Subsequently, stale index entries remain in the index caches of these nodes. Note that these index entries will be redistributed in the system due to the epidemic dissemination. We show in Section 3.4.4 that even repeated flooding of Invalidation messages does not significantly reduce the number of hits for stale index entries [LW03].

This observation is consistent with [DGH+87] which reports that deleted database items "resurrect" in a replicated database environment in the Internet due to epidemic data dissemination. In [DGH+87], 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 mostly passive (or "lazy") approach for the epidemic dissemination of Invalidation messages, which is illustrated in Figure 3.3. For the initial propagation of an Invalidation message by the origin node, we rely on flooding as described above (Figure 3.3 (a)). Each node maintains a data structure called *lazy invalidation cache*, which is drawn as a third box below the mobile devices in Figure 3.3. 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 3.3 (b)). 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 contributes to the dissemination of Invalidation messages, so that distribution of information and Invalidation messages is separated. To enable the epidemic propagation of the Invalidation message, a node scans the invalidation cache for all values contained in an overheard response message (Figure 3.3 (c)). If a value v is found, the node generates an Invalidation message for v (Figure 3.3 (d)). The Invalidation message is not flooded through the entire network, but only with a certain scope TTL_{inv} similar to flooding Query and Response messages as described in Section 3.2.2. A node that receives a cached Invalidation message for value v stores v in the invalidation cache and removes 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 using PDI.

As the index cache size, the invalidation cache size is limited to a fixed number B_{inval} of values and LRU replacement is employed. We show in Section 3.4.4 that setting the invalidation cache size to a fraction below 20% of the index cache size achieves sufficient reduction of false hits assuming a reasonable rate of data modifications. Note that LRU replacement does neither guarantee that an invalidation cache

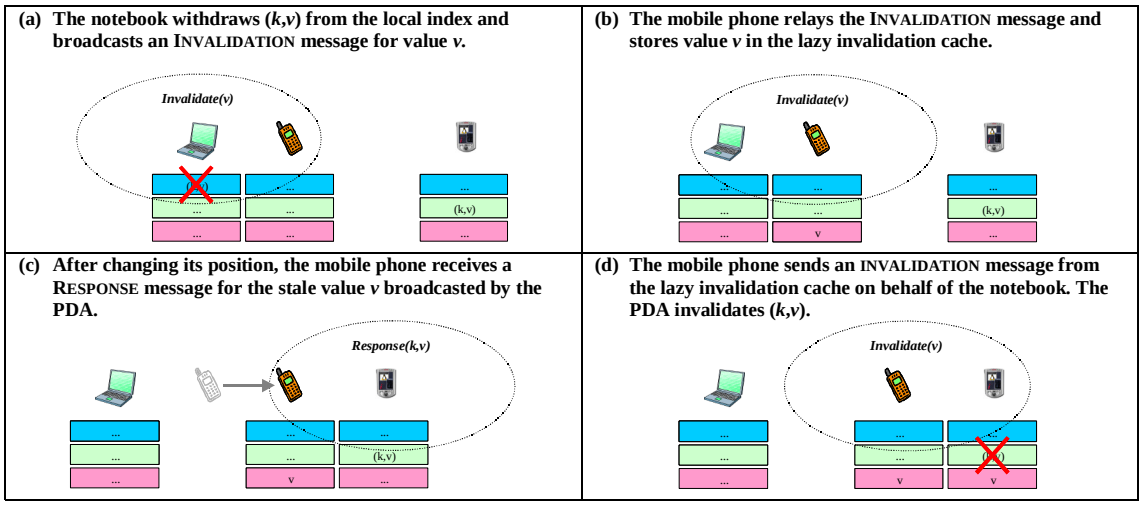

Figure 3.3. Epidemic dissemination of INVALIDATION messages using invalidation caches

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 3.2.3 yields an efficient mechanism to decide whether a result for a value is more recent than an INVALIDATION message.

3.2.5 Additional Protocol Features

In some of the applications described in Section 3.3, some values may be accessed very infrequently or the query activity of the mobile nodes may be low. In this case, the epidemic dissemination of the index entries may be very slow. To cope with this problem, a node may periodically send out RESPONSE messages for some entries of the local index without having received a QUERY message. Index entries in such RESPONSE messages are stored in the index caches of nearby mobile devices and, thus, disseminate in the system. We refer to such messages as *active response* messages. Active responses are broadcast periodically within intervals T_{AR} and have a TTL TTL_{AR}.

Additionally, in some of the applications described in Section 3.3 keys may match a huge number of values. Storing all index entries for such keys would certainly exceed the storage capacity of a mobile device. To cope with this problem, PDI can only return the top hits based on an arbitrary ranking of the values defined by the

application. We refer to this feature as *result threshold*. With the result threshold feature enabled, only the top Tr_{result} results matching a query are returned.

3.3 Workload Models for Mobile Applications

3.3.1 Lookup Requirements of Mobile Applications

In this section, we discuss how we can characterize the way a mobile application makes use of a lookup service such as PDI for simulation purposes. That is, we describe the *lookup requirements* of the mobile application. We assume that N_m users with mobile devices participate in the mobile application. Depending on the mobile application, these may be supplemented by N_f fixed devices that serve as data sources. Within a finite observation period, the application defines a finite set of keys $\mathcal{K}$ with cardinality $|\mathcal{K}| = K$. For ease of exposition, we associate the keys with natural numbers, i.e., $\mathcal{K} = \{1, ..., K\}$. Additionally, the application defines a set of values $\mathcal{V}$ with cardinality $|\mathcal{V}| = V$. Similar to keys, we also identify values with natural numbers, i.e., $\mathcal{V} = \{1, ..., V\}$. We denote keys by k and values by v with $1 \leq k \leq K$ and $1 \leq v \leq V$, respectively. We describe the lookup requirements of the mobile application by five basic functions denoted as *workload functions*:

- **Query function** $w_{query} : \mathcal{K} \rightarrow [0, 1]$. The function $w_{query}(k)$ denotes the probability mass function (pmf) of the distribution of queries in the request stream.
- **Selection function** $w_{select} : \mathcal{K} \times \mathcal{V} \rightarrow [0, 1]$. The function $w_{select}(k, v)$ denotes the probability that a key k matches a value v.
- **Pause function** $w_{pause} : I\!R^+ \rightarrow [0, 1]$. The function $w_{pause}(t)$ denotes the *probability density function* (*pdf*) of the pause times between two successive queries.
- **Expiration function** $w_{expire} : I\!R^+ \rightarrow [0, 1]$. The function $w_{expire}(t)$ denotes the pdf of value expiration times.
- **Arrival function** $w_{arrival} : I\!R^+ \rightarrow [0, 1]$. The function $w_{arrive}(t)$ denotes the pdf for the time between two consecutive arrivals of nodes. We assume that the system is in steady state. Thus, the departure function is equal to the arrival function.

Note that we use the *independent reference model* [Kin71], where for a query q_i holds $P(q_i = k) = w_{query}(k)$, for $1 \leq k \leq K$ and $i = 1, 2, \ldots$. That is, the key in query q_i is

independent of the keys used in previous queries q_j for $0 \leq j < i$. The independent reference model is commonly used to study the performance of caching and paging algorithms.

Subsequently, we define workload functions for characterizing four selected mobile applications, i.e., mobile P2P file sharing, mobile instant messaging, a mobile information portal, and disconnected web search. Note that for all these applications no complete workload model has been published in academic literature. Subsequently, we have to rely on intuitive assumptions for the definition of the workload functions. Note that the customized workload functions are sufficient for generating synthetic workloads as input for the simulation studies presented in Section 3.4.

3.3.2 Mobile Peer-to-Peer File Sharing

The operation of a P2P file sharing system requires two basic mechanisms: (1) a keyword-based search algorithm for searching for files shared by remote peers and (2) a mechanism for downloading files from the search results. PDI can effectively implement the search algorithm of a mobile P2P file sharing system; subsequent file downloads may be performed using standard MANET routing protocols or a cellular network infrastructure. In a P2P file sharing application, keys are given by keywords from the file name or some meta-data associated with a file. Values denote a location from which a file can be downloaded, e.g., given by a URL. Note that this definition implies a many-to-many matching between keys and values. Using PDI for mobile P2P file sharing obviously requires invalidation mechanisms for two reasons: First, a mobile device may stop sharing a file, e.g., by deleting it from the local disk. In this case, the value denoted by this file expires. Second, a mobile node depart from the system, e.g., due to a failure. In this case, all files shared by the user are removed from the system, and all corresponding values expire.

For defining the workload functions for P2P file sharing, we assume that both $\mathcal{K}$ and $\mathcal{V}$ are finite in an observation period of finite length T. Consistent with Section 2.2, we assume that the query function w_{query}^{P2P} for a P2P file sharing system can be approximated by a Zipf-like distribution with parameters α. Thus, the query function for P2P file sharing w_{query}^{P2P} is given by:

$$w_{query}^{P2P}(k) = \frac{1}{\sum_{j \in \mathcal{K}} j^{-\alpha}} k^{-\alpha} \tag{3.1}$$

For the definition of the selection function, we use the observation from [YGM01] that the probability that a query k matches a value v can be modeled by an exponential distribution in an OpenNapster P2P file sharing system. Thus, the selection

function for P2P file sharing w^{P2P}_{select} is given by:

$$w^{P2P}_{select}(k, v) = \beta e^{-\beta k} \tag{3.2}$$

Note that following [YGM01], the selection function is independent of the value v.

For defining the pause function, we assume that the intervals between two successive queries by a node are exponential distributed with parameter λ. Thus, for $t > 0$ the pause function for P2P file sharing w^{P2P}_{pause} is given by:

$$w^{P2P}_{pause}(t) = \lambda e^{-\lambda t} \tag{3.3}$$

For definition of the expiration function, we assume that each value expires exactly once in an observation period of length T. Thus, for $0 \leq t \leq T$ the expiration function for P2P file sharing w^{P2P}_{expire} is given by:

$$w^{P2P}_{expire}(t) = \frac{1}{T} \tag{3.4}$$

For the same observation period T as used for the definition of the expiration function, we assume that on average dN_m mobile nodes depart from the system with $0 \leq d \leq 1$. Subsequently, we assume that the arrival function $w^{P2P}_{arrival}$ is given by an exponential distribution with parameter $\frac{dN_m}{T}$. Then, the number of arrivals in the observation period T is Poisson distributed with parameter $T\frac{dN_m}{T} = dN_m$ and mean dN_m, matching the assumption. That is for $0 \leq t \leq T$ the arrival function for P2P file sharing $w^{P2P}_{arrival}$ is given by:

$$w^{P2P}_{arrival}(t) = \frac{dN_m}{T} e^{-\frac{dN_m}{T}t} \tag{3.5}$$

All workload parameters for the P2P application are shown in Table 3.1

Parameter	Value
N_M	20 - 110
N_F	0
K	512
V	$16 \cdot N_m$
α	0.9
β	1/100
λ	1/120
d	0.3

Table 3.1. Workload parameters for the P2P application

3.3.3 Mobile Instant Messaging

Instant messaging has long been one of the Internet's highly popular consumer applications, thanks to its combination of real-time communication, the ability to know in advance if people are available to chat, and the ease of use. A basic instant messaging (IM) system consists of three components: (1) an authentication server, which determines if clients trying to enter the system are permitted to do so, (2) a presence server, which identifies which contacts are available for messaging, and (3) a message server, which handles the communication process. Presence technology constitutes an integral part of an IM system, since it lets users determine if their contacts are online, signed onto the IM applications, and ready to communicate. Presence technology is used in a number of applications other than classical IM systems, e.g., in numerous computer-supported-cooperative-work applications. In fact, the 2003 release of Microsoft's Office constitutes one of the most prominent examples of a presence-enabled application. Note that it is possible to request presence information on demand in a reactive fashion. However, all state-of-the-art IM systems display presence information proactively. The protocol design for disseminating presence information in the Internet has been matured and organizations such as the IETF and the Jabber software foundation have developed the protocol proposals SIMPLE [DRS00] and XMPP [SA04b], [SA04a]. However, due to the lack of fixed infrastructure, dynamic network topology, and intermittent connectivity the dissemination of presence information in MANET poses a challenging research problem. In this section we customize the general purpose PDI system to disseminate presence information in a mobile IM system. A system that is tailored to the dissemination of presence information is presented in Chapter 4.

PDI can be effectively employed to disseminate presence information in a mobile IM system; additional features of IM systems such as message-based communication may be implemented using, e.g., a standard MANET routing protocol. Note that even if a contact resides in another network partition and communication via MANET routing is not possible, it is still beneficial to know its current presence state since it may be available for communication using an alternative network technology, e.g., a cellular data connection. In a mobile IM system based on PDI, each mobile user contributes his current presence state to the system as only value, which is matched by a key given by the users IM identifier. The IM identifier is a unique key that is assigned to a user when registering to the IM system. Thus, we have a one-to-one matching between keys and values. Using PDI in a mobile IM obviously requires invalidation mechanisms, since a user occasionally changes his presence state or departs from the system. That is, the value contributed by the user expires and is

replaced by a new value. Note that with this semantic of keys and values, queries for a user with few contacts will rarely occur and hit rates for this user will be low. Thus, in an IM application the active response feature should be enabled to disseminate the presence state of such users.

We define a simple model of the workload generated by a mobile IM application here that is sufficient to evaluate the performance of PDI with the IM application. A more sophisticated model of an IM workload is used in Chapter 4. We define $\mathcal{K} = \{1, \ldots, N_m\}$, i.e., the user's IM identifiers are given by natural numbers that provide the keys. Furthermore, define $\mathcal{V} = \{(u, s)\}$ with $1 \leq u \leq N_M$ and $S = 1, 2, \ldots$. When user u changes his current presence state, he sends an INVALIDATION message for (u, s) and subsequently contributes $(u, s+1)$ to the system. We assume that each mobile user maintains a list of his favorite contacts. Each user u is on the contact list of each other user with probability Cu^{-1}, where C is a constant. I.e., the social contacts are described by a Zipf distribution [Zip49]. An IM client periodically polls the presence state of each contact on a user's contact list and displays the current presence state of a user. Assume that the contact list of user n is given by the vector $\mathbf{c}$, where the dimension of $\mathbf{c}$ is the length $l(n)$ of the contact list and $\mathbf{c}(i)$ for $0 \leq i < l(n)$ is the i-th user on the contact list. Based on this model, the query function w^{IM}_{query} for the query q send by user n is given by:

$$w^{IM}_{query}(k) = \begin{cases} 1, & k = \mathbf{c}\,(q \bmod l(n)) \\ 0, & else \end{cases} \tag{3.6}$$

Recall that each user contributes his presence state as only value, which is matched by the user's IM identifier. Thus, we have a one-to-one matching between keys and values and the selection function is trivial. Given that each contact is polled once within a period T_p, for $t > 0$ the pause function w^{IM}_{pause} of the IM application is given by:

$$w^{IM}_{pause}(t) = \begin{cases} 1, & t = \frac{T_p}{l(n)} \\ 0, & else \end{cases} \tag{3.7}$$

Parameter	Value
N_M	20 - 110
N_F	0
K	N_m
V	N_m
γ	0.9
d	0.3

Table 3.2. Workload parameters for the IM application

For the expiration function, we assume that users change presence state in exponentially distributed intervals with parameter γ. Thus, for $t > 0$ the expiration function w^{IM}_{expire} for a mobile IM application is given by:

$$w^{IM}_{expire}(t) = \gamma e^{-\gamma t} \tag{3.8}$$

Furthermore, for the arrival function we assume exponentially distributed arrivals similar to the P2P file sharing application. Thus, the arrival function $w^{IM}_{arrival}$ is given by:

$$w^{IM}_{arrival}(t) = \frac{dN_m}{T} e^{-\frac{dN_m}{T}t} \tag{3.9}$$

All workload parameters for the IM application are shown in Table 3.2

3.3.4 Mobile Information Portals

Mobile devices are frequently used to provide value-added services, such as access to *mobile information portals* (*IPs*). As an example, a shopping mall might enable customers to locate shops based on queries for certain products using their mobile phones or PDAs. In this application, data is provided by some fixed infostations with high storage capacity and large radio coverage due to (virtually) unlimited energy resources. In general, a different consortium of shops operate each infostation. Thus, all infostations provide a different data set to the system. In contrast to the infostations, mobile nodes act only as clients and do not provide any data to the system. PDI can effectively implement such a mobile information portal. In this application, keys are given by product names or descriptive keywords while values are given by names and locations of shops. Note that as in P2P file sharing, there is a many-to-many matching between keys and values. Using PDI to implement a mobile information portal does not require invalidation mechanisms, since the assortment of a shop changes in intervals much longer than the duration of a visit to a shopping mall and mobile nodes do not supply data, that is, node departures do not affect index consistency. Instead, PDI faces the problem that customers only issue few queries during a visit, limiting the epidemic dissemination of information. To cope with this problem, the infostations may use the active response feature. Note that even if the radio coverage of the infostations does not cover the complete area of the shopping mall, the information portal is available in most places due to the collaboration of the mobile nodes.

To define the workload functions for the mobile information portal, consider a shopping mall with V shops, each shop providing an assortment of aN_p products,

$0 \leq a \leq 1$, out of a set of N_p products. In general, a product may be in the assortment of more than one shop. Again, we assume some locality in the products preferred by the users, thus, the query function w^{IP}_{query} for the mobile information portal is given by a Zipf-like distribution with parameter β:

$$w^{IP}_{query}(k) = \frac{1}{\sum_{j \in \mathcal{K}} j^{-\alpha}} k^{-\alpha} \tag{3.10}$$

Furthermore, we assume that each product is equally likely to be in the assortment of any shop. Thus, the selection function for the mobile information portal w^{IP}_{select} is given by:

$$w^{IP}_{select}(k, v) = a \tag{3.11}$$

As for P2P file sharing, we assume that query interarrival times for all mobile nodes are exponentially distributed with parameter λ, so that the pause function w^{IP}_{pause} for the mobile information portal is given by:

$$w^{IP}_{pause}(t) = \lambda e^{-\lambda t} \tag{3.12}$$

Recall that information of the information portal expires on a much longer time scale then a visitor's dwell time in the shopping mall. Thus, we do not define an expiration function and assume that the arrival function is given as for P2P file sharing, i.e., according to 3.5.

All workload parameters for the IP application are shown in Table 3.3

3.3.5 Disconnected Web Search

When browsing the Web, one of the most frequent activities is searching for content using a WWW search engine. Even sophisticated search engines return a high number of results that are not relevant to the users query. Thus, a mobile user connected to the Internet using a pay-per-volume cellular Internet connection has to pay for

Parameter	Value
N_M	30 - 300
N_F	4
K	10,000
V	10,000
α	0.9
a	0.05
λ	1/1800

Table 3.3. Workload parameters for the IP application

the transmission of information without even using them. A WWW search service that uses free spectrum technology and ad hoc collaboration of mobile nodes would imply high financial benefits for each user. Such services, denoted as *disconnected Web search* (*DWS*), can be implemented by employing infostations at hot-spots that offer access to the databases of a WWW search engine. In contrast to the mobile information portal, all infostations provide an identical set of data. Again, mobile nodes act only as clients, providing no data to the system but operating on the data provided by the infostations. PDI can effectively implement disconnected Web search. In this application, keywords from web pages constitute keys, while values are given by URLs. Again, using PDI for disconnected Web search does not require an invalidation mechanism, since mobile nodes do not supply data and most web pages change in intervals much longer than a web browsing session. The major difference between the DWS and the P2P applications constitutes the fact that there are much more keys and values in the DWS application domain. To cope with this huge amount of data, very large index caches would be required that would certainly exceed the capabilities of a mobile node. Note that all major search engines use a ranking to sort the returned hits according to their relevance to the query. In the most cases, a user looks only at the top 30 results. Thus, we can assume that an efficient PDI implementation of disconnected Web search uses the result threshold feature based on the ranking of the hits.

It has been shown that the distribution of user queries in Web search engines follows a Zipf-like distribution [XO02]. Thus, the query functions are identical to Equation 3.1. To consider the increased matching of keys and values, we use a Zipf-like distribution with parameter *beta* for the selection function. That is, the selection functions w_{select}^{DWS} for the DWS application is given by:

$$w_{select}^{DWS}(k,v) = \frac{h}{\sum_{j \in \mathcal{K}} j^{-\beta}} k^{-\beta} \tag{3.13}$$

Parameter	Value
N_M	100
N_F	2 - 16
K	$10,000$
V	$1,000,000$
α	1.2
β	1
h	40
λ	$1/120$

Table 3.4. Workload parameters for the DWS application

Here, h denotes the average number of keys matching a given value, with $h \leq \sum_{j \in \mathcal{K}} j^{-\beta}$.

We use a pause function similar to Equation 3.3. Furthermore, since Web documents expire on a much larger timescale than session durations of mobile nodes and index entry consistency is not affected by node arrivals and departures, we do not consider a expiration and arrival function for the DWS application.

All workload parameters for the DWS application are shown in Table 3.4

3.4 Performance Studies

3.4.1 Simulation Methodology and System Model

To evaluate the performance of PDI, we conduct simulation experiments using the network simulator ns-2. We developed an ns-2 application implementing the basic concepts of PDI, selective forwarding, value timeouts, and lazy invalidation caches as described in Section 3.2. 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. The settings of all configuration parameters of PDI for each of the mobile applications introduced in 3.3 are shown in Table 3.5. These settings are used for the experiments in this section if not stated otherwise. In the simulation studies we show that PDI can be configured to the demands of different applications by adjusting these parameters.

Besides the characteristics of the mobile applications that use the lookup service, the performance of PDI is affected by several characteristics of the mobile environment. We denote these characteristics as *system characteristics* in contrast to the *application characteristics* defined in Section 3.3. System characteristics include

Parameter	P2P	IM	IP	DWS
B_{index}	32 - 2048	16 - 128	32, 2048	32 - 2048
B_{inval}	128	256	0	0
TTL_{query}	4	4	4	4
TTL_{inv}	1	2	0	0
TO_{value}	$1000s$	$600s$	∞	∞
T_{AB}	∞	$120s$	$4s$	∞
TTL_{AB}	0	1	1	0
TR_{res}	∞	∞	∞	30

Table 3.5. Settings of PDI configuration parameters for different applications

the density of mobile nodes, the transmission range of the wireless communication interfaces, and the mobility model describing the movement of the mobile nodes.

For modeling system characteristics, we assume in accordance to Section 3.3 that N_m mobile and N_F fixed infostations participate in a mobile application. We configure the transmission power of the wireless interface to provide a transmission range R_M. In general the fixed infostations may use a different transmission power that leads to a transmission range $R_F > R_M$. Default transmission ranges are $R_M = 115m$ and $R_F = 230m$. In most experiments, we assume that the mobile nodes move in an area of size A according to the random waypoint (RWP) mobility model as described in Section 2.2. If not stated otherwise, the default parameters for the RWP mobility model are $v_{max} = 2m/s$ and $T_{hold} = 50s$. Recall that the RWP mobility model mimics the movement of pedestrians, which constitutes a conservative assumption for disseminating information by exploiting mobility.

We use two other mobility models for the evaluation of the IP application. First, we use the *Manhattan Grid* mobility model [dWG05]. With this mobility model, nodes move on predefined paths that are arranged similar to the streets in a urban grid. The number of vertical and horizontal streets is controlled by two parameters x_{grid} and y_{grid}, respectively. Node speed is chosen uniformly from $(0, v_{max}]$. When a node reaches a crossing, it goes straight on with probability $\frac{1}{2}$, or turns left or right, respectively, with probability $\frac{1}{4}$ each. If a node changes direction, it chooses a new speed from $(0, v_{max}]$, while speed is preserved when going straight on. We use an extention of the Manhattan grid model where a node that reaches a crossing may pause for time T_{pause} with probability P_{pause} to mimic stopping at a traffic light. A node that reaches the border of the simulation area turns and continues the movement into the opposite direction. The default parameters for the Manhattan Grid mobility model are $x_{grid} = y_{grid} = 10$, $v_{max} = 15m/s$, $P_{pause} = 0.5$, and $T_{pause} = 30s$.

As second additional mobility model we use the *Reference Point Group RPG* mobility model [HGPC99]. With the RPGM model, the mobile nodes move in G groups that cover each a circular area with radius r_g. Groups move according to the random waypoint model with $T_{hold} = 0$. Each mobile node is associated with a reference point uniformly chosen from the area covered by the group. The mobile nodes are placed at positions that are randomly chosen from a circular area with a displacement radius r_n around their reference point. The default parameters for the RPG mobility model are $G = N/5$, $v_{max} = 2m/s$, $r_g = 200m$, and $r_n = 50m$.

In all experiments, 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 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 $precision = 1 - SHR$. As additional performance measure related to result coherence, we measure the reduction of stale hits by the coherence efficiency $EF = 1 - \left(H_s/\hat{H}_s\right)$, where H_s denotes the number of hits for stale index entries without employing an invalidation mechanism. Additional performance measures constitute the *message volume*, i.e., the total size of all messages that have been sent by all nodes in the system, and (where applicable) the *infrastructure hit rate* IHR given by $IHR = H'_F/K_F$, i.e., the hit rate for results retrieved from a fixed infostation, where H'_F denotes the number of such results.

In all experiments, we conduct transient simulations starting with initially empty caches. For each run, the total simulation time is set to $T = 7200s$. To avoid inaccuracy due to initial warm-ups, we reset all statistic counters after a warm-up period of 10% of the simulation time. In all curves 99% confidence intervals determined by independent replicates are included in all figures. Since these confidence intervals are very tight, for some values their borders may not be visible.

3.4.2 P2P Application: Sensitivity to System Characteristics

In this section, we evaluate the performance of PDI with respect to the system characteristics and the P2P application. Thus, 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 T. The impact of consistency issues is evaluated in Section 3.4.4.

In a first experiment, we investigate the sensitivity of PDI to the number of nodes, i.e., the number of mobile nodes moving in the simulation area and participating in the P2P application. The results of this performance study are shown in Figures 3.4 and 3.5. Figure 3.4 plots the hit rate as a function of the number of nodes 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 number of nodes limits the epidemic information dissemination due to a lack of encounters between the nodes. For an increasing number of nodes, we observe three different effects. First, an increasing number of encounters between the nodes significantly fosters epidemic information dissemination for sufficient large

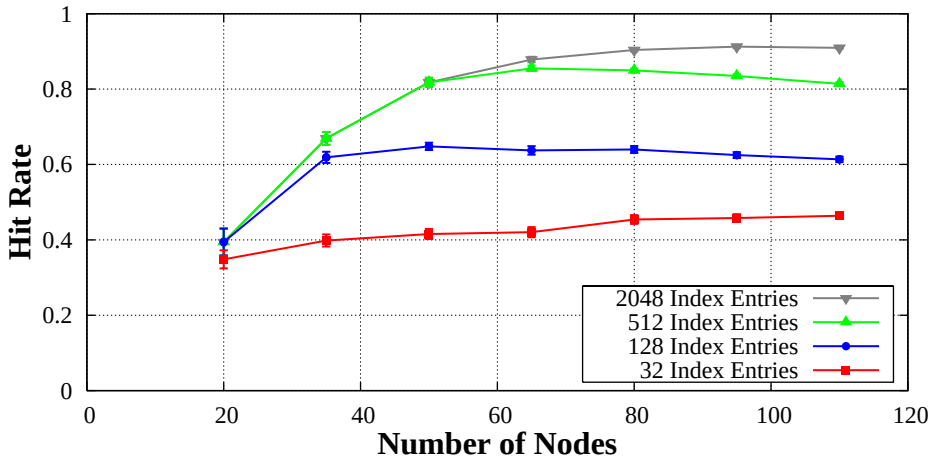

Figure 3.4. Recall vs. system size for different index cache sizes

index caches (i.e., 128 index cache entries or more). Second, connectivity increases with the number of nodes. Thus, selective forwarding increases hit rate even for small index caches (i.e., 32 index entries). Third, the hit rate for large caches decreases when the number of nodes passes a certain number of nodes. To understand this effect, recall that the overall number of values in the system increases with the number of nodes because each node contributes additional values to the lookup service. Furthermore, the keys in the 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 3.4 that epidemic data dissemination requires a sufficient number of nodes. 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 number of nodes, 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 number of nodes, as shown in Figure 3.5. Forwarding messages for more than four hops yields only marginal improvements in the hit rate due to limited connectivity. However, for an increasing number of nodes, the hit rate grows faster in systems with message forwarding enabled than in non-forwarding systems, since increasing connectivity favors selective forwarding. In environments with about 64 nodes, enabling selective forwarding improves hit rate by almost 30%. Nevertheless, non-forwarding systems benefit from a growing number of nodes as it

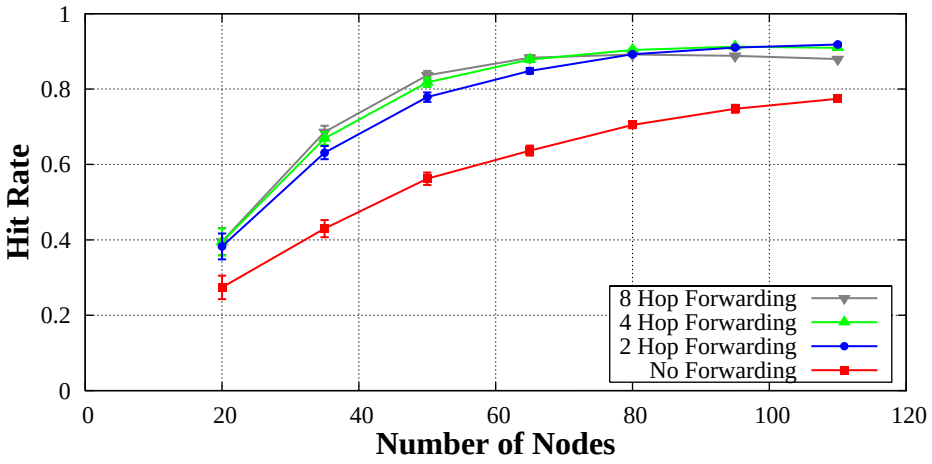

Figure 3.5. Recall vs. system size for different numbers of forwarding hops

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 decreases 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 successive queries. We provide further insight into this behavior in Chapter 5. We conclude from Figure 3.5 that message forwarding is beneficial in system environments showing a medium number of nodes, while in systems with a high number of nodes message forwarding should be disabled.

In a second experiment, we fix the number of nodes to $N = 64$ and 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 Figures 3.6 and 3.7. Figure 3.6 shows the hit rate as a function of the transmission range for different cache sizes. For a transmission range below $100m$, i.e., a radio coverage of about 1% of the simulation area, PDI does not gain sufficient hit rates despite of the index cache size. Here, in most cases broadcasted query messages are received only by a small number of nodes. Consistent with the results shown in Figure 3.4, given a reasonable size of the index cache and a transmission range of $115m$ (i.e., a radio coverage of about 4% of the considered area) PDI achieves a sufficiently high hit rate for P2P search queries. For larger transmission ranges, most queries 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 3.6 that for transmission ranges the number of participating devices must be high to enable the effective employment of PDI, whereas for large transmission ranges the system does not significantly benefit

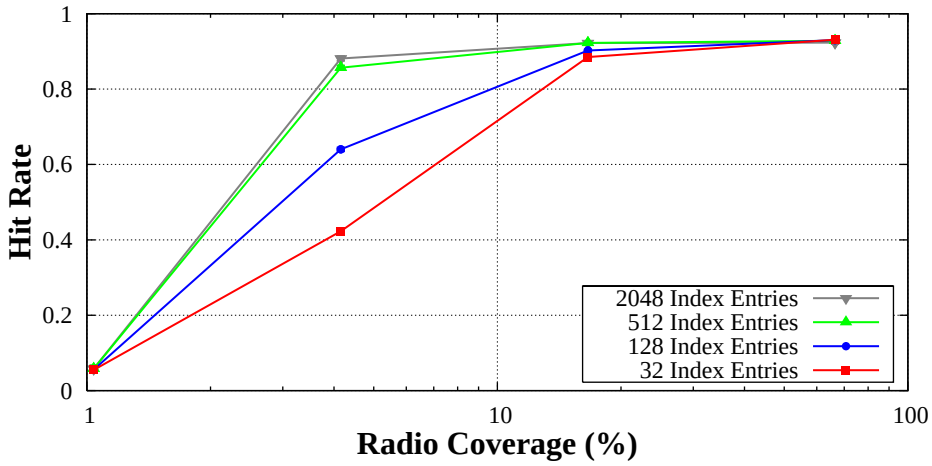

Figure 3.6. Recall vs. radio coverage for different index cache sizes

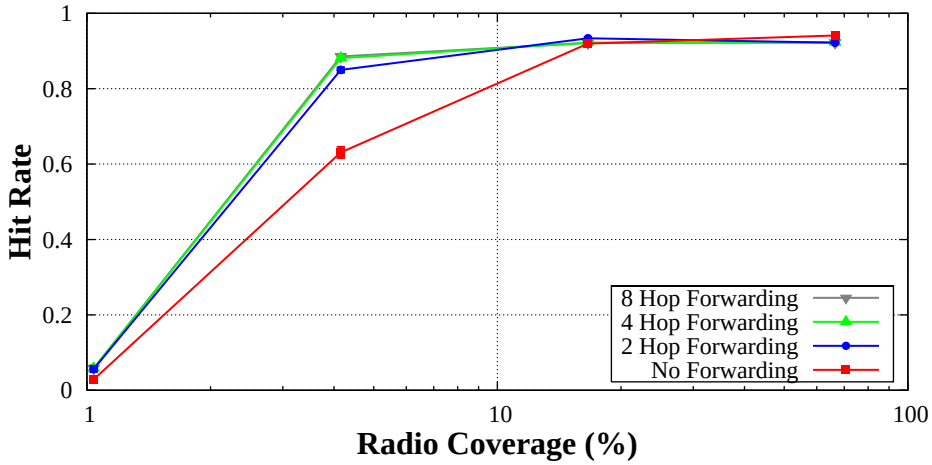

Figure 3.7. Recall vs. radio coverage for different numbers of forwarding hops

from caching index entries.

Figure 3.7 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 systems 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, forwarding of Response messages again reduces the variety of index cache content, since identical index entries are stored in the caches of all nodes that receive the message. We conclude from Figure 3.7 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 storing index entries in to many caches.

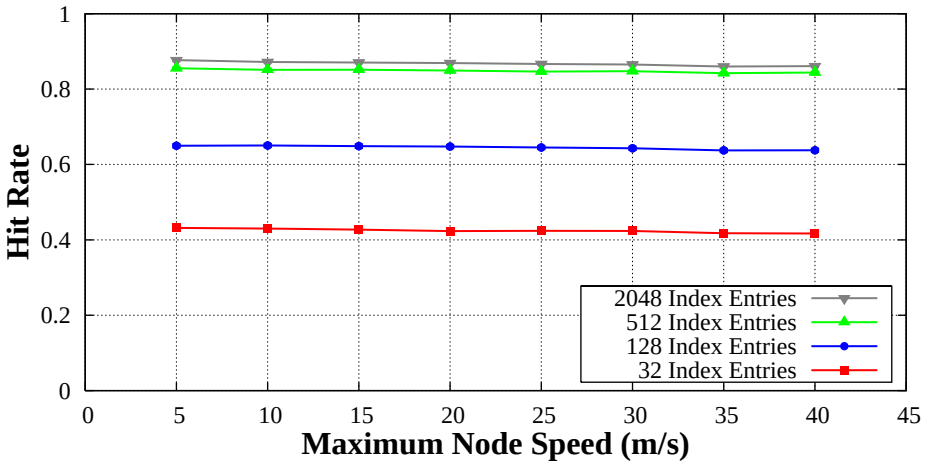

Figure 3.8. Recall vs. mobility for different index cache sizes

As final system parameter, we investigate the sensitivity of PDI to node mobility in Figures 3.8 and 3.9. Figure 3.8 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 the hit rate is almost independent of the 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 changes its 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 the total volume of transmitted PDI messages as shown in Figure 3.9. This figure assumes a binary message format with a size of 100 Bytes for a Query message and a size of 100 Bytes for each entry in a Response message. As increasing mobility fosters the epidemic information dissemination to some extend, message volume is reduced up to 25% by 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 are suppressed by the selective forwarding mechanism, resulting in substantial savings in message volume. We conclude from Figures 3.8 and 3.9 that PDI performs well for a reasonable degree of mobility, while high mobility increases the benefit of selective forwarding.

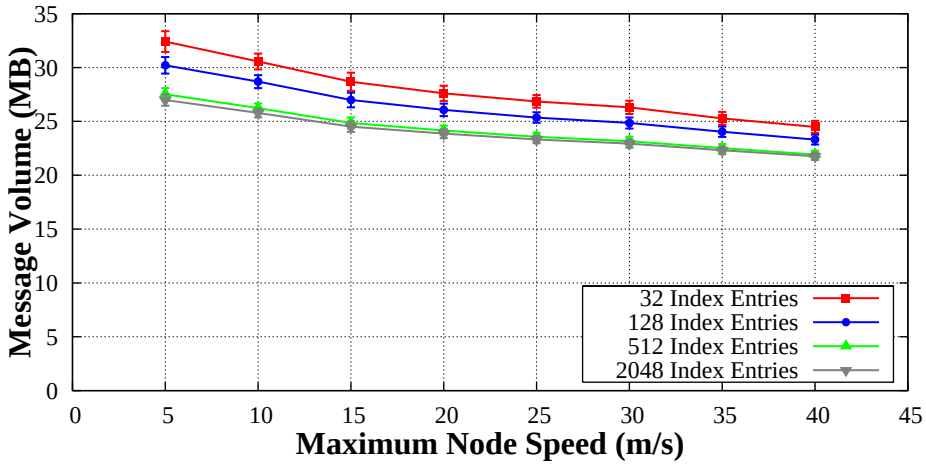

Figure 3.9. Message volume vs. mobility for different numbers for forwarding hops

3.4.3 P2P Application: Sensitivity to Application Characteristics

To evaluate the performance of PDI with respect to the characteristics of the application running on top of it, again, we use the P2P file sharing application and do not consider consistency issues originating from node departures and data modifications. As first application 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 the P2P application, we assume that the number of values is about 16, where each node in an instant messaging application only contributes a single value. For the disconnected Web search application, the overall number of values might be considerable larger, but does not depend on the number of nodes. To illustrate the impact of the number of values on the performance of PDI, we perform a sensitivity study based upon the P2P application.

Figures 3.10 and 3.11 plot hit rate as a function of the number of values contributed by each node for $N = 64$. We observe in Figure 3.10 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 3.4, 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

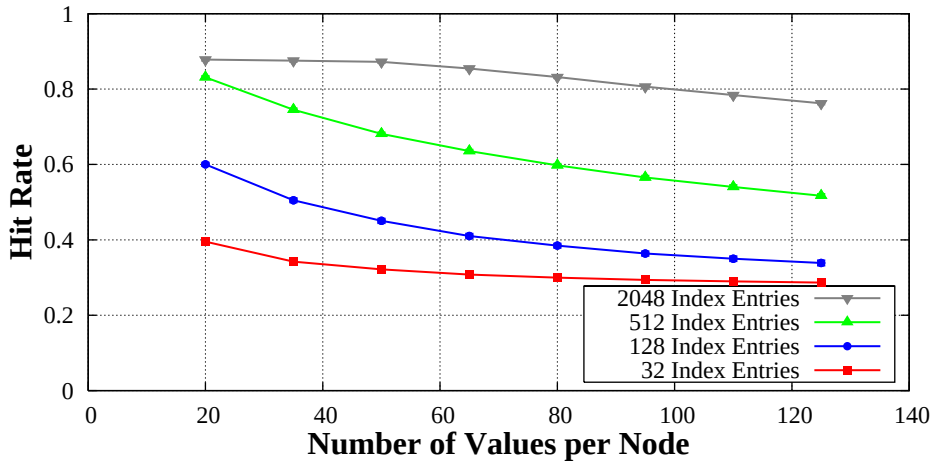

Figure 3.10. Recall vs. shared data for different numbers of forwarding hops

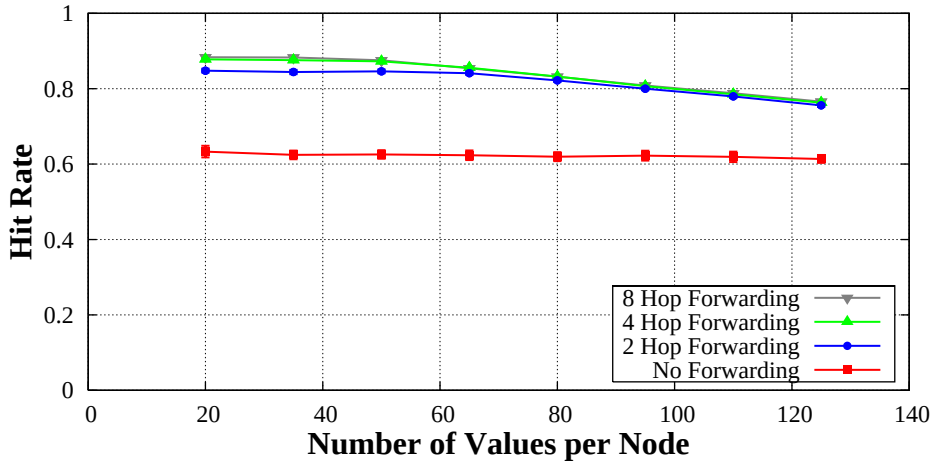

Figure 3.11. Recall vs. shard data for different numbers of forwarding hops

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 3.11. 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 Figures 3.10 and 3.11 that large index caches should be provided and selective forwarding enabled to handle large numbers of values.

In an experiment not shown here, we found that for the P2P application PDI is almost insensitive to the shape parameter of the matching function λ, 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

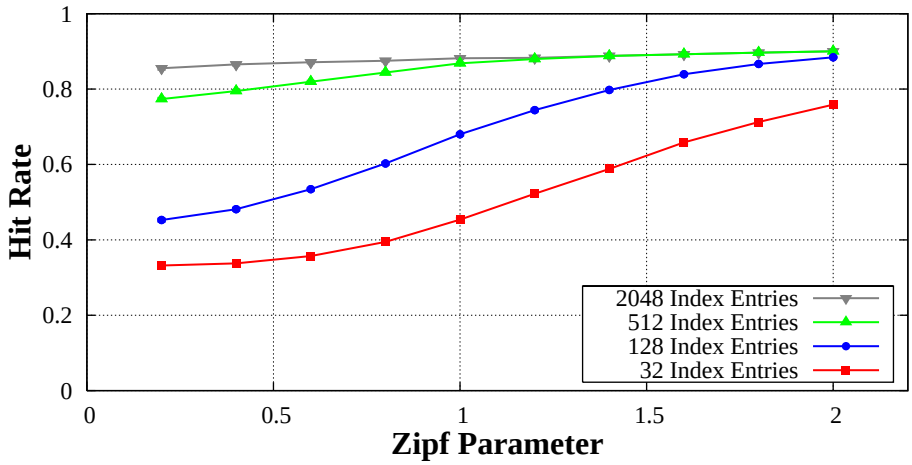

Figure 3.12. Recall vs. locality for different index cache sizes

for the P2P application the selection function is given by a Zipf-like distribution with shape parameter β. For $0 < \beta \leq 0.5$, the locality in the query stream is low, whereas $\beta > 1$ indicates high locality. The hit rate as a function of the shape parameter β is shown in Figures 3.12 and 3.13 for different index cache sizes and forwarding options, respectively. Figure 3.12 reveals that PDI is extremely sensitive to the 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 3.13 shows that PDI is most sensitive to the query locality if selective forwarding is disabled. With $TTL_{query} \geq 2$, however, PDI can combine results from several index caches and effectively cope with low locality. We conclude from Figures 3.12 and 3.13 that sufficient index cache size should be provided and 2 hop forwarding enabled if the application using PDI provides a low degree locality in the query stream.

3.4.4 P2P Application: Impact of Consistency Mechanisms

In this section, we determine the impact of value timeouts and lazy invalidation caches on the performance of PDI for the P2P application. Thus, opposed to Section 3.4.2 and 3.4.3, we have to consider document modifications according to the expiration function as well as node arrivals and departures. We assume that on a departure, a node does not invalidate the values it has supplied to the system, which constitutes a worst case assumption.

In a first experiment, we investigate the coherence of the index caches maintained by PDI with and without the invalidation mechanisms presented in Section 3. These

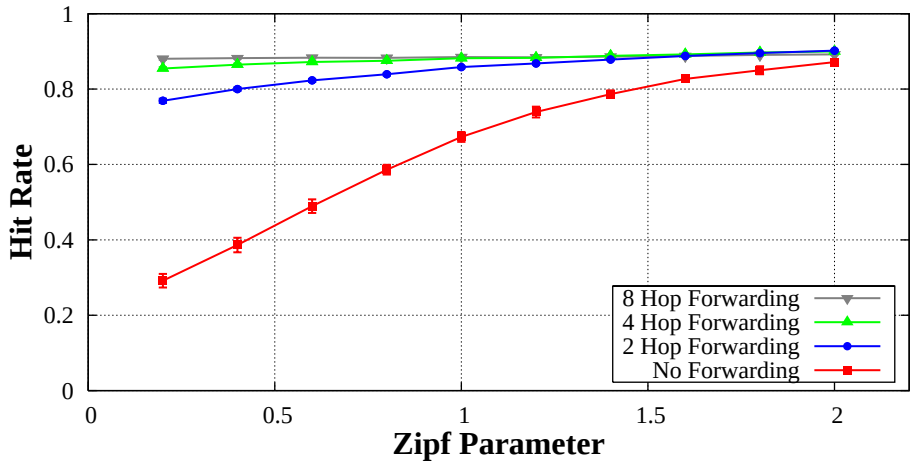

Figure 3.13. Recall vs. locality for different numbers of forwarding hops

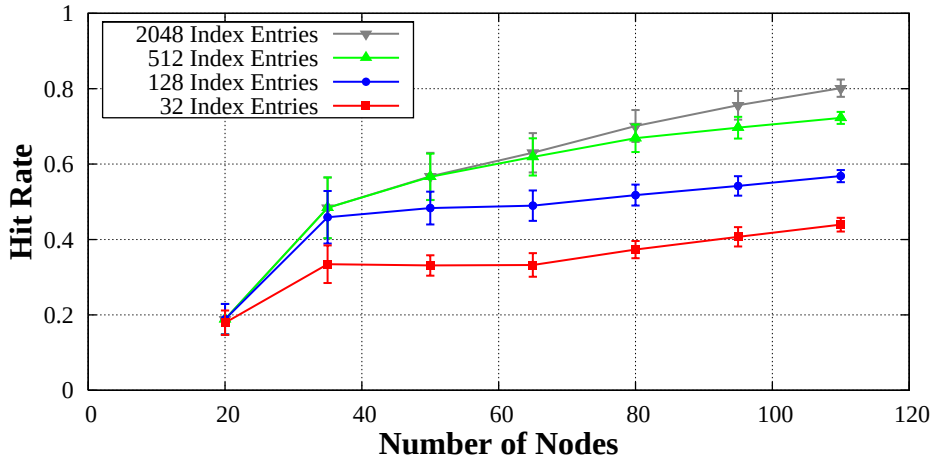

Figure 3.14. No invalidation: recall vs. system size

performance curves are shown in Figures 3.14 and 3.15. Figure 3.14 plots hit rates as a function of the number of nodes for different sizes of the index cache. Consistent with Figures 3.4 and 3.5 the results reveal that hit rate increases with a growing number of nodes because an increasing number of nodes increase the dissemination of information. Furthermore, the increase slows down with an increasing number of nodes because the total number of values in the overall system increases. In these cases, the hit rate is clearly limited by the overall index cache size. Note that increasing the index cache size from 512 to 2048 entries, i.e., by factor 4, only increases the hit rate about 10% for systems comprising many nodes.

The stale hit rates as a function of the number of nodes is plotted in Figure 3.15. We find that without invalidation the stale hit rate is at most 0.4. For smaller index cache sizes, the the stale hit rate decreases with the number of nodes. Jointly

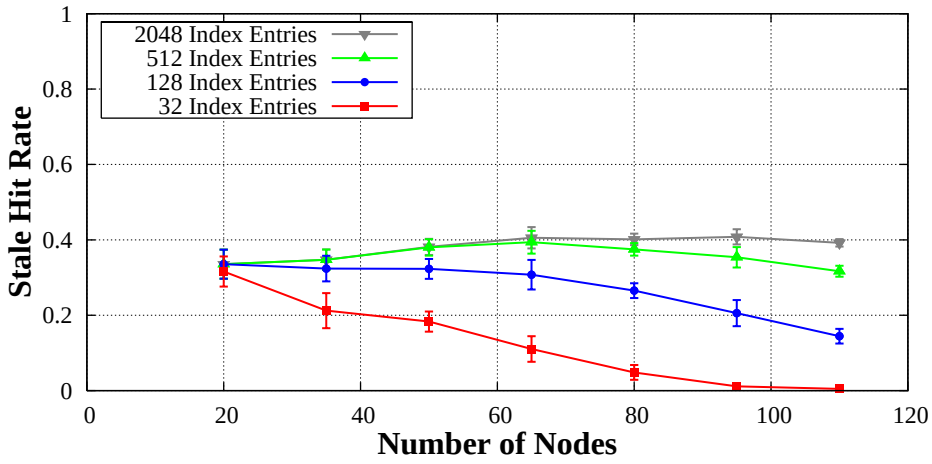

Figure 3.15. No invalidation: coherency vs. system size

considering Figures 3.14 and 3.15 reveals that for increasing number of nodes the stale hit rate drops rapidly at the point where the growth of the hit rate slows down. Looking closer at the index caches in these scenarios, we find that the cache content is highly variable. Therefore, stale index entries are removed from the caches early. We conclude from Figure 3.14 that large caches yield a high amount of stale hits when no invalidation mechanism is used. In contrast, small index caches naturally reduce stale hits, while they fail to provide high hit rates. This evidently illustrates the need for invalidation mechanisms in order to achieve both high hit rates and low stale hit rates.

For the following experiment, we fix the number of nodes to $N = 80$ nodes and investigate the performance of PDI with enabled value timeouts as implicit invalidation mechanism. Figure 3.16 plots the hit rates versus the timeout duration for different index cache sizes. As value timeouts invalidate both stale and up-to-date index entries, the hit rate increases with an increasing timeout duration, since invalidations occur less frequently. Unfortunately, as Figure 3.17 reveals, the stale hit rate also increases. However, comparing Figures 3.16 and 3.17 illustrates that the stale hit rate grows almost linear with an increasing timeout duration while the hit rate grows in a log-like fashion. Based on this observation, we choose a low timeout duration in order to limit the decrease in hit rate. For example, given an index cache with 2048 entries, a timeout of 1000 seconds decreases the hit rate by about 0.07, while the stale hit rate is decreased from 0.5 to 0.11. That is about 75% improvement compared to the corresponding scenario without invalidation as shown in Figures 3.14 and 3.15, respectively. Note that the optimal timeout duration clearly depends on the rate of information modification as well as on the arrival and departure rates

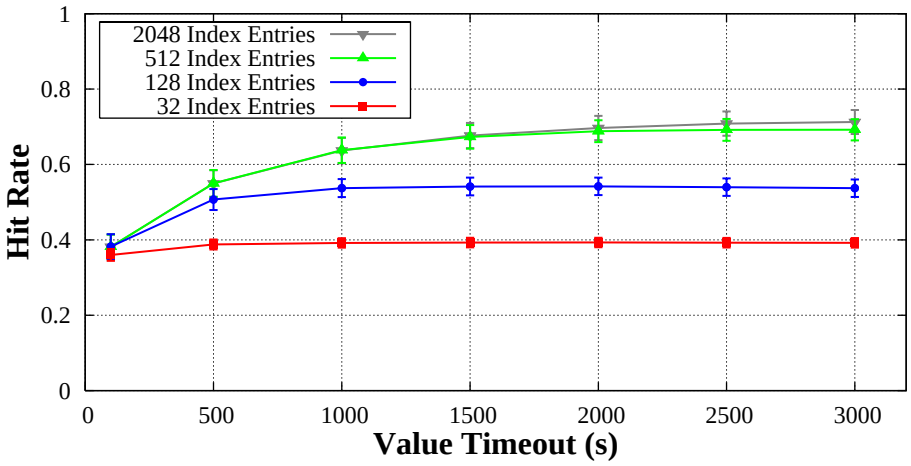

Figure 3.16. Value timeouts: recall vs. timeout duration

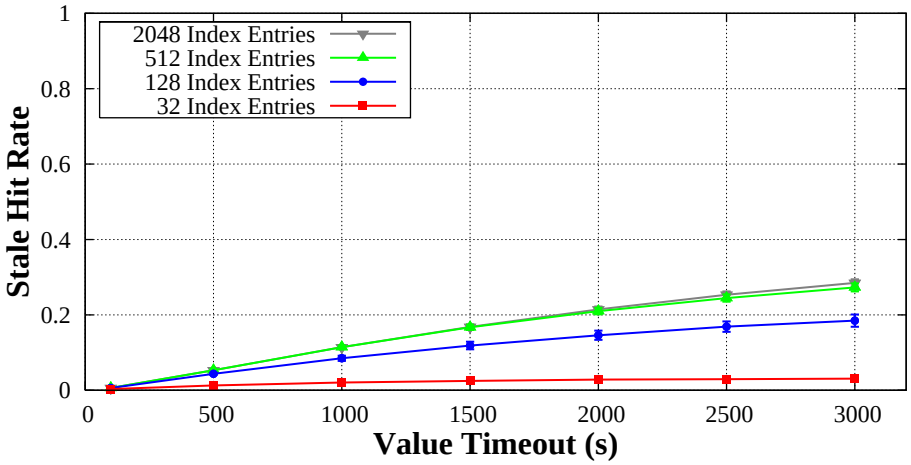

Figure 3.17. Value timeouts: coherence vs. timeout duration

for nodes. Thus, a timeout of 1000s may not be the best choice for all application scenarios. To gain further inside into the behavior of value timeouts, Figure 3.18 plots the coherence efficiency versus the timeout duration. We find that the coherence efficiency rapidly drops with increasing timeout duration due to more infrequently occurring invalidations. Surprisingly, we find that value timeouts are less efficient for small cache sizes than for large ones. Again, this is a confirmation that small caches naturally reduce hits for stale index entries by frequent replacements, cutting of the room for improvements by a timeout-based invalidation mechanism. We conclude from Figures 3.16 to 3.18 that value timeouts provide an efficient mechanism for implementing implicit invalidation, especially for large index caches.

In the following experiment, we investigate the performance of lazy invalidation caches as explicit invalidation mechanisms. As above, the number of nodes is kept

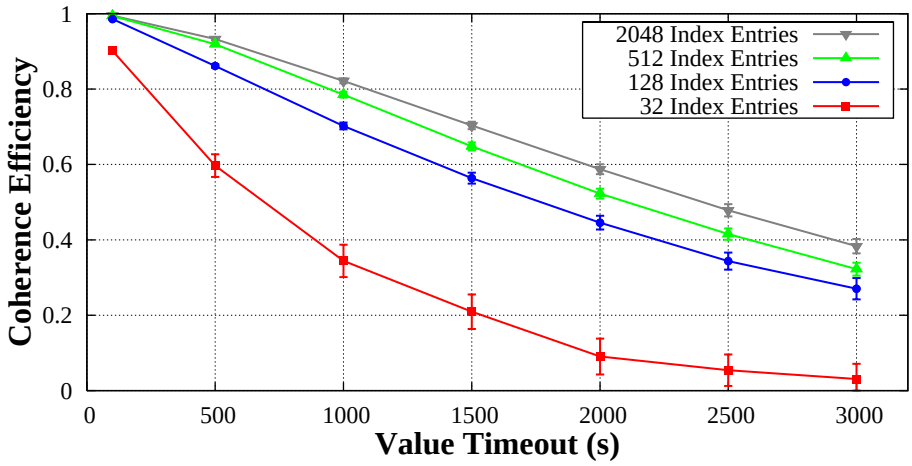

Figure 3.18. Value timeouts: coherence efficiency vs. timeout duration

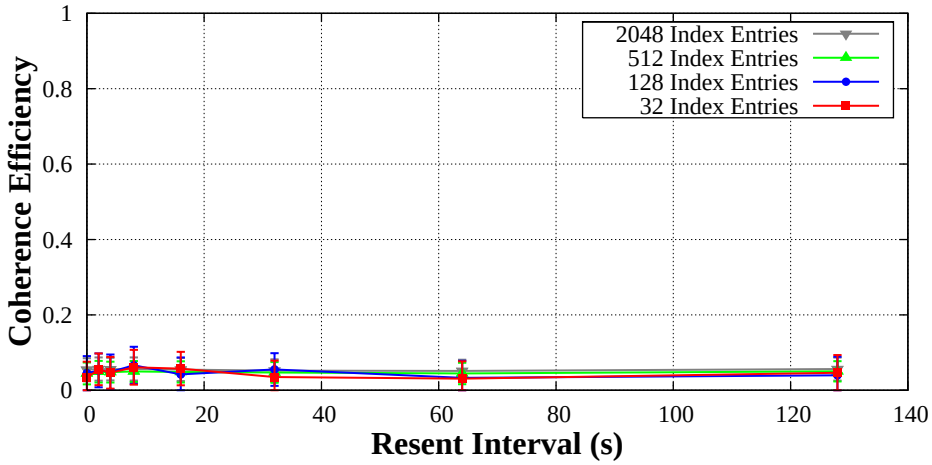

Figure 3.19. Flooded invalidation messages: Coherence efficency vs. resent interval

fixed to 80 nodes. Figure 3.19 plots the coherence efficiency achieved by pure periodical flooding of invalidation messages without using invalidation caches versus re-sent time, i.e., the interval between two successive transmissions of an invalidation message. Confirming the claim from Section 3.2.4, we find that independent of the re-sent time the coherence efficiency is below 0.1 regardless of the index cache size. Note that due to weak connectivity, flooding of invalidation messages cannot eliminate all stale index entries. Subsequently, the epidemic dissemination of remaining index entries leads to a redistribution of stale values. Thus, as illustrated by Figure 3.19, periodic flooding of invalidation messages fails to implement explicit invalidations.

Next, we investigate the performance of the epidemic dissemination of invalidation messages using invalidation caches. In various experiments not shown here, we

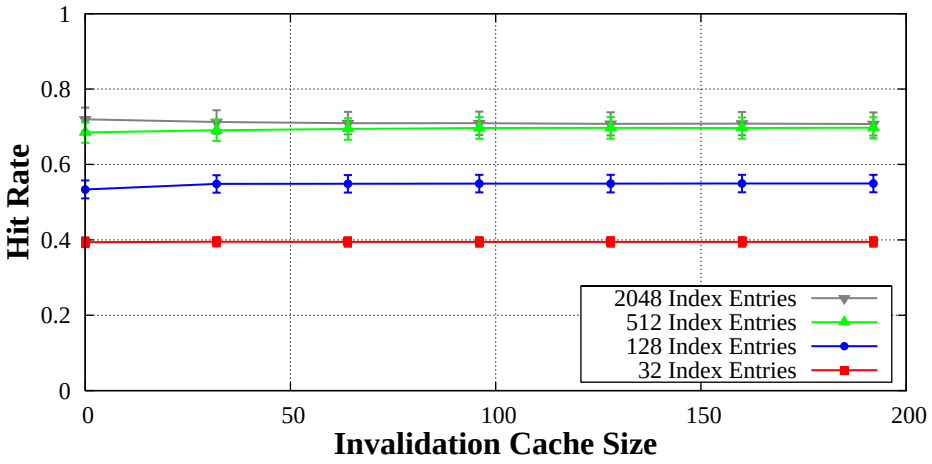

Figure 3.20. Invalidation caches: recall vs. index cache size

investigated the sensitivity of all performance measures to the TTL_{inv}. We found that performance does not significantly increase for $TTL_{inv} > 2$ hops. Thus, we set $TTL_{inv} = 2$ hops in all subsequent experiments. Figure 3.20 illustrates that lazy invalidation caches regardless of their size do not affect the hit rate, since opposed to value timeouts explicit invalidation messages only invalidate stale index entries.

However, lazy invalidation caches significantly reduce the stale hit rate, especially for large index cache sizes, as shown in Figure 3.21. We find that for large caches the stale hit rate is reduced by more than 50% compared to Figure 3.15. Note that if the invalidation cache size increases beyond 20% of the index cache size, no significant further reduction of the stale hit rate can be achieved. For practical applications, this means that the invalidation caches can be small compared to the index caches. This observation is confirmed by the results for the coherence efficiency shown in Figure 3.22. Opposed to the coherence efficiency of value timeouts, the coherence efficiency of lazy invalidation caches is best for small cache sizes. Comparing Figure 3.20 to Figure 3.18, we find that the coherence efficiency of lazy invalidation caches is smaller than for value timeouts when using large index caches. To understand this observation, recall that nodes leaving the system do not explicitly invalidate all index entries they have supplied, a worst-case scenario for each explicit invalidation mechanism. We conclude from Figures 3.20 to 3.22 that lazy invalidation caches efficiently implement explicit invalidation.

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 intermittent connectivity and modification of information. We fix the duration of the value timeout to $1000s$ and the invalidation cache size to 128 entries. Figure

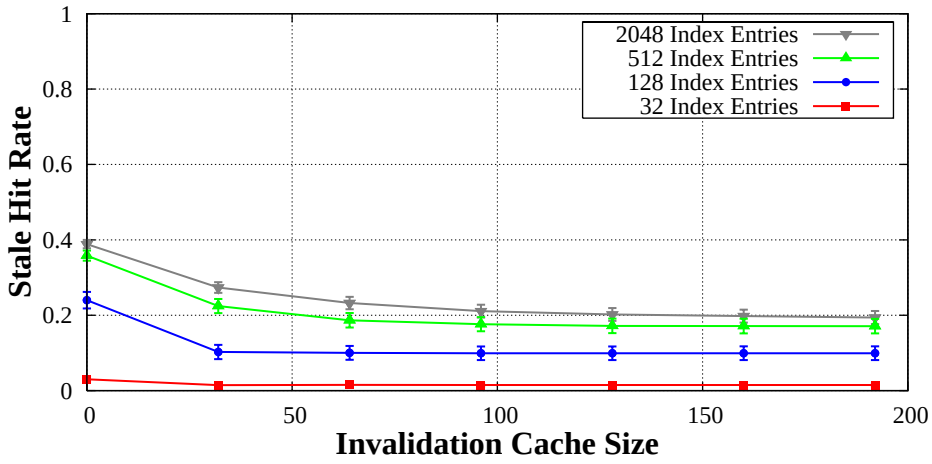

Figure 3.21. Invalidation caches: coherence vs. index cache size

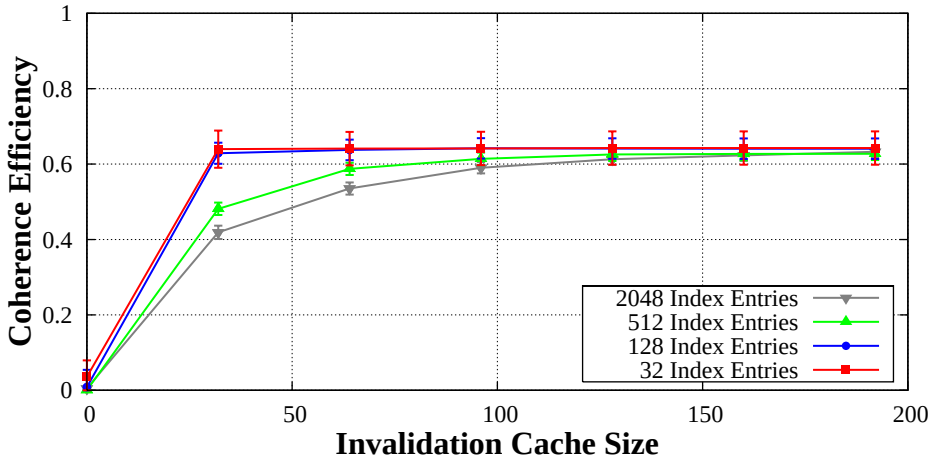

Figure 3.22. Invalidation caches: coherence efficiency vs. index cache size

3.23 plots the hit rate versus the number of nodes. We find that hit rate is reduced mostly for systems with few nodes due to invalidations for 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 a growing number of nodes, the hit rate converges towards the results without an invalidation mechanism as already shown in Figure 3.14.

As compensation of the reduction of hit rate, the stale hit rate is significantly reduced compared to a system without invalidation. As shown in Figures 3.24 and 3.25, the stale hit rate is highest and the coherence efficiency is worst for a medium number of nodes and sufficient large index cache sizes. The reason is that a fixed cache size of 128 entries is somewhat too large for systems with few nodes, while

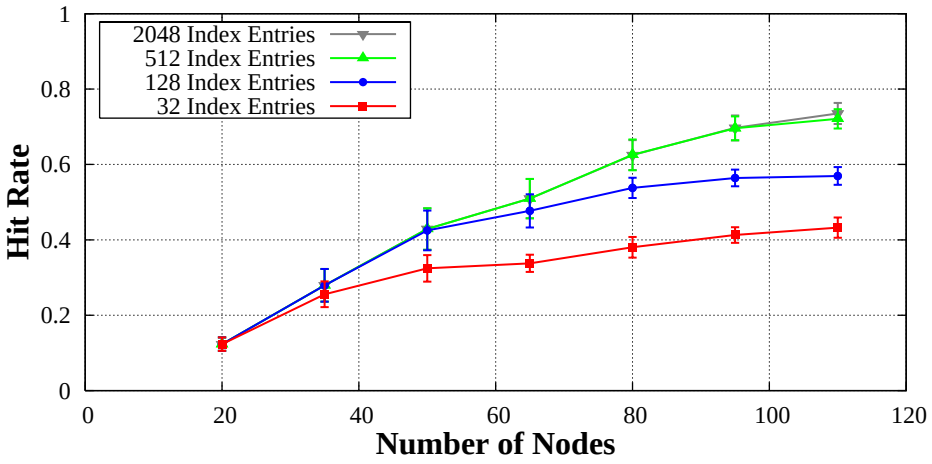

Figure 3.23. Integrated approach: recall vs. systems size

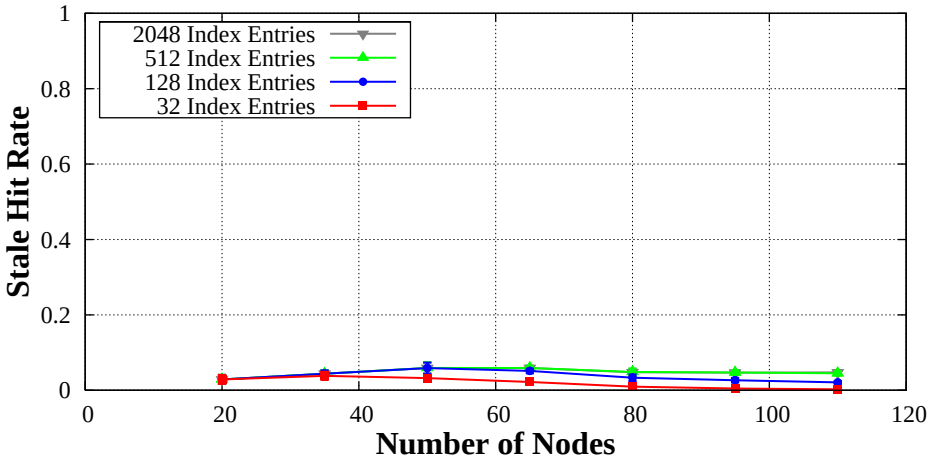

Figure 3.24. Integrated approach: coherence vs. systems size

for large systems the natural limit of stale index entries increases the coherence efficiency. Again, the coherence efficiency drops rapidly for small index cache sizes due to a natural reduction of stale hits. We conclude from Figures 3.23 to 3.25 that the integrated approach comprising 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.

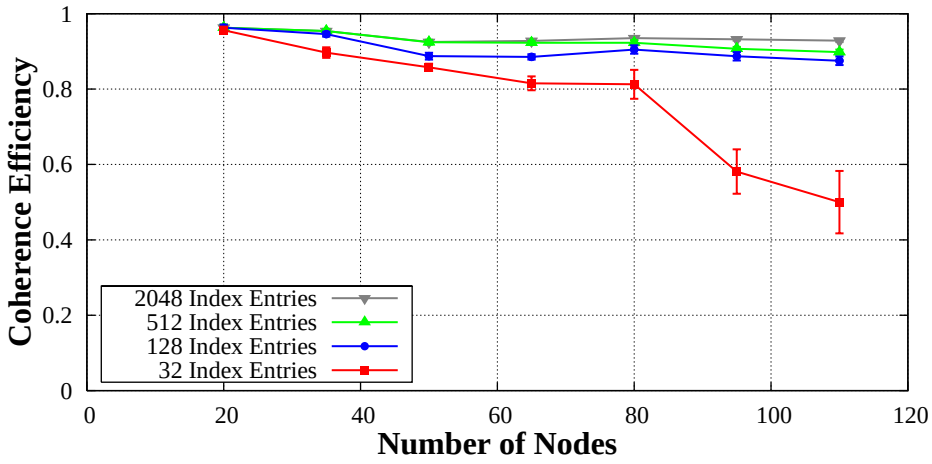

Figure 3.25. Integrated approach: coherence efficiency vs. systems size

3.4.5 Performance of other Applications

In this section, we present performance results for PDI together with the IM, IP, and DWS applications. We have performed a comprehensive performance study for each of these applications. Some of the results are very similar to the results obtained for the P2P file sharing application in Sections 3.4.3 and 3.4.4. However, other results show significant differences or provide further key insight into the operation of PDI. We restrict the performance curves in this section to the latter results.

Performance of the Instant Messaging Application

In a first set of experiments, we analyze the recall and coherence for the IM application. Recall that opposed to the P2P application the IM application uses significant fewer values, i.e., each node contributes only one fresh value at a time. However, values expire much more frequently due to changes of the present state of a mobile user. To take into account these different application characteristics, we use significantly smaller index cache sizes with a capacity ranging from 16 to 128 entries. Furthermore, we use smaller value timeouts of $600s$ compared to $1000s$ for the P2P application.

Figure 3.26 shows recall and consistency for PDI and the IM application without using a consistency mechanism. Comparing Figures 3.26 and 3.14, we find that the hit rate for systems with few nodes is almost identical for both applications. However, for systems with many nodes, the difference between the performance of the different index cache sizes is much smaller for the IM application than for the P2P application due to the small number of values in the system. Looking at the result coherence in

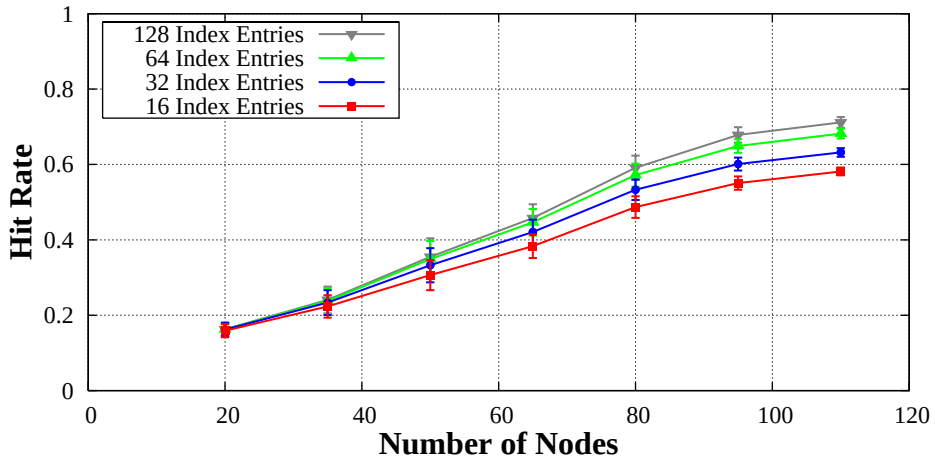

Figure 3.26. Recall vs. system size for the instant messaging application without invalidation mechanism

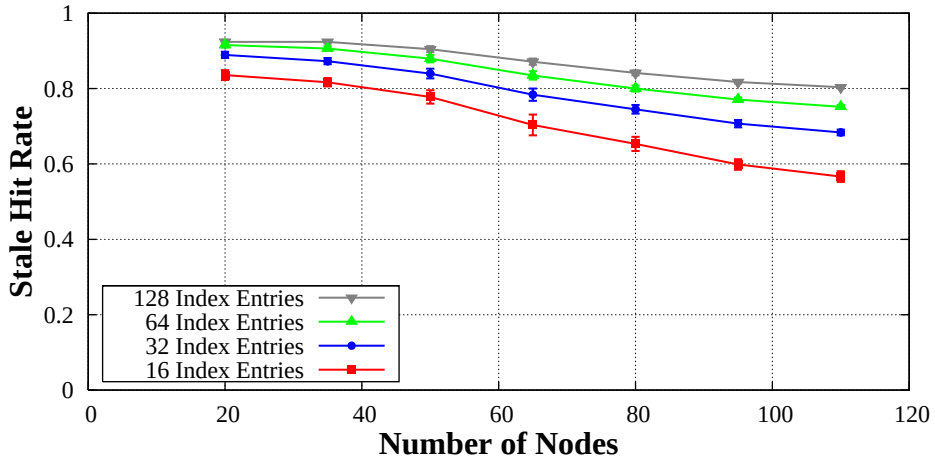

Figure 3.27. Coherence vs. system size for the instant messaging application without invalidation mechanism

Figure 3.27, we also find that the stale hit rate is much higher for the IM application. Even for small caches, cache replacement does not occur frequently enough to reduce stale hit rate below 0.55.

When enabling value timeouts and lazy invalidation caches, Figure 3.28 shows that hit rate does not drop due to value timeouts as observed for the P2P application in Figure 3.23, but does increase for all cache sizes. This shows that index cache space is wasted by stale index entries, which are removed by the consistency mechanisms. Looking at Figure 3.29, we find that the stale hit rate is very similar for all index cache sizes. Here, value timeouts reduce the stale hit rate from more than 0.9 to below 0.3, i.e., by more than 66%. With a growing number of nodes, epidemic dissemination

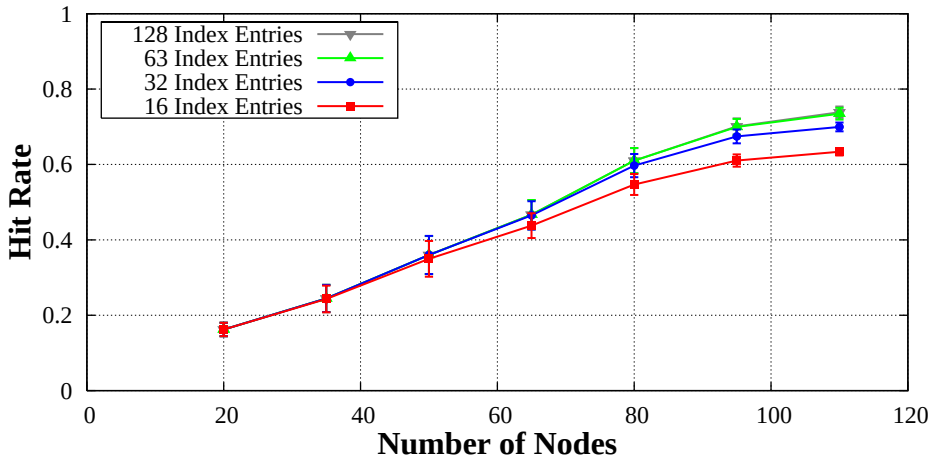

Figure 3.28. Recall vs. system size for the instant messaging application with hybrid invalidation

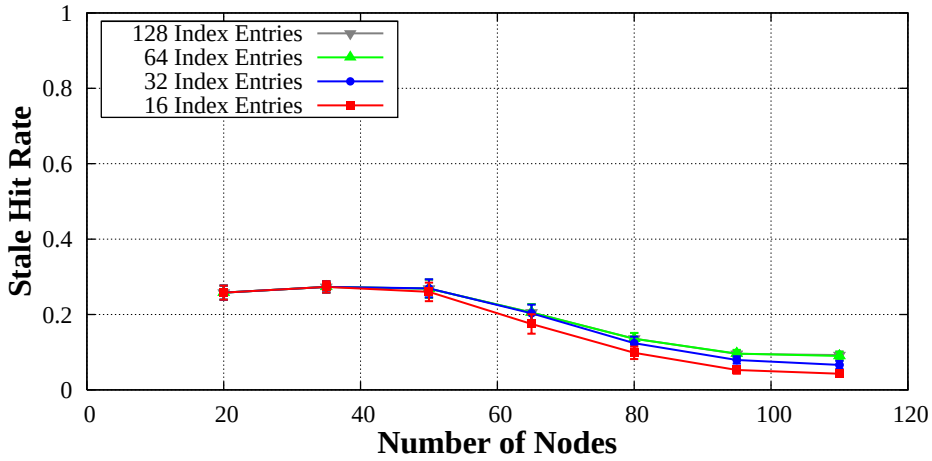

Figure 3.29. Coherence vs. system size for the instant messaging application with hybrid invalidation

of invalidation messages is fostered. This reduces the stale hit rate below 0.1 for all cache sizes, that is a reduction of more than 87%.

A hit rate of below 0.5 and a stale hit rate up to 0.25 seem to be unacceptable for an instant messaging application. However, note that by the addition of timestamps to the presence state that is represented by a value, the application can easily resolve conflicts between stale and fresh present states by selecting the most recent presence state of a contact from the results returned by PDI. Furthermore, the application periodically polls the presence state and can display the most recent presence state starting from the query for which this particular state has been returned for the first time. Thus, the coherence of the presence state displayed by a properly designed IM

application is much higher than predicted by the stale hit rate of PDI. In Chapter 4 we present a system that is tailored for dissemination of presence information and introduce a performance measure that predicts result consistency more accurate than stale hit rate.

Performance of the Mobile Information Portal Application

In a second set of experiments, we analyze the performance of PDI for the IP application. Recall that we do not consider value expiration for this applications. However, we use this application to illustrate two aspects of PDI: First, we illustrate the behavior of PDI for very large system sizes. Second, we illustrate the impact of the mobility model on the performance of PDI.

Figure 3.30 plots the hit rate vs. the number of nodes for the RWP, Manhattan Grid, and RPG mobility model, respectively, with an index cache size of 2048 entries. Looking at the curve for the RWP mobility model, we find that the hit rate grows only up to about 180 nodes. For larger systems, hit rate stagnates at 0.7 for large caches. This stagnation has two reasons. First, the system reaches almost full connectivity at this system size. That is, connectivity does not increase by adding further nodes. Nevertheless, the reachability of the fixed infostations is limited due to a TTL query of 4 hops and in many cases a node does not reach one of the four infostations directly. Second, although there are typically many nodes in the proximity of the inquiring node, the variety of the values stored in these caches does not increase. Again, this is because broadcasting and selective forwarding of Response messages causes the insertion of all index entries contained in the message into the caches of all nodes in proximity of the inquiring node. Thus, fewer different values are available from these caches.

Comparing the curves for the different mobility models shown in 3.30, we find that the RWP model exhibits the best performance. This is due to the fact that the spatial node distribution is not uniform with the RWP mobility model but shows a high concentration of nodes in the center of the simulation area [BRS03], leading to an increased connectivity in this area. Performance for the Manhattan Grid mobility model is about 0.04 smaller for large index caches, since with this mobility models the nodes are uniformly distributed across the streets in the simulation area. The RPGM model exhibits worst behavior for both cache sizes since groups of nodes tend to have very similar index entries in the index caches. Thus, the RPGM mobility model catches up with the other mobility models for very large system sizes, since encounters of groups occur frequently enough to disseminate information among groups. We conclude from Figure 3.30 that node concentrations of all nodes are

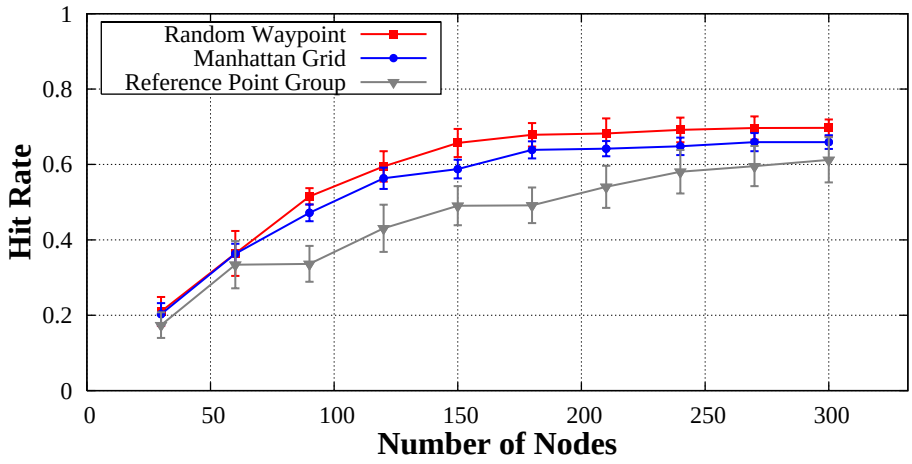

Figure 3.30. Recall vs. system size for the IP application and different mobility models

advantageous for PDI, whereas the grouping of nodes hinders epidemic information dissemination.

Performance of the Distributed Web Search Application

In a last set of experiments we consider the DWS. Recall that the characteristics of this application are very similar to the P2P application with the exception that much more documents are matched by a key, requiring the usage of the response threshold feature. Furthermore, similar to the IP application values are contributed to the DWS application by fixed infostations. In contrast to the IP application, all fixed infostations in the DWS application contribute an identical data set.

We use the DWS application to illustrate the impact of the fixed infostation on the performance of PDI. Figure 3.31 shows the hit rate and the infrastructure hit rate as a function of the number of infostations. We find in Figure 3.31 that the hit rate increases very fast up to a number of eight infostations. Further increasing the number of the infostations has only a limited impact on the performance of PDI, due to the fact that with random placement of infostations the radio coverage of many infostations may overlap. For a large number of infostations, all index cache sizes perform almost equal in terms of hit rate. However, Figure 3.32 shows that with small index caches many hits are received directly from the infostation, whereas with large index caches many hits are received from the index caches, although connectivity to an infostation is available. Similar to the observations made for the analysis of PDI with different node speed in Figure 3.9, selective forwarding significantly reduces traffic in these scenarios. We conclude from Figures 3.31 and

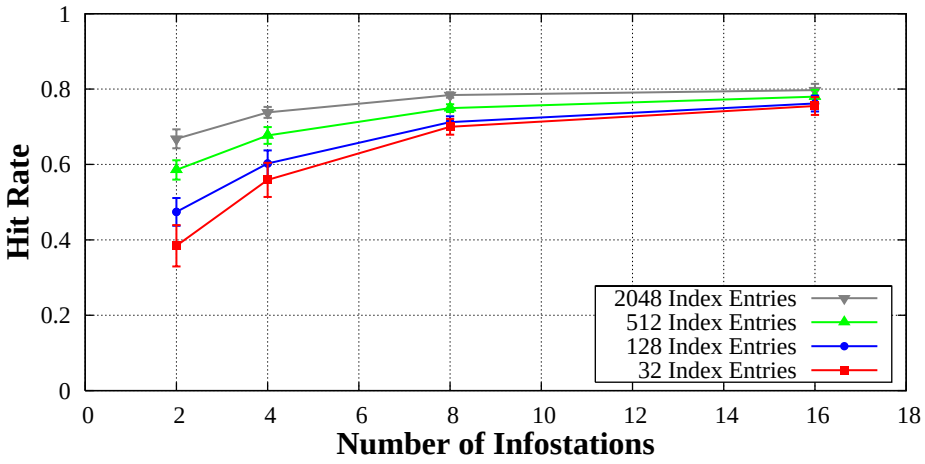

Figure 3.31. Recall vs. system size for the DWS application

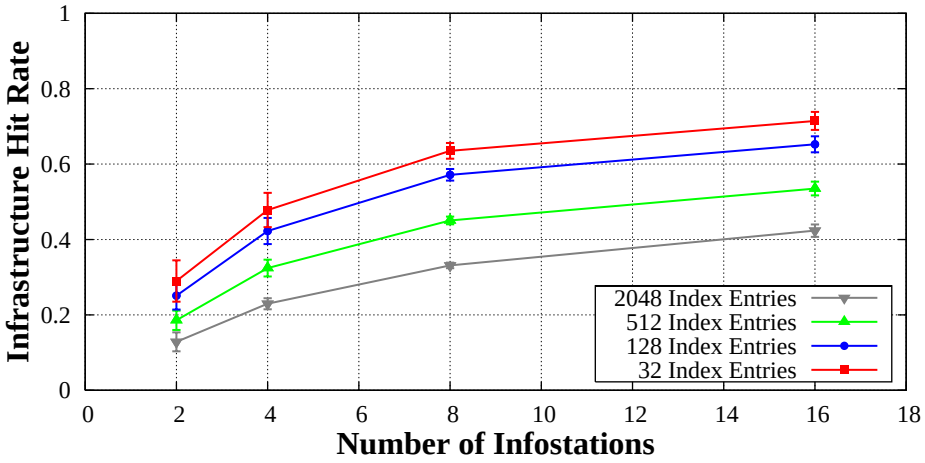

Figure 3.32. Infrastructure hits vs. system size for the DWS application

3.32 that a very high number of randomly placed infostations does not significantly increase PDI performance. Nevertheless, large index caches are advantageous even with many infostations to reduce data traffic by selective forwarding.

3.5 Implementation Aspects

The functionality of PDI and the interface between PDI and the application as well as all message types used by PDI are defined by a draft PDI specification [LW04b]. Since PDI is designed as a general-purpose lookup service, it provides a generic application-programming interface (API). To supply index entries (k, v) to the PDI system, an application may use two different methods: When using a limited set of predefined index entries the application may call an `add` method for each index

entry. When using a large or even dynamic set of index entries the application can provide a lookup function that is called by PDI on reception of a QUERY message. In the first case, index entries are inserted into a local index provided by PDI, whereas in the second case the local index is replaced by the functionality provided by the application. To support sophisticated lookup operations, each index entry (k, v) can be associated with a score, which represents the relevance of v to the key k. To resolve keys to values, the application calls a `resolve` method. For the convenience of the application developer, PDI incorporates a resolve method that accepts a list of keys and a Boolean operation, i.e., "and" or "or". PDI resolves this query to all values matching the keys with respect to the Boolean operation. Since the transmission of query and response messages may consume considerable time, queries are processed asynchronously. That is, the application provides a notification function that is called each time a response to a query is received. To provide authentication and message integrity, the application has to provide some additional functionality, which is discussed at the end of this section.

Assuming that sufficient network bandwidth is available and that processing power is not the bottleneck, the current PDI specification [LW04b] defines all message formats using the *Extensible Markup Language XML*. These assumptions hold for mobile devices such as notebooks and high-end PDAs that are equipped with IEEE 802.11x WLAN interfaces. For resource-limited devices such as mobile phones, it might be advantageous to implement compact and regularly structured binary messages in order to save network bandwidth and computational resources. However, binary messages have to incorporate all elements contained in the XML messages defined by [LW04b].

An XML Schema [W3C03] that provides the XML namespace `pdi` defines all PDI message formats. The XML Schema definition can also be found in the draft specification [LW04b]. All PDI messages use `<pdi:message/>` as root element. The attributes of the root element specify all information that is required to identify and forward a PDI message. That is, a message is uniquely identified by the attributes `srcId` and `srcSeq`. In the examples shown in Figures 3.33 to 3.35, all node identifiers are set to statically assigned IP addresses. However, any other unique string could be used as identifier, e.g., the MAC address of the wireless communication interface or a unique ID assigned to each instance of a distributed application by the application vendor. Forwarding is controlled by the attributes `ttl` and `hopCount`. The type of a PDI message is specified by the second level elements `<query/>`, `<response/>` or `<invalidation/>`, respectively. As shown in the QUERY message in Figure 3.33, a `<query/>` element consists of a single element `<queryDescr/>`. The

attribute `boolOp` specifies the Boolean operation that combines the keys. To match responses to queries, the inquiring node is specified by the attributes `inqId` and `inqSeq`. The keys in a query are specified using `<key/>` elements. These elements may contain an optional attribute `id` that is used to refer to a key when specifying scores in response messages.

A `<response/>` element consists of a single element `<queryDescr/>` and a single element of type `<responseDescr/>`, as shown in Figure 3.34. To enable a node to extract information from overheard response messages, the content of the `<queryDescr/>` element is copied from the Query message that has triggered this particular Response message. The `<responseDescr/>` element contains `<responseEntry/>` elements, each describing a value matching the `<queryDescr/>`. To support explicit invalidations using lazy invalidation caches, the attribute `responderId` specifies a unique identifier of the node that supplied this value. To enable implicit invalidation by value timeouts, the attributes `age` and `maxAge` specify the time elapsed since response generation by the origin node and the timeout for this particular value, respectively. The actual value is given by the `<value/>` element. Note that efficient operation of PDI requires efficient comparison of values. If an application uses large values, the PDI specification allows encapsulation of these values in a `<meta/>` element. This element may contain data of any type, including further XML elements. When using the `<meta/>` element, the `<value/>` element must be set to a unique identifier of the data in the `<meta/>` element, e.g., given by an MD5 hash sum. For specifying the score of a value with respect to each key, an `<responseEntry/>` may contain `<score/>` elements. Each `<score/>` element refers to a key in the query

```
<?xml version="1.0" encoding="UTF-8"?>
<pdi:message
    xmlns:pdi="http://mobicom.cs.uni-dortmund.de/pdi/pdi.xsd"
    srcId="129.217.16.109" srcSeq="12879"
    ttl="2" hopCount="2">
    <query>
        <queryDescr boolOp="and"
            enqId="129.217.16.109" enqSeq="12879">
            <key>Madonna</key>
            <key>Girl</key>
        </queryDescr>
    </query>
</pdi:message>
```

Figure 3.33. A PDI Query message

descriptor using a reference attribute `ref`.

An `<invalidation/>` element consists of a single element of type `<invalDescr/>`, as shown in Figure 3.35. The Invalidation message may invalidate one or more values, each specified in an `<invalEntry/>` element inside the `<invalDescr/>`. Each `<invalEntry/>` incorporates the unique identifier of the invalidating node using the `invalId` attribute. Similar to `<responseEntry/>` elements, supply time of an `<invalEntry/>` element can be calculated using the attribute age. The `<value/>` element contained in an `<invalEntry/>` element describes the value to invalidate. The content of this element is specified as for the `<responseEntry/>` element.

The description of PDI in Section 3.2 does not consider security issues, e.g., faked index entries or denial-of-service attacks. To provide a basic authentication

```
<?xml version="1.0" encoding="UTF-8"?>
<pdi:message
    xmlns:pdi="http://mobicom.cs.uni-dortmund.de/pdi/pdi.xsd"
    srcId="129.217.16.96" srcSeq="4365"
    ttl="2" hopCount="2">
    <response>
        <queryDescr boolOp="and"
            enqId="129.217.16.109" enqSeq="12879">
            <key id="k01">Madonna</key>
            <key id="k02">Girl</key>
        </queryDescr>
        <responseDescr>
            <responseEntry id="r01"
                responderId="129.217.16.96"
                age="PT41S" maxAge="PT10M">
                <value>
                    //129.217.16.96/share/madonna_material_girl.mp3
                </value>
                <meta>
                    <artist>Madonna</artist>
                    <tile>Material Girl</tile>
                </meta>
                <score ref="k01">3</score>
                <score ref="k02">3</score>
            </responseEntry>
        </responseDescr>
    </response>
</pdi:message>
```

Figure 3.34. A PDI Response message

mechanism for mobile nodes running PDI and to ensure integrity of the content of PDI index caches, we propose the incorporation of digital signatures in PDI. Such signatures are inspired by XML signatures [ERS02]. We define a proprietary mechanism for signing message components rather than adopting XML signatures. The proposed mechanism uses shorter messages and, thus, requires fewer computational resources and fewer bandwidth than the mechanisms specified in [ERS02]. For message authentication, all message types can be extended by a `<securitDesrc/>` element that may contain signatures for `<queryDescr/>`, `<responseEntry/>`, and `<invalEntry/>` elements. PDI does not verify digital signatures itself, but passes the signature together with the signed element to the application. The application verifies the signature by an arbitrary method and returns "*true*", if the signature is valid, and "$false$" otherwise. Based on these results, PDI discards invalid elements. Furthermore, PDI may limit the ratio for processing information from a particular node. Note that if required by the application, security certificates can be distributed in the mobile network in a self-organizing fashion using PDI.

During the creation of this thesis, several software-prototypes of PDI have been implemented. A complete simulation environment for performance studies of PDI has been implemented in [BÖ3]. [Are03] has implemented a complete P2P file sharing system based on an early version of PDI that uses binary messages. [Bih04] and [Nag04] implemented a XML based PDI version with messages according to the

```
<?xml version="1.0" encoding="UTF-8"?>
<pdi:message
    xmlns:pdi="http://mobicom.cs.uni-dortmund.de/pdi/pdi.xsd"
    srcId="129.217.16.43" srcSeq="312784"
    ttl="2" hopCount="2">
    <invalidation>
        <invalDescr>
            <invalEntry
                invalId="129.217.16.96"
                age="PT12M31S" maxAge="PT30M">
                <value>
                    //192.168.0.21/MySongs/madonna_who_s_that_girl.mp3
                </value>
            </invalEntry>
        </invalDescr>
    </invalidation>
</pdi:message>
```

Figure 3.35. A PDI Invalidation message

XML schema definition in [LW04b] and sophisticated security mechanisms. This protocol version was used to implement a mobile instant messaging system [Bih04] and a distributed Web search application [Nag04].

3.6 Summary

In this chapter, we introduced Passive Distributed Indexing (PDI), a distributed lookup service for mobile applications. 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.

For the evaluation of PDI in simulation experiments, we define a framework for characterizing the workload generated by a mobile application by five workload functions. We presented customizations of these workload functions for four mobile applications, i.e., mobile P2P file sharing, mobile instant messaging, a mobile information portal, and disconnected Web search. The workload characterizations for the different applications were used as input of a comprehensive simulation study. Within the study, we analyzed the recall and the result coherence achieved by PDI for the P2P file sharing application. 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.

Extending the results for the P2P file sharing application we found that for the instant messaging application the PDI invalidation mechanisms can easily cope with the high rate of value expirations and even increase hit rate, since with invalidation no index cache space is wasted for stale index entries. For the mobile information portal, we demonstrated that increasing the number of nodes up to 300 does not significantly increase hit rate, since the content of nearby caches is identical in many cases. For the disconnected Web search application, we showed that a moderate number of infostations provides fair hit rate. Furthermore, large index cache sizes do not increase the hit rate, but save a large amount of traffic due to the selective forwarding mechanism.

Motivated by the results of the simulation study, a draft specification of PDI has been created. The specification uses XML based messages and features a basic security mechanism. Prototypes that conform to the specification have been successfully implemented for P2P file sharing, mobile instant messaging, and disconnected web search.

Chapter 4

Epidemic Dissemination of Presence Information

After the introduction of presence technology as an integral part of all state-of-the-art instant messaging (IM) systems, users quickly noticed that it is beneficial to know whether a remote person is currently online and ready to communicate in many other applications. Examples include sophisticated office application that enable computer supported collaborative work, e.g., the the 2003 release of Microsoft's Office Suite. Presence technology will definitely play a decisive role in mobile applications. In Chapter 3 it has been shown that PDI can be employed for epidemic dissemination of presence information. However, PDI constitutes a general-purpose lookup service and neither the mechanisms proposed nor the performance measures applied in Chapter 3 are tailored to the requirements of the dissemination of presence information. Considering these requirements leads to more efficient solutions. In this chapter, we present sustained consistency of presence information as a novel performance measure and explores the efficiency of flooding and epidemic algorithms for presence information dissemination in a highly partitioned MANET. Based on the key insight on the efficiency of such approaches, the *System for Presence information Exchange by Epidemic Dissemination* (*SPEED*) is proposed and evaluated against an optimized approach based on periodical flooding of presence information. The results presented in this chapter have been published in *Technical Report 799* by the CS department of the University of Dortmund [LLW05].

4.1 A Coherence Measure for Frequently Changing Information

Recall that the building blocks of an IM system comprise authentication, message exchange, and managing of presence information. In this Chapter, we focus on communication paradigms for disseminating presence information, since a central presence server cannot be employed for an IM system running in a highly partitioned MANET. In such systems, presence information must be communicated in a peer-to-peer fashion from a mobile node to all its contacts either in a single hop or in multiple hops via other mobile nodes. Note that state-of-the-art IM systems proactively provide a coherent view of the current presence states of a user's contacts. Thus, if a contact changes its presence state, all associated users are notified by the IM system immediately.

In a recent paper, Westphal showed how to disseminate information in an ad hoc network in order to achieve information persistence in presence of node failures [Wes05]. [Wes05] assumes unlimited validity of information and limited availability of nodes. Opposed to [Wes05], this Chapter consider presence information with limited validity together with always available nodes in a partially connected network. For evaluating the effectiveness of different communication paradigms for proactively disseminating presence information in such environment, we introduce *sustained consistency* as novel performance measure.

Consider N mobile nodes which have subscribed to an IM system. We denote mobile nodes by the variable n, m, and k, with $1 \leq n, m, k \leq N$. We derive a discrete-event stochastic system as a model for describing the dynamics of an IM system. The model comprises two stochastic processes: the *mobility model* and the *system model*. The location of each mobile node within a geometric area $\mathcal{A} = a_1 \times a_2$ is determined by the mobility model. The mobility model is basically a continuous-time, continuous-state stochastic process $(X_n(t), Y_n(t))$, $0 \leq X_n(t) \leq a_1$, $0 \leq Y_n(t) \leq a_2$. That is, at an instance of time t the mobility model associates a location $(X_n(t), Y_n(t))$ with each mobile node n. The mobility model interacts with the system model by determining whether two nodes can communicate at a given instance of time t. To derive connectivity information from the node positions at an instance of time t, we assume that all nodes have a homogenous transmission range R. Nodes n and m can communicate directly (i.e., in one hop), if and only if holds $d \leq R$ for

$$d = \sqrt{(X_n(t) - X_m(t))^2 + (Y_n(t) - Y_m(t))^2}. \tag{4.1}$$

Note that this constitutes a best-case assumption, since a transmission between two nodes may fail due to various reasons, including the impact of the medium access protocol, interference, and wireless errors. However, we use this idealized assumptions for computing performance bounds for various communication approaches. For evaluating our system proposal in Section 4.4, we will revert to a more realistic transmission model by using the network simulator ns-2.

Besides direct communication, an approach for disseminating presence information in MANET may employ multihop communication. That is, communication between two nodes n, m is performed over a distance of l hops, where $l-1$ mobile nodes act as relay. For an instance of time t, we define a distance matrix $D(t)$ that is a random matrix with dimensions $N \times N$ with $D_{n,m}(t) \in 0, 1, ..., N-1, \infty$. Here, $D_{n,m}(t) = l$ constitutes the length of the shortest sequence of nodes connecting nodes n and m, $n \neq m$. We set $D_{n,m}(t) := \infty$ if no such sequence between n and m exists and $D_{n,n}(t) := 0$. Note that $D(t)$ describes a continuous-time discrete-state stochastic process that is controlled by the mobility model.

For the system model, we assume that at an instance of time t a node n is in a presence state $S_n(t)$ taking values in a set $\mathcal{S}$, e.g., $\mathcal{S} = \{online, busy, do-not-disturb, away\}$. Besides its own presence state, a mobile node n stores information about the presence states $R_{n,k}(t)$ of all other nodes k, $k \neq n$. Presence states $R_{n,k}(t)$ take values in $\mathcal{S} \cup \{undefined\}$, where $undefined$ denotes that the presence state of device k is unknown at device n. One can think of $R_{n,k}(t)$ as a cache for storing remote information that can be used by the communication approach. Note that at an instance of time t it may hold $R_{n,k}(t) \neq S_k(t)$. Putting it together, at any instant of time t, a mobile node n can be completely described by a set of random variables:

$$(S_n(t), R_{n,1}(t), R_{n,2}(t), \ldots, R_{n,n-1}(t), R_{n,n+1}(t), \ldots, R_{n,N}(t)) \tag{4.2}$$

State changes in the system model are triggered by one of the following three types of events:

Presence state change: A mobile node changes its presence state after some dwell time has elapsed. A state change event that occurs for node k at an instance of time t changes the state of the system by changing $S_k(t)$. Additionally, it may trigger one or more communication events that are described below.

Elapsed Timer: The presence communication approach of an IM system may perform periodic communication tasks triggered by timeouts. Depending on the communication tasks, several timeouts with different parameters may be active concurrently. In general, the occurrence of a timeout does not change the state of the system directly, but triggers one or more communication events.

Communication: Each communication operation used by the IM system is decomposed into one or more communication events, regardless if it is one-to-one or one-to-many communication. A communication event constitutes the information exchange between two nodes. A communication event that occurs between nodes n and k at time t changes the state of the system by changing either $R_{n,k}(t)$ or $R_{k,n}(t)$. Whether a communication event between nodes n and k occurs is determined by both the presence communication approach of the IM system and the distance matrix $D_{n,k}(t)$.

Presence state change events constitute stochastic events. For all mobile nodes, the dwell times in a presence state are drawn according to the same probability distribution with finite mean and variance. Elapsed Timer events occur exactly after a certain period of time has elapsed. Thus, the time until the occurrence of a timeout event is chosen according to a deterministic distribution. Since communication takes place on the timescale of milliseconds while presence state changes take place on the timescale of minutes, we consider communication events as immediate events occurring in zero-time.

We denote the instances of time at which presence state changes of $S_k(t)$ take place by $T_k(1), T_k(2), \ldots$. That is, $T_k(i)$ constitutes the instant of time of the i-th state change of the presence state of mobile node k. A presence state change of node k occurring at time $T_k(i)$ may be perceived at a node n due to communication events. Subsequently, $R_{n,k}(t)$ makes the i-th state change of the presence state. We denote the instance of time at which the i-th presence state change of node k is perceived at node n as $\hat{T}_{n,k}(i)$. With an optimal communication approach, we have $\hat{T}_{n,k}(i) = T_k(i)$. That is, node n has a coherent view of the presence state of node k. However, in a highly partitioned MANET typically holds $\hat{T}_{n,k}(i) > T_k(i)$ or even $\hat{T}_{n,k}(i) > T_{n,k}(i{+}1)$ if the i-th presence state change of k is never perceived by the node n. For convenience in the notation of Equations 4.3 and 4.6, we set $\hat{T}_{n,k}(i) := T_k(i+1)$ in this case.

In an IM system, the current state of node n is displayed only to a subset of all other nodes denoted as the contacts of node n. Hence, each mobile node n has assigned a number of contacts c_n drawn from some probability distribution. Given c_n, the set of contacts $\mathcal{C}_n \subseteq \{1, 2, \ldots, n-1, n+1, \ldots, N\}$ is determined according to some probability distribution with $|\mathcal{C}_n| = c_n$. For ease of exposition, we assume that for each node the set of contacts does not change over time. Obviously, the goal of a communication approach for disseminating presence information in MANET constitutes the maintenance of a coherent view of the presence state of a node n at all its contacts $k \in C_n$. To evaluate the effectiveness of a particular communication approach, we define the sustained consistency for a node n with respect to contact k,

denoted by $C(n,k)$, as the fraction of time that its presence information is coherent at contact k. We determine this fraction of time by summing up the time gap between perceiving the i-th presence state change at contact k and the next presence state change of node n normalized to the dwell time of node n in the present state after the i-th state change:

$$C(n,k) = \sum_{i=1}^{\infty} \frac{T_n(i+1) - \hat{T}_{k,n}(i)}{T_n(i+1) - T_n(i)} \tag{4.3}$$

The sustained consistency of node n, denoted by $C(n)$, is determined as the sustained consistency with respect to an individual contact k given in Equation 4.3 averaged over all contacts. Thus, we have:

$$C(n) = \frac{\sum_{k \in \mathcal{C}_n} C(n,k)}{|\mathcal{C}_n|} \tag{4.4}$$

Note that sustained consistency of node n is only determined by the remote state information $R_{k,n}(i)$ and stored at contacts $k \in \mathcal{C}_n$. Since it is more important to maximize sustained consistency for nodes with many contacts, we define in Equation 4.5 the overall sustained consistency, denoted by C_{System}, as the sum of the sustained consistency for each node, $C(n)$, weighted by the number of contacts of node n. That is:

$$C_{System} = \frac{\sum_{n=1}^{N} |\mathcal{C}_n| C(n)}{\sum_{n=1}^{N} |\mathcal{C}_n|} \tag{4.5}$$

In the infinite time horizon, a mobile node n changes its presence state infinitively often; I.e., the random variable $S_n(t)$ makes an infinite number of state transitions. In the remainder of this chapter, we will determine sustained consistency for various communication approaches using finite time horizon simulation. Thus, we consider a finite observation interval $(0,T]$. Let the counting process $K_n(t)$ denote the number of presence state changes of mobile node n during the time interval $(0,t]$. In the observation interval $(0,T]$, a mobile node n makes $K_n(T)$ presence state changes. We compute sustained consistency in the observation interval $(0,T]$ for a node n with respect to contact k, denoted by $C(n,k,T)$, by summing up over i, $1 \leq i \leq K_n(T) - 1$ in Equation 4.3. Analogous to Equations 4.4 and 4.5, we can compute overall sustained consistency in time T denoted as $C_{System}(T)$ by averaging $C(n,k,T)$ over all contacts and nodes, respectively. That yields:

$$C_{System}(T) = \sum_{n=1}^{N} \frac{\sum_{k \in \mathcal{C}_n} \sum_{i=1}^{K_n(T)-1} T_n(i+1) - \hat{T}_{k,n}(i)}{\left(T_n\left(K_n(T)\right) - T_n(1)\right) \sum_{m=1}^{N} |\mathcal{C}_m|} \tag{4.6}$$

4.2 Performance Bounds

4.2.1 Modeling Assumptions

The goal of this section is to derive upper bounds for sustained consistency and lower bounds for required control traffic with respect to different approaches for communicating presence information in highly partitioned MANET. To achieve this goal, we conduct a discrete-event simulation study based on the high-level stochastic model of the IM system presented in Section 4.2. Note that the idealized transmission semantics implicitly lead to upper bounds for sustained consistency, since all transmissions are successful and no information is lost. For the same reason, the transmission semantics lead to a lower bound for control traffic, since re-transmissions are not required.

For the high-level simulation study, we have to define a workload model by specifying the distribution of contacts, the distribution of dwell times in the individual states, and the mobility model of the mobile nodes. We will use a more sophisticated workload model than the one presented for the evaluation of PDI together with the IM application in Section 3.3.3. Note that for the computation of the sustained consistency, it is not required to define the exact presence state, but only the instances of time on which presence state changes occur. Thus, without loss of generality we can assume that for the set $\mathcal{S}$ of presence states holds $\mathcal{S} = \mathbb{N}_0$, the initial present state for each node is 0, and each presence state change increments the presence state by one.

To the best of our knowledge, no workload study for IM systems has been published in the academic literature. Thus, we have to rely on intuitive assumptions for both the distributions of the number and the popularity of contacts as well as for the dwell times. We assume a number of mobile nodes $N = 100$. For the contact distribution, we assign a popularity of $p(n) = c/n$ to each node n with some constant c. That is, the popularity is given by a Zipf distribution [Zip49]. Node n inserts some other node k into $\mathcal{C}_n$ with probability $p(k)$. Note that contact relations in most IM systems are bi-directional. That is, n is on the contact list of k and vice versa if either n has chosen k or k has chosen n as a contact. The probability for a contact relation between n and k is $1-(1-p(n))(1-p(k))$. Since it seems to be reasonable to have at most 10 contacts on average in a community of 100 people, we set the constant c to 0.87. This yields a mean cardinality of 8.7 for the set $\mathcal{C}_n$ with a variance of 9.9. The dwell time of mobile node n in presence state i, i.e., $T_n(i+1) - T_n(i)$ is drawn from a lognormal distribution. Since it seems reasonable to have presence state dwell times between 5 and 10 minutes, we use the shape parameter $\sigma = 0.6$,

and scale parameter $\mu = 6$. With this parameter setting, the distribution has a mean of $483s$, i.e., $8.05min$, and a standard deviation of $14.4s$.

The node mobility on a plane of size $1000m \times 1000m$ is determined by the RWP mobility model. In fact, we use an approach for the perfect simulation of the RWP mobility model that avoids non-stationary behavior in both the spatial node distribution and the distribution of node speed at the start of the simulation [BV05]. Both initial node positions and node speed are selected according to the stationary spatial and speed distributions, respectively. As a necessary condition of the existence of stationary spatial and speed distributions, this approach requires $v \in (0, v_{max}]$ for the node speed v. Furthermore, the approach selects pause times uniformly chosen from $[0, T_{hold}]$. In the experiments, we use the parameters $v_{max} = 2m/s$ and $T_{hold} = 120s$.

For all performance curves in the remainder of this section, we conducted finite time horizon simulations of length $T = 7200s$ for scenarios generated according to the descriptions above. For all performance measures, average values are calculated using 20 independent replicates. All figures include 99% confidence intervals. Since these confidence intervals are very tight, for some values their borders may not be visible in the figures.

4.2.2 Performance Bounds for Flooding-based Approaches

Recall that communicating a presence state change for a node to all its contacts constitutes a one-to-many communication operation. Flooding as described in Section 3.2.2 is a straightforward approach for implementing one-to-many communication in a MANET see, e.g., [JHMJ01]. Applications of flooding include, e.g, the location of cached data items in a MANET based caching system [FGS04]. Recall that with flooding, messages are relayed using local broadcast transmissions. A node that receives a flooded message from a nearby node for the first time relays it exactly once. Subsequently, a message reaches all nodes k with $D_{n,k}(t) < \infty$ for a source node n. Flooding may generate redundant message transmissions if a node relays a message that has already been received by all nodes in its transmission range. Several approaches for reducing the number of redundant transmissions have been proposed, e.g., [NTCS99], [TNS03], [YGK03]. However, since the goal of this section is to derive performance bounds for flooding-based approaches, we do not consider flooding optimizations here, but point out that SPEED presented in Section 4.3 incorporates flooding optimizations according to [TNS03].

As a first approach for disseminating presence information in a MANET we consider one-time flooding of the new presence state on each present state change.

For an algorithmic description of the state transitions induced in the discrete event system by a particular approach, we denote the current presence state of node k as s_k and the remote information stored about the state of node k at node n as $r_{n,k}$. For ease of exposition, we define $r_{n,n} := s_n$. Recall that we assume without loss of generality that each presence state change at node k increments the presence state s_k by one for $s_k \in \mathbb{N}_0$. Thus, the relation $r_{n,k} < r_{m,k}$ implies that the state of the shared contact $k \in \mathcal{C}_n \cap \mathcal{C}_m$ is more recent at node m than at node n. Recall that a presence state change event may trigger one or more zero-time communication events depending on the distance matrix D. Using the above definitions, state transitions of a system using one-time flooding can be described by Algorithm 4.1.

Obviously, the condition $D_{k,n}(t) < \infty$ does frequently not hold in highly partitioned MANET. For example, Figure 4.1 plots a MANET with $N = 100$ nodes with a transmission range of $R = 75m$ on a plane of $1000m \times 1000m$. Nodes that are located in the same network partition are drawn with identical symbols. Looking at a particular node (marked with a black circle in Figure 4.1) and all its contacts (marked with squares), we find that the node can only communicate his presence state directly to 5 out of 20 contacts. As a result, simple one-time flooding yields low sustained system consistency. Figure 4.2 plots the upper bound for the sustained consistency introduced by Equation 4.6 for Approach $F1$ applied in a system with $N = 100$ nodes and different transmission ranges. Additionally, it plots the control traffic given by the number of messages generated by the communication approach. The number of messages generated by flooding a message with source node k is given by the size of the network partition k is located in. Note that this constitutes a lower bound only in cases where no flooding optimization is used. However, since a flooding optimization also affects sustained consistency, we do not consider such optimization here. For an evaluation of the impact of flooding optimizations on the performance of a real system we refer to Section 4.4.

Recall that presence state changes occur infrequently due to the assumed distribution of dwell times. Thus, Figure 4.2 shows that the control traffic is low for all considered transmission ranges. Furthermore, sustained consistency is below 0.5 for weakly connected networks, i.e., networks with a transmission range below $100m$. For a sustained consistency of 0.9 and more, transmission ranges larger than $130m$ are

Algorithm 4.1 Approach $F1$ (One-time Flooding)

On each presence state change event for node k at time t **do**
  for all nodes $n \neq \mathcal{C}_k$ with $D_{k,n}(t) < \infty$ **do**
    Set $r_{n,k} := s_k$

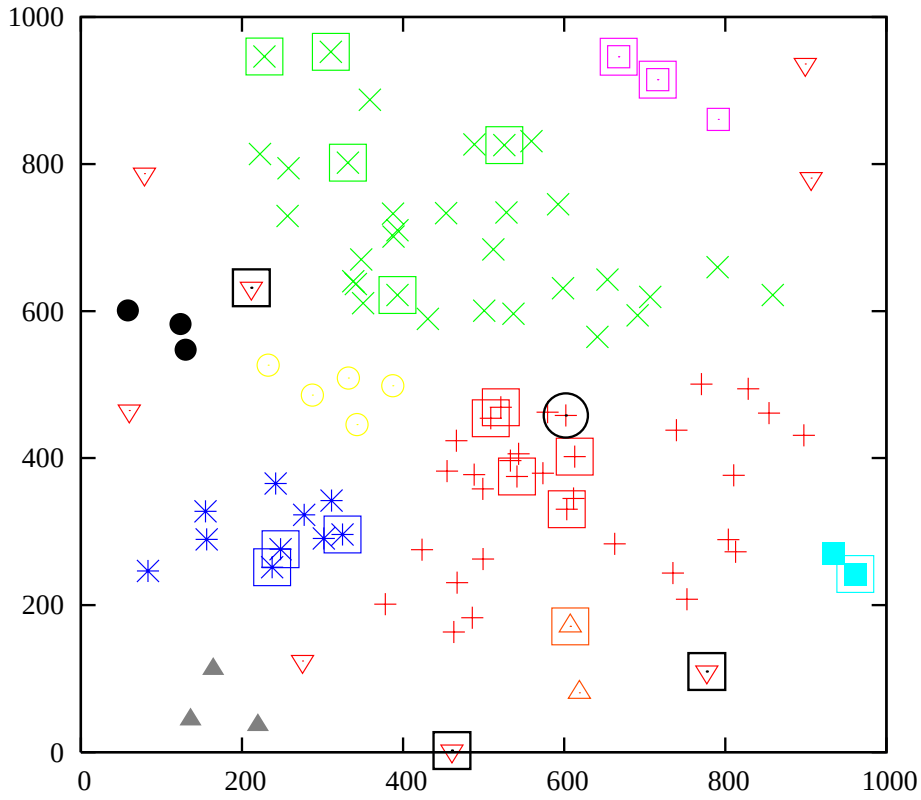

Figure 4.1. Illustrations of network partitions

required. We conclude from Figures 4.1 and 4.2 that more sophisticated communication approaches than simple flooding are required to achieve reasonable sustained consistency in MANET with low to medium connectivity, since in such networks most contacts are not reached by one-time flooding of state information.

Obviously, the sustained consistency of Approach $F1$ can be improved by periodically repeating the flooding operation hoping that new contacts become reachable. Hence, we consider a periodic flooding approach, in which a node schedules a timer event with deterministic time T_{Push} starting at a presence state change event. When the timer event occurs, the node pushes its current state to all contacts using a flooding operation that triggers several zero-time communication events. Subsequently, it reschedules the timer event. Algorithm 4.2 describes the state transitions in the discrete event system using the periodic flooding approach.

To illustrate the trade-off between sustained consistency and control traffic for Approach $F2$, we plot the upper bound for the sustained consistency versus the lower bound for the control traffic for different values of T_{Push} and different transmission ranges in Figure 4.3. Note that in this type of plots, a higher slope indicates that changing the parameter T_{Push} is more beneficial for the performance since it in-

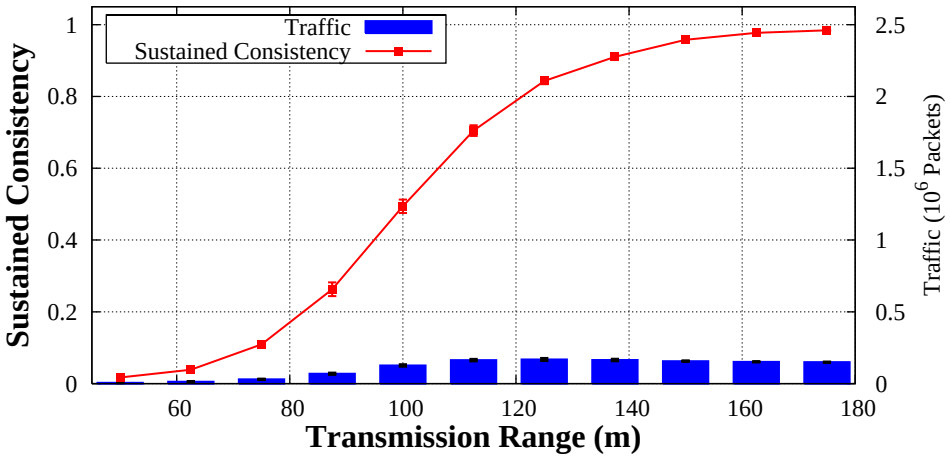

Figure 4.2. Sustained consistency and control traffic for an approach based on one-time flooding of presence state information

creases sustained consistency with a negligible increase of control traffic. Comparing Figure 4.3 to Figure 4.2 shows that periodic flooding with $T_{Push} = 10s$ substantially increases sustained consistency. The gain is most significant for $R = 75m$, where sustained consistency is increased by almost factor 4. Furthermore, Figure 4.3 shows that increasing the T_{Push} from $10s$ to $20s$ reduces control traffic by more than 49% for all transmission ranges. At the same time, sustained consistency is reduced, with a maximum reduction of 7% for a transmission range of $R = 75m$. Increasing T_{Push} further yields less significant traffic reduction while sustained consistency is more significantly affected. Sustained consistency shows highest sensitivity to T_{Push} for weakly connected networks, since in such networks most contacts are not reached by one-time flooding of state information as shown in Figure 4.1. We conclude from Figure 4.3 that periodic flooding of state information constitutes an appropriate way to disseminate state information in MANET. Nevertheless, we show in the next section how to further increase sustained consistency.

Algorithm 4.2 Approach $F2$ (Periodic Flooding)

On each timer event for node k at time t **do**
  for all nodes $n \neq \mathcal{C}_k$ with $D_{k,n}(t) < \infty$ **do**
    Set $r_{n,k} := s_k$
  Schedule a timer event at time $t + T_{Push}$

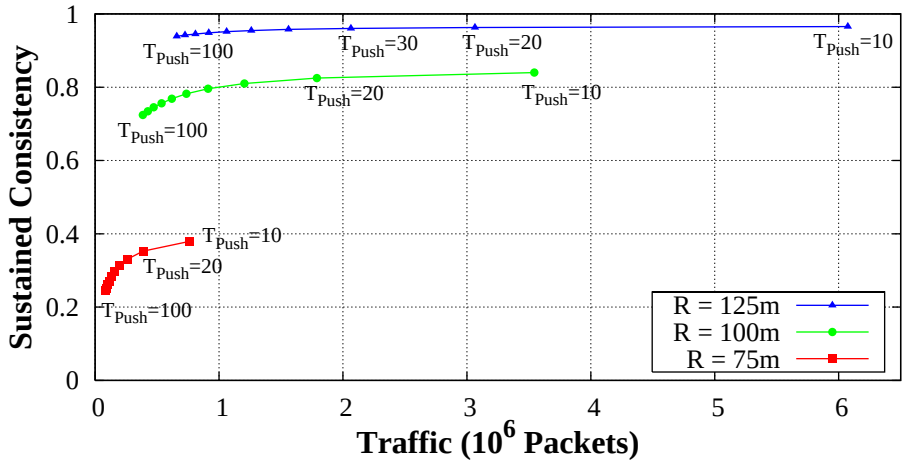

Figure 4.3. Trade-off between sustained consistency and control traffic for periodic flooding

4.2.3 Performance Bounds for Epidemic Information Dissemination

Note that both simple flooding and periodic flooding use direct communication. That is, presence information is directly communicated from a node k to a contact $n \in \mathcal{C}_k$. Recall from Section 2.4 that [GT02] shows that indirect communication (i.e., exploiting the mobility of intermediate nodes) improves the performance in delay-tolerant applications. Although dissemination of presence information is a highly delay-sensitive application, we show that indirect communication by epidemic dissemination of presence information can improve sustained consistency.

As basic concept of the approaches for epidemic dissemination of presence information analyzed in this section, a node n pushes new state information at time t to some nodes m with $D_{n,m}(t) < \infty$. Subsequently at time t', a node n' polls some nodes m with $D_{n',m}(t') < \infty$ for up-to-date state information about other nodes k. Polling is performed periodically in intervals T_{Poll}. Based on the communication events that are triggered by presence state change and timer events, we distinguish between three types of approaches, which differ in the sets of nodes that are involved in each communication operation.

Epidemics via contacts (E1): On a presence state change event at time t, a node n pushes new state information to all contacts $k \in \mathcal{C}_n$ with $D_{n,k}(t) < \infty$. Furthermore, on a timer event at time t', a node n polls more up-to-date state information for its contacts $k \in \mathcal{C}_n$ from all nodes m with $D_{n,m}(t') < \infty$, $m = k$ or $k \in \mathcal{C}_m$. This approach implements epidemic information dissemination only among nodes with

shared contacts.

Epidemics via reachable nodes (E2): On a presence state change event at time t, a node n pushes new state information not only to its contacts, but to all nodes m with $D_{n,m}(t) < \infty$. Furthermore, on a timer event at time t', a node n polls more up-to-date state information about its contacts $k \in \mathcal{C}_n$ from all nodes m, $D_{n,m}(t') < \infty$. Note that this approach comprises two stages: On the first stage, nodes exchange information about non-contacts, while on the second stage a node only gathers information about its contacts.

Epidemics via all nodes (E3): On a timer event at time t, a node n gets presence state information that is more up-to-date than the local information from all nodes m with $D_{n,m}(t) < \infty$. In this approach all nodes exchange information about all other nodes regardless if they have a contact relation. The algorithmic description of Approach $E2$ is given in Algorithm 4.3.

Algorithm 4.3 can be easily modified to represent Approach $E1$ by running the for-loop in line 2 only for nodes m with $n \in \mathcal{C}_m$, and intensifying the condition in line 6 to $k \in (\mathcal{C}_n \cap \mathcal{C}_m) \cup \{m\}$. Furthermore, the algorithmic description can be modified to represent Approach $E3$ by omitting lines 1 to 3 and running the for-loop in line 6 for $k = 1, 2, \ldots, n-1, n+1, \ldots, N$.

Note that another possible approach lies in incorporating periodical pushing of state information in Approaches $E1$ and $E2$. However, we found in experiments not shown here that additional push operations have little impact on sustained consistency while substantially increasing control traffic. Thus, in the remainder of this chapter we do not consider such approaches.

To compare the performance of the Approaches $E1$ to $E3$ with each other and with the periodic flooding approach $F2$, we derive upper bounds for the sustained consistency that can be achieved by each approach. Obviously, optimum sustained

Algorithm 4.3 Approach $E2$ (Epidemics via Reachable Nodes)

```
1 On each presence state change event for node n at time t do
2   for all nodes m = 1, ..., N with D_{n,m}(t) < ∞ do
3     Set r_{m,n} := s_n
4 On each a timer event for node n at time t do
5   for all nodes m with D_{n,m}(t) < ∞ do
6     for all nodes k ≠ C_n do
7       if r_{n,k} < r_{m,k} then
8         Set r_{n,k} := r_{m,k}
9   Schedule a timer at time t + T_Poll
```

consistency is achieved with any approach for presence information dissemination if communication between two nodes n and m takes place as soon as a connection between n and m is available, i.e., at time t with $D_{n,m}(t') = \infty$ for $t' < t$ and $D_{n,m}(t') < \infty$ for $t' > t$. We can easily modify the Approaches $F2$ and $E1$ to $E3$ to accomplish this requirement by setting $T_{Push} \rightarrow 0$ and $T_{Poll} \rightarrow 0$, respectively. We refer to these approaches as $F2_{opt}$ and $E1_{opt}$ to $E3_{opt}$. This modification yields optimum performance with respect to sustained consistency. However, in a real application, the control traffic generated by this modification would be unbounded. To derive a lower bound for traffic requirements with respect to each approach, we compute the traffic for initial pushing new presence state after a state change event similar to Section 4.2.2. Additionally, we assume that the traffic used for a successful exchange of state information in subsequent push or poll operations is equal to the number of hops on the shortest path between the communicating nodes. That is, if node n sets $r_{n,k}$ to $r_{m,k}$ as result of a communication operation with node m at time t, we count traffic of $D_{n,m}(t)$ packets. If we have more than one possible synchronization operation after a timed event, the operations are performed in ascending order of the distance given by $D(t)$. We plot the upper bounds for the sustained consistency and the lower bound for the control traffic of the different optimum approaches in Figure 4.4 and 4.5, respectively. We find that using indirect communication increases sustained consistency by up to 0.22 compared to direct communication. Here, $E3$ clearly outperforms $E1$ and $E2$, which increase performance by up to 0.06 and 0.11, respectively. Looking at the control traffic generated by the approaches in Figure 4.5, we find that $E1$ and $E2$ generate almost identical traffic. That indicates that $E1$ and $E2$ perform almost the same number of communication operations. Nevertheless, due to the fact that presence state information is not only stored by contacts, but by all nodes, $E2$ disseminates presence state information faster that $E1$. $F2$ generates lower control traffic than $E1$ and $E2$. This holds in particular for medium-connected networks, where epidemic dissemination of state information can be performed over long distances with $E1$ and $E2$. Full epidemic dissemination as employed in Approach $E3$ generates most traffic, since information is exchanged over multiple hops in many cases. We conclude from Figures 4.4 and 4.5 that indirect communication using epidemic dissemination can significantly improve performance in terms of sustained consistency. However, the control traffic generated by the "epidemic" synchronization operations must be considered when implementing fully epidemic dissemination. Motivated by Figures 4.4 and 4.5, we focus on Approach $E2$ in the remainder of this chapter, since $E2$ constitutes a reasonable trade-off between sustained consistency and control traffic.

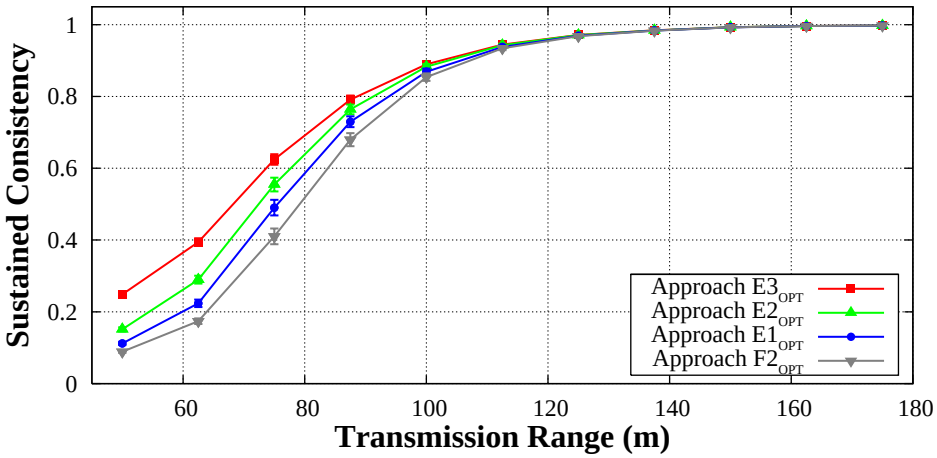

Figure 4.4. Upper bounds for the sustained consistency achieved by different approaches

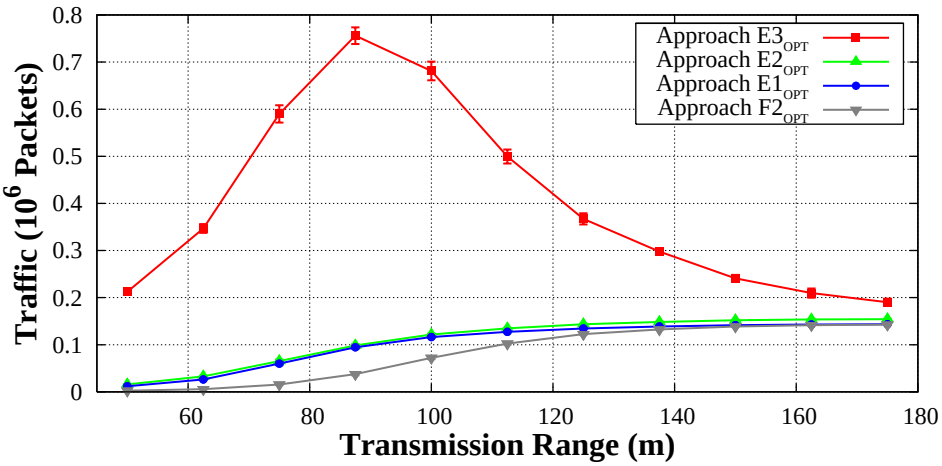

Figure 4.5. Lower bounds for the control traffic generated by different approaches

4.3 A System for Disseminating Presence Information

In this section, we propose *SPEED*, the System for Presence information Exchange by Epidemic Dissemination in MANET that uses a combination of a push and a poll approach similar to approach $E2$ described in Section 4.2.3. That is a node pushes its new presence state on a presence state change to nearby nodes by flooding. Furthermore, each node n also polls the presence states of its contacts $k \in \mathcal{C}_n$ periodically by flooding. Opposed to Section 4.2.3, flooding is not performed for the entire network partition, but only with a limited scope given by a maximum number of hops.

Furthermore, an age is assigned to each information about the presence state of a remote node. A node polls the state of a contact only if the age has exceeded a certain threshold. A contact itself or any node having more up-to-date presence information responds to a Poll message. As further difference to Section 4.2.3 Response messages are not transmitted using unicast-communication, but transmitted by an approach denoted as *directed flooding*. Directed flooding constitutes a compromise between traffic efficiency and epidemic dissemination. In contrast to a conventional flooding operation, with directed flooding a message is only retransmitted by the nodes m that have received it from a node n with $D_{m,k}(t) < D_{n,k}(t)$ for the inquiring node k. All intermediate nodes overhear Response messages and extract new presence information. Information are cached to implement epidemic dissemination and to decrease control traffic by sending Response messages from nearby nodes. In the remainder of this section we describe all mechanisms in more detail.

In addition to a list of the current state of its contacts denoted as *contact list*, each node n has a data structure $cache_n$ that stores information about the presence state of nodes $k \neq \mathcal{C}_n$. Note that $cache_n$ has at most $N - |\mathcal{C}_n|$ entries. We do not consider cache replacement, since required cache size is small. Consistent with Section 4.2, we denote presence information of node k stored at node n by $r_{n,k}$, regardless if it is stored in the contact list or in the cache. Note that cached entries do not affect sustained consistency. To determine how up-to-date a state information is, we assign an age denoted by $a_{n,k}$ to all $r_{n,k}$. For convenience $a_{n,n}$ is set to 0. Note that $a_{n,k}$ constitutes a time difference. Thus, $a_{n,k}$ can be easily computed using the local clock of a node and there is no need for synchronizing clocks.

SPEED uses messages of types Push, Poll, and Response. Besides payload, each message contains the additional fields *Type*, *Hops*, *TTL*, *Origin* and *Inquirer* that are described in Table 4.1. Upon receiving a message the *Hops* field is incremented by one and the *TTL* field is decremented by one. To approximate the distance matrix D defined in Section 4.1, each node maintains an approximate distance to all nearby nodes in a data structure d. That is, the receiving node n stores the distance

Field	Description
Type	Message Type (Push, Poll, Response)
Hops	Distance to origin node in hops
TTL	Maximum number of forwarding hops
Origin	Origin node of this message
Inquirer	Identification of inquiring node in Response

Table 4.1. Description of message fields

to the origin node m of the message, i.e., $d_{n,m} := Message.Hops$. Since distances change due to node mobility, we assign each distance with a timeout of $10s$. Experiments have shown that such timeout is sufficient, since each node that has to forward a Response message has also received the corresponding Poll message. If node n has no up-to-date distance information for node m, $d_{n,m}$ is set to ∞. Note that each node n only stores the column k of the matrices $r_{n,k}$, $a_{n,k}$ and $d_{n,k}$.

When a node k changes its presence state at time $T_k(i)$, $1 \leq i \leq K_k(T)$, it pushes the new state to all nodes within a distance of at most TTL_{Push} using scoped flooding based on the field $Message.TTL$. The Push message comprises a tuple (k, s), where $s = S_k(T_k(i))$ constitutes the new presence state of node k. Each node n that receives the message sets $r_{n,k} := s$ and $a_{n,k} := 0$. To synchronize contacts for that no current state information are received by a Push message, each node polls the state of its contacts periodically in time intervals T_{Poll}. To reduce the number of messages, a single Poll message can contain queries for the states of multiple contacts. A node assures that the information about the presence state of its contacts is more up-to-date than some information up-to-date timeout TO_{iu}. A contact is only polled, if its state information in the contact list is older than TO_{iu}. A Poll message contains entries that comprise a tuple (k, a), where k is a contact and a the age of the information in the contact list. The age information enables nodes that receive a Poll message to decide if they have more up-to-date state information. Like a Push message, a Poll message is flooded with scope TTL_{Poll}. Note that an alternative design approach is to implement expanding ring search as used, e.g., for the optimization of the route discovery process in AODV [PRD03]. That is, to successively flood the Poll message with increasing scope ttl, $1 \leq ttl \leq TTL_{Poll}$, until Response messages for all contacts in the Poll message are received. However, we found that this approach does not improve performance since the fraction of Poll messages that trigger Response messages for all contacts is low for small TTL_{Poll}. An algorithmic description of requesting the presence states of all contacts is given by Algorithm 4.4.

Algorithm 4.4 Node n polls its contacts

```
Poll := ∅
for all nodes k ∈ C_n do
  if a_{n,k} > TO_iu then
    Add (k, a_{n,k}) to Poll
if Poll ≠ ∅ then
  Flood Poll with TTL_Poll
```

If a node receives a Poll message and has more up-to-date state information either in its contact list or in $cache_n$ for one or more contacts in the Poll message, it generates a Response message. Entries of a Response message comprise a triple (k, a, s) of the contact k, the age of the information a and the contacts presence state s. If the local state information for a contact is sufficient recent, i.e., the age of that information is lower than the timeout TO_{iu}, the contact is removed from the Poll message, since it is not necessary that other nodes send Response messages for this contact. That applies in particular when the contact itself responds. A non-empty Poll message is forwarded, if $TTL > 0$ to implement flooding. The message is not immediately forwarded, but inserted into a forward queue together with a random value denoted as Δ. An algorithmic description of the actions upon receiving a Poll message is given by Algorithm 4.5.

Recall that Response messages are returned to the inquiring node using directed flooding. To implement directed flooding, a node r broadcasts a Response message, where TTL is set to the distance to the inquiring node. A node n relays the message if it is closer to inquiring node q than responder r, i.e., $d_{n,q} < d_{r,q}$. Note that node n can determine $d_{r,q}$ by summing up the number of hops and remaining TTL of the Response message. A node n within the transmission range of a responding

Algorithm 4.5 Node n receives a Poll message

```
Response := ∅
for each entry (k, a) ∈ Poll do
  if k = n then
    Add (k, 0, s_n) to Response
    Remove entry (k, a) from Poll
  if k ∈ C_n or k ∈ Cache_n then
    if a_{n,k} < a then
      Add entry (k, a_{n,k}, r_{n,k}) to Response
    if a_{n,k} < TO_iu then
      Remove entry (k, a) from Poll
if Response ≠ ∅ then
  Response.TTL := d_{n,Poll.Inquirer}
  Δ := uniformrandom(0, 1]
  Response → ForwardQueue with Δ
if Poll ≠ ∅ and Poll.TTL > 0 then
  Δ := uniformrandom(0, 1]
  Poll → ForwardQueue with Δ
```

or relaying node overhears the message and extracts new information about nodes k contained in the Response message. For each entry (k, a, s) in the Response message, node n searches for k in its contact list and in $cache_n$, respectively. If k is not found or the presence information s in the Response message is more up-to-date than the local information, i.e., $a < a_{n,k}$, the local state information $r_{n,k}$ is updated. Otherwise, the entry (k, a, s) is eliminated from Response message, since the node n itself has already responded with at least as up-to-date information with high probability. Thus, we avoid forwarding stale or redundant information. Furthermore, if it holds $d_{n,q} > d_{r,q}$, SPEED eliminates all entries, which are not more up-to-date than information in the received Response message from Response messages pending in the forward queue, since a node closer to the inquirer has already responded. An algorithmic description of the actions upon receiving a Response message is given by Algorithm 4.6 on page 110.

As a disadvantage of flooding, redundant transmissions of Push and Poll messages may occur. That is, a node receives the same message from multiple sources. We found out that more than 80% of all Poll messages are redundant with the approach described so far. To eliminate redundant transmissions we use a counter based flooding approach [TNS03].

Such approach probabilistically reduces the number of forwarding nodes while keeping the number of covered nodes almost constant. Before forwarding a message, SPEED adds a random delay $\delta \in (0, 1]s$, which is chosen according to a uniform distribution. The message is stored in the forward queue until the delay δ has elapsed. If the node n receives an identical message from Tr or more neighbors for a threshold Tr, before the end of the delay, node n discards the message, because with high probability all neighbors of node n have already received that message. We found out that an optimal value of the threshold Tr depends on the message type. Thus, we use a threshold Tr_{Push} for Push messages and Tr_{Poll} for Poll messages, respectively.

To illustrate the impact of filtering and directed flooding of Response messages, Figure 4.6 illustrates forwarding of a Response message for $N = 100$ and $R = 125m$. The node that has generated a Response message is marked with a black circle. Furthermore, nodes that forward Response messages are marked with boxes and their transmission ranges are drawn as circles. The origin node of each Response message is marked with a double box. Identical dashed lines mark nodes that forward the same Response messages. The impact of directed flooding is well illustrated by the message drawn with medium dashed lines. The message propagates from the responding node to the polling node. Furthermore, filtering of Response messages is illustrated by both Response messages drawn with dashed lines. These

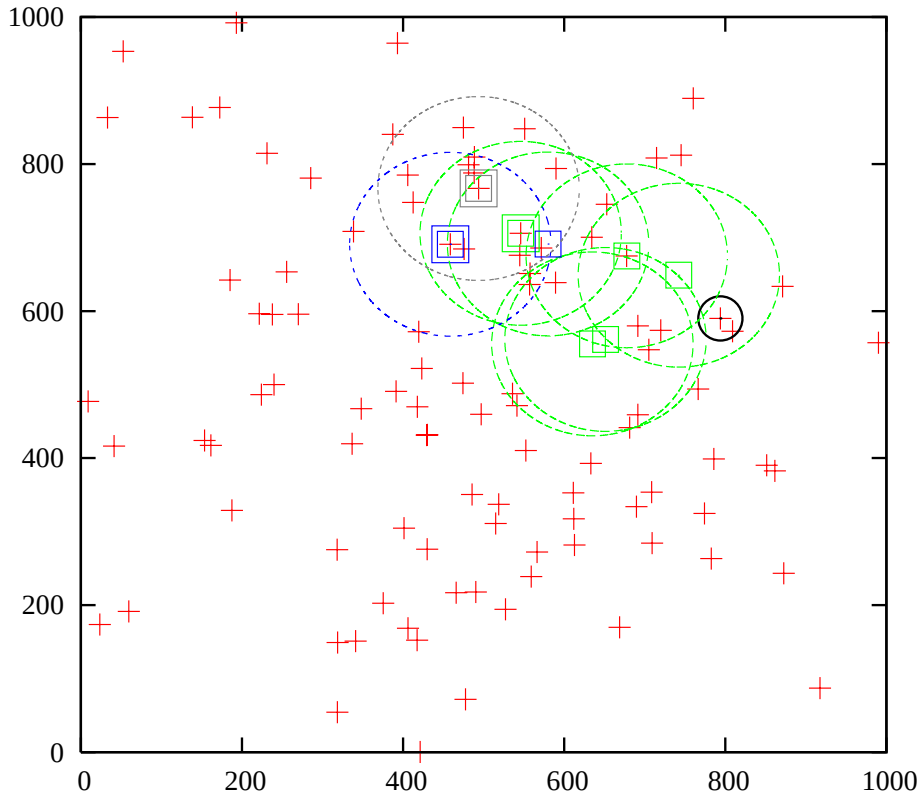

Figure 4.6. Forwarding of a Response message

messages are not forwarded by nodes that have already received the first message. We conclude from Figure 4.6 that filtering and directed flooding of Response messages constitutes a reasonable mechanism for response forwarding that ensures low control traffic.

4.4 Performance Studies

4.4.1 Simulation Methodology

Opposed to the high-level stochastic simulation performed for computation of performance bounds, we provide an evaluation of SPEED in a realistic setup in this section. For evaluation purpose, we use the network simulator ns-2 instead of the high-level discrete event simulator used in Section 4.2. ns-2 considers communication delays, packet collisions, and interference. We implemented SPEED as described in Section 4.3 as an ns-2 application. The application uses the UDP/IP protocol stack as well as the IEEE 802.11 MAC and physical layer running in ad hoc mode. The payload for each entry in a Push, Poll or Response message has a size of 20 Bytes. Con-

sistent with Section 4.2, we analyze scenarios with $N = 100$ mobile nodes in a plane of $1000m \times 1000m$ and use different transmission ranges to account for different node density. Contacts, state dwell times and node mobility are chosen according to the probabilistic models presented in Section 4.2.1. In fact, we use the same input data for identical scenarios to assure that the results of this section are comparable to Section 4.2. Thus, we performed finite time horizon simulations of length $7200s$ for all points in the performance curves. The curves show the mean values derived from 20 independent replicates. We calculated 99% confidence intervals for all points in the performance curves. For readability, we did not include the confidence intervals in most of the graphs. However, we state that all confidence intervals are tight, i.e., $< 10\%$ of the mean value.

4.4.2 Impact of System Parameters on Sustained Consistency and Control Traffic

To determine an optimal parameter setting, we conducted excessive simulations for exploring the entire design space. Goal of the simulation was to find a parameter setting that constitutes a reasonable trade-off between sustained consistency and control traffic. We omit the results of the simulation studies and show the resulting parameter setting in Table 4.2. In the remainder of this section, we keep all but one parameter fixed to the setting in Table 4.2, and change the remaining parameter to illustrate its impact on the system performance.

In Figure 4.7, we plot sustained consistency vs. control traffic for polling intervals $T_{Poll} \in [10s, 600s]$. We find that traffic is highly sensitive to the polling interval. The sensitivity is most significant for a transmission range of $75m$. At this transmission range the sustained consistency is reduced most, i.e., by 12%. Further reduction of the polling interval yields a less significant reduction of traffic with a higher reduction of sustained consistency in each step. We conclude from Figure 4.7 that reducing the polling interval beyond $60s$ is not appropriate.

Parameter	Value
T_{Poll}	$60s$
TTL_{Push}	10
TTL_{Poll}	6
Tr_{Push}	3
Tr_{Poll}	1
TO_{iu}	$100s$

Table 4.2. Default settings of system parameters

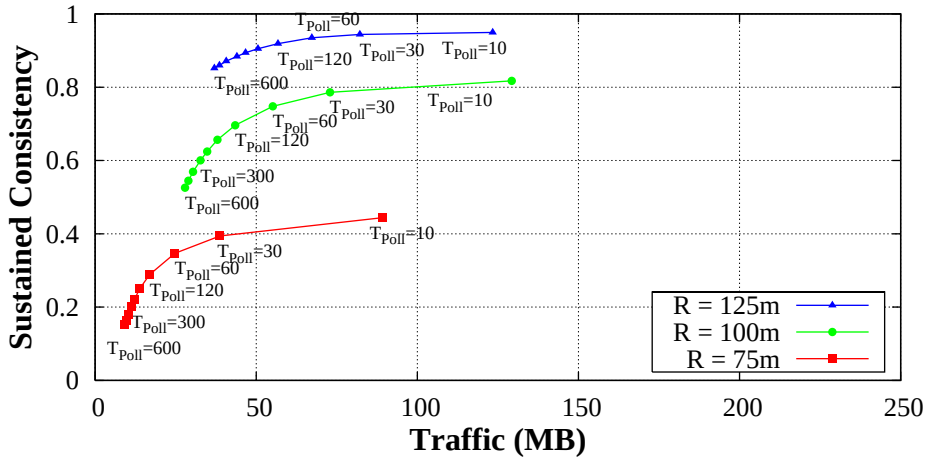

Figure 4.7. Sensitivity to polling interval

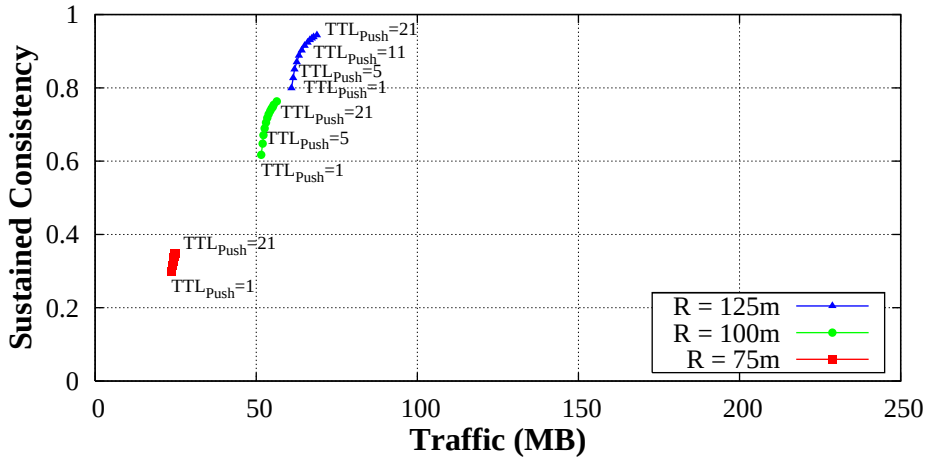

Figure 4.8. Sensitivity to scope of Push messages

Figure 4.8 plots sustained consistency vs. control traffic for different scopes of pushing new presence state information, $TTL_{Push} \in [1, 21]$. We find that the sustained consistency is highly sensitive to TTL_{Push}. However, the counter-based flooding approach for Push messages causes the traffic to increase gracefully with TTL_{Push}. That is, the sustained consistency can be increased by 16% to 28% by increasing TTL_{Push} from 1 to 11. At the same time, the traffic increases by at most 12%. We conclude from Figure 4.8 that using a large scope for pushing new presence information is an appropriate tool for increasing sustained consistency with low control traffic.

Figure 4.9 plots sustained consistency vs. control traffic for different scopes of Poll messages $TTL_{Poll} \in [1, 21]$. Recall that polling state information is a more frequent operation than pushing state information. Thus, traffic is more sensitive to

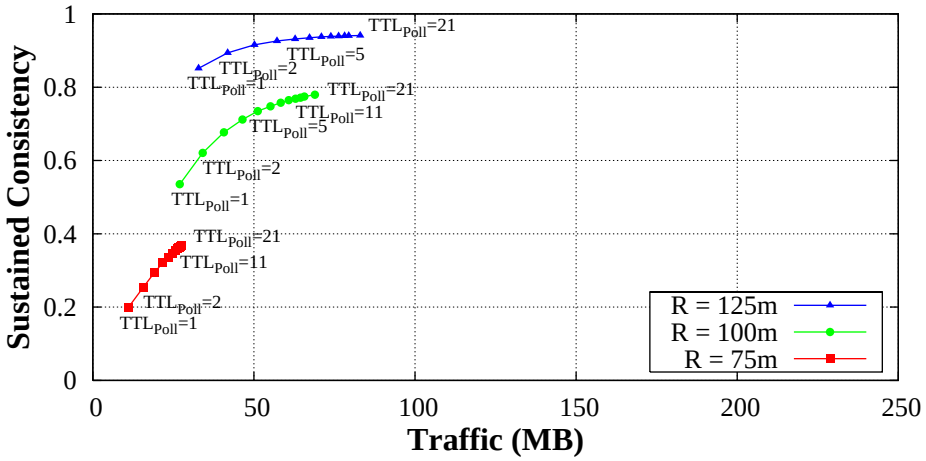

Figure 4.9. Sensitivity to scope of Poll messages

TTL_{Poll} despite of the counter-based approach for flooding Poll messages. Nevertheless, increasing TTL_{Poll} from 1 to 6 yields a significant increase of sustained consistency for low and medium transmission range. Further increasing TTL_{Poll} does not significantly increase sustained consistency, but increases traffic by up to 18%. We conclude from Figure 4.9 that $TTL_{Poll} = 6$ yields reasonable sustained consistency at moderate traffic.

In the next experiments, we analyze the impact of the thresholds Tr_{Push} and Tr_{Poll} for the counter-based flooding approaches for Push and Poll messages. Sustained consistency vs. control traffic for values $Tr_{Push}, Tr_{Poll} \in [1, 10]$ are shown in Figures 4.10 and 4.11, respectively. Figure 4.10 indicates a low sensitivity of the traffic to Tr_{Push} due to the low frequency of push operations. Nevertheless, it is essential to set Tr_{Push} to a value of 3, since these values increase sustained consistency by up to 4% compared to $Tr_{Push} = 1$ while increasing control traffic by at most 5% for high transmission ranges. Larger values of Tr_{Push} yield higher control traffic without significantly increasing sustained consistency. Figure 4.11 shows that the counter-based flooding approach for Poll messages reduces traffic by up to 36%. For high node density ($R = 125m$) $Tr_{Poll} = 1$ is optimal, since sustained consistency is almost constant. For $R = 75m$ increasing Tr_{Poll} from 1 to 2 leads to a gain of sustained consistency by 8%, but the additional traffic is much too high (28%). We conclude from Figure 4.10 and 4.11 that the optimal thresholds are $Tr_{Push} = 3$ and $Tr_{Poll} = 1$, respectively.

The last experiment shown in Figure 4.12 plots sustained consistency vs. control traffic for information up-to-date timeouts $TO_{iu} \in [0, 1500]s$. This figure shows that setting TO_{iu} to $100s$ significantly reduces control traffic with only a slight impact

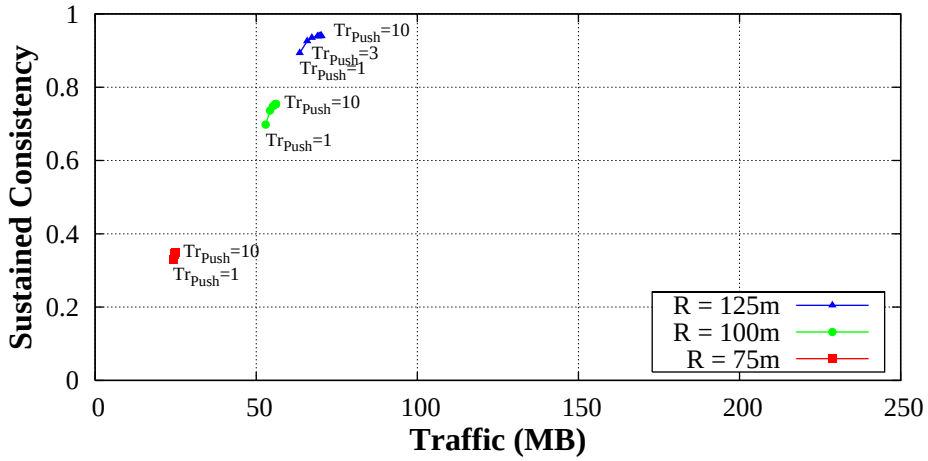

Figure 4.10. Sensitivity to threshold for Push message forwarding

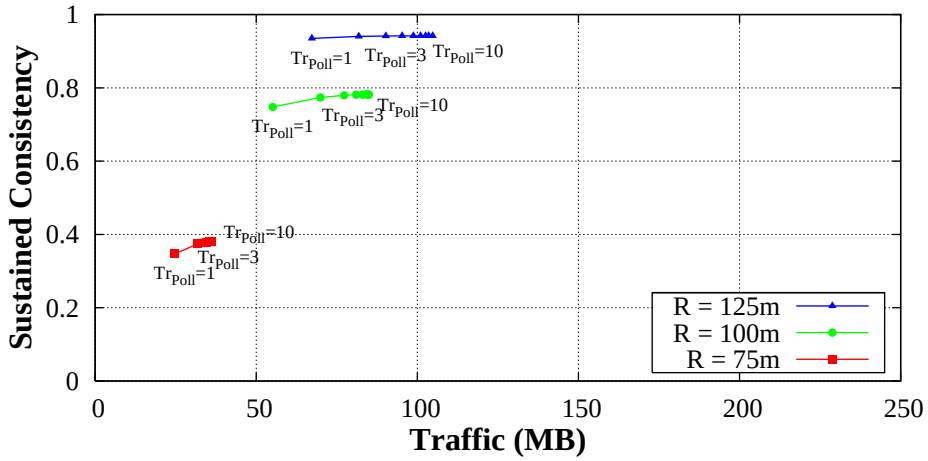

Figure 4.11. Sensitivity to threshold for Poll message forwarding

on sustained consistency.

4.4.3 Adaptive Selection of Protocol Parameters

Comparing Figure 4.7, 4.9, and 4.12 to Figures 4.8, 4.10, and 4.11 indicates that the choice of the poll interval T_{Poll}, the scope for Poll messages TTL_{Poll}, and the information up-to-date timeout TO_{iu} is much more crucial than the choice of the other system parameters. In particular, the choice of the three parameters is highly dependent on the transmission range R. E.g., using $T_{Poll} = 60s$ instead of $T_{Poll} = 10s$ has almost no impact on sustained consistency for $R = 125m$ while it reduces sustained consistency by 22% for $R = 75m$. Thus, inspired by [TNS03] we propose an adaptive approach for selecting T_{Poll}, TTL_{Poll}, and TO_{iu}. That is,

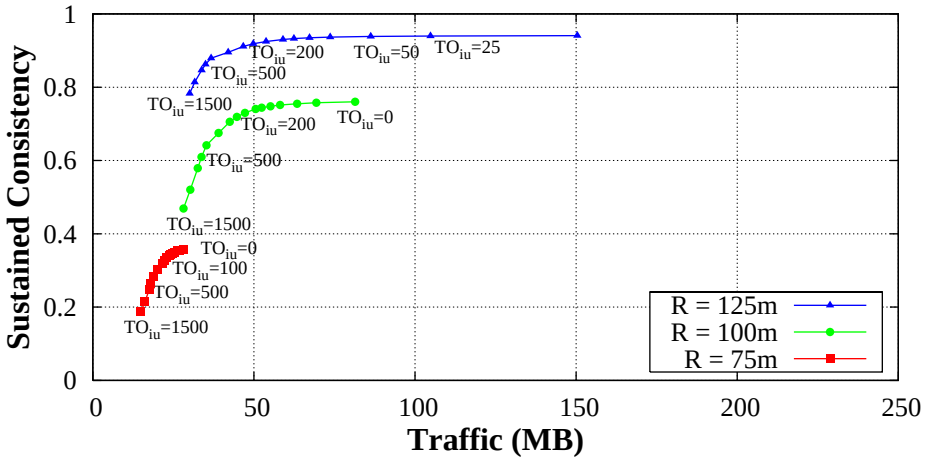

Figure 4.12. Sensitivity to information update timeout

each node keeps track of the number of nodes currently located in the transmission range. That can be easily accomplished by a node n by counting the nodes m with $d_{n,m} = 1$. Depending on the current number of nodes, T_{Poll}, TTL_{Poll}, and TO_{iu} are set according to Table 4.3. We do not show the experiments for determining these values, but refer to Section 4.4.4 for an illustration of the impact of the adaptive selection on the sustained consistency and control traffic of SPEED.

4.4.4 Performance Comparison with an Optimized Flooding Approach

To show the effectiveness of SPEED, we compare sustained consistency and control traffic to the performance bounds calculated for $E2$ and to the values achieved by an optimized flooding approach inspired by $F2$. To achieve comparable results, the flooding approach uses a counter-based flooding optimization with threshold $Tr_{Push} = 3$, which has been proven optimal in a simulation study not shown here. Furthermore, the flooding approach uses adaptive selection of $T_{Push_flooding}$ with settings given by Table 4.3.

Figures 4.13 and 4.14 compare sustained consistency and control traffic of the

# Neighbors	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	> 14
T_{Poll} (s)	20	30	40	50	55	55	65	65	75	75	75	75	80	100	105	110
TTL_{Poll}	7	6	5	5	5	5	4	4	4	3	3	3	2	2	1	1
TO_{iu} (s)	30	40	50	60	75	90	100	150	150	150	200	200	200	250	250	250
T_{Push} (s)	10	10	10	20	20	25	30	50	50	75	75	100	100	150	150	150

Table 4.3. Adaptive parameter settings for SPEED and the flooding based approach

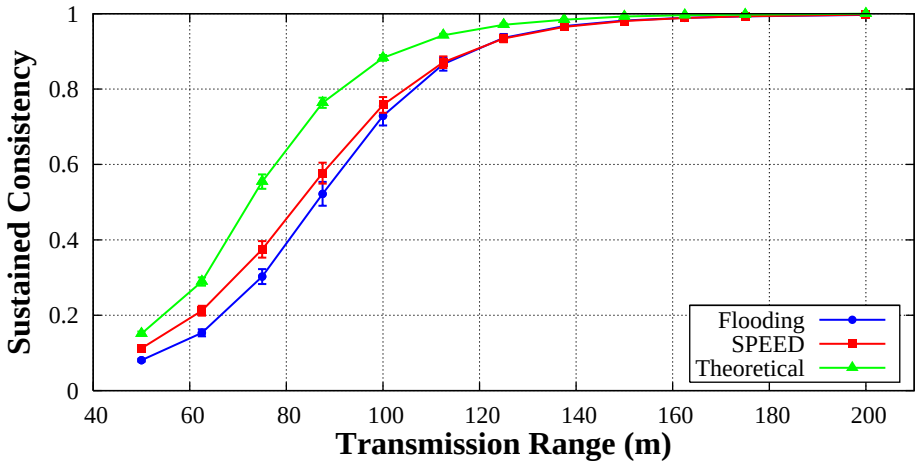

Figure 4.13. Sustained consistency for different design alternatives

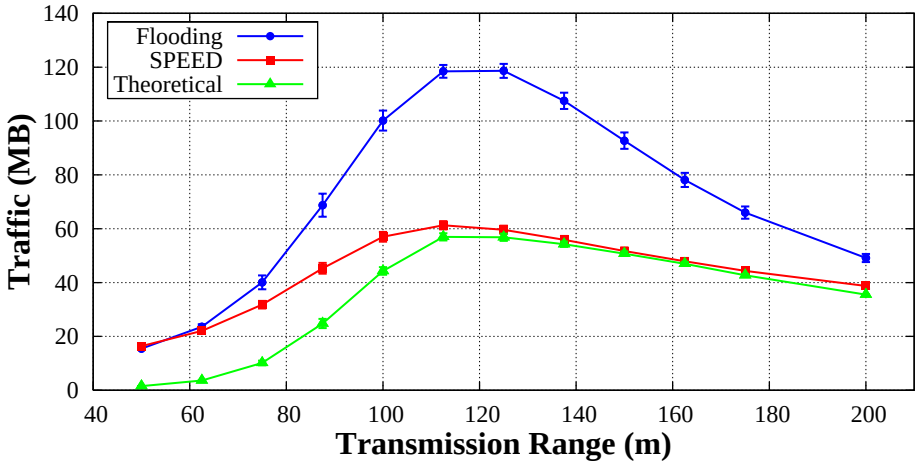

Figure 4.14. Control traffic for different design alternatives

flooding approach, the theoretical optimum and SPEED for different transmission ranges. For calculating the control traffic generated by the theoretical optimum, we assume that each message has a size of 92 bytes. Again, the parameters given by Table 4.2 are used for SPEED, with exception of T_{Poll}, TTL_{Poll}, and TO_{iu}, which are chosen adaptively according to Table 4.3.

Figure 4.13 shows that SPEED performs equal or better compared to the flooding approaches in terms of sustained consistency. In particular in scenarios with transmission ranges $R \leq 115m$, SPEED is superior to the flooding approach with a performance gain of up to 20%. Recall from Figure 4.1 that network partitions frequently occur in such scenarios which constitute a hurdle to flooding of presence information, but can be traversed by epidemic dissemination. For transmission ranges $R > 115m$, the flooding-based approach performs equal to SPEED with respect to

sustained consistency, since increasing connectivity favors flooding. However, Figure 4.14 shows that in such scenarios SPEED saves up to 48% of the control traffic generated by the flooding approach. Note that the flooding approach generates most traffic in scenarios with medium transmission ranges. These scenarios exhibit large network partitions; however, most nodes are connected by routes with many hops. Thus, flooded messages must be retransmitted several times to reach every node in a network partition. In contrast, the combination of epidemic dissemination, polling, and directed flooding used by SPEED ensures that presence information is only transmitted between nearby nodes.

Comparing both the sustained consistency and the control traffic generated by SPEED to the upper and lower bound, respectively, shows that with the configuration presented in Tables 4.2 and 4.3 performs close to the optimum with respect to control traffic. However, there is much space for improvement with respect to sustained consistency. Recall that Section 4.4.2 demonstrated a significant trade-off between traffic and sustained consistency. Thus, depending on the requirements of the application the parameters given by Tables 4.2 and 4.3 can be easily modified to achieve a higher sustained consistency at the expense of increased control traffic.

4.5 Summary

In this chapter, we provided a comprehensive analysis of approaches for disseminating frequently changing presence information in highly partitioned mobile ad hoc networks. We proposed sustained consistency as a measure for the coherence of the presence information distributed within the system. As further performance measures of interest we consider the control traffic. We compared several approaches for presence information dissemination in terms of upper bounds for sustained consistency and lower bounds for the control traffic. All bounds were derived by discrete-event simulation based on a high-level stochastic model of the IM system. We showed that an approach that uses epidemic dissemination of presence information by one-time pushing of state information to all reachable nodes and successive periodical polling for up-to-date state information of all contacts constitutes the method of choice.

Based on the performance bounds, we presented SPEED, the System for Presence information Exchange by Epidemic Dissemination in a MANET. SPEED comprises optimized mix of controlled flooding, caching, and epidemic dissemination of information. In a detailed simulation study we illustrated the impact of different design parameters on system performance and proposed adaptive selection of the poll interval, the scope for flooding poll messages, and the information timeout based on

the number of nodes located in the transmission range. A comparison to an optimized approach based on periodical flooding of presence information reveals that epidemic information increases sustained consistency for small transmission ranges, while it reduces control traffic for large transmission ranges. In fact, for transmission ranges up to $115m$ SPEED achieves up to 20% higher sustained consistency than the flooding-based approach while for larger transmission ranges, epidemic information dissemination saves up to 48% control traffic. A comparison between SPEED and the theoretical bounds reveals that the presented configuration of SPEED achieves high traffic efficiency at the expense of reduced sustained consistency. However, the sensitivity studies show that SPEED can be easily configured to achieve higher sustained consistency at the expense of increased control traffic.

Algorithm 4.6 Node n receives a Response message

if Response.Inquirer = n **then**
  for each entry $(k, a, s) \in Response$ **do**
    if $a < a_{n,k}$ **then**
      Set $r_{n,k} := s$ and $a_{n,k} := a$
else
  $Redundant = \varnothing$
  for each entry $(k, a, s) \in Response$ **do**
    if $k = n$ **then**
      remove entry (k, a, s) from $Response$
    if $k \in \mathcal{C}_n$ or $k \in Cache_n$ **then**
      if $a \leq a_{n,k}$ **then**
        Add k to $Redundant$
      if $a < a_{n,k}$ **then**
        Set $r_{n,k} := s$ and $a_{n,k} := a$
      else
        remove entry (k, a, s) from $Response$
    else
      Add entry (k, a, s) to $Cache_n$
  $q := Response.InquiringNode$
  $r := Response.Origin$
  if $d_{n,q} < d_{r,q}$ and $Response.TTL > 0$ **then**
    if ForwardQueue contains a response to this poll **then**
      Add $Response$ to queued response
      whereas older entries are replaced
    else
      $\Delta := uniformrandom(0, 1]$
      $Response \rightarrow ForwardQueue$ with Δ
  else
    for each response message in $ForwardQueue$ **do**
      Remove all (k, a, s) with $k \in Redundant$

Chapter 5

Stochastic Modeling of Epidemic Information Dissemination

As recalled in Section 2.4, buffers of finite size have been neither considered in current proposals for EID systems nor in performance models of such systems. Furthermore, all modeling approaches perform transient analysis, and some even require off-line simulations to determine model parameters. This chapter presents an approximate modeling approach for evaluating the steady-state performance of EID systems with finite buffers and Least Recently Used replacement without requiring off-line simulations. As a first step towards such an approach, we show that EID systems with finite buffers can be modeled by discrete-time Markov chains (DTMCs). We derive customized DTMCs for the EID systems 7DS as described in [PS01] and PDI as described in Chapter 3.2. Unfortunately, the state space of the DTMC models grows exponentially in the number of nodes and in the number data items in the system. Thus, the model cannot be utilized for the numerical computation of steady state performance measures. Nevertheless, we present an approximative approach for modeling the steady state performance for EID systems that fulfill certain constraints. Using this approximative approach, a performance model for 7DS is derived and employed for a comprehensive performance study. Both the approximative modeling approach and the performance study of 7DS have been published in the proceedings of the *International Conference on Measurement & Modeling of Computer Systems* (*ACM SIGMETRICS 2005*) [LW05b].

5.1 A Discrete-Time Markov Chain Model

For deriving stochastic models of EID systems, consider a system that manages a set of distinct data items $\mathcal{D}$ with cardinality $|\mathcal{D}| = D$. The data items are matched by keys from a set $\mathcal{K}$ with cardinality $|\mathcal{K}| = K$. For ease of exposition, we identify both the data items and the keys with natural numbers, i.e., $\mathcal{D} = \{1, 2, \ldots, D\}$ and $\mathcal{K} = \{1, 2, \ldots, K\}$. In the remainder of this chapter, we use d with $0 \leq d \leq D$ and k with $0 \leq k \leq K$ to denote data items and keys, respectively. A node sends a query for a key k to retrieve a data item d matching key k. Similar to Section 3.3, we assume that the query stream follows the independent reference model. That is, each key k has access probability $\alpha(k)$ where $\sum_{k=1}^{K} \alpha(k) = 1$, and for a query q_i holds $P(q_i = k) = \alpha(k)$ regardless of the history of previous queries. Without loss of generality, we assume that $\alpha(k) > 0$ holds for all $k \in \mathcal{K}$, since a key k with $\alpha(k) = 0$ will never be requested and, thus, does not affect the state of the system. Since the goal of this chapter is to provide insight into the basic mechanisms that drive an EID system, modifications or expirations of data items are not considered. Thus, the query popularity distribution α is assumed to be constant over time.

Recall from Chapters 2 and 3 that EID systems use application-specific keys and data items with many-to-many matching. That is, each key k matches a set of data items $\mathcal{S}_k \subseteq \mathcal{D}$, and each data item d is matched by a set of keys $\mathcal{R}_d \subseteq \mathcal{K}$. However, it is easy to show that such many-to-many matching can be mapped on a one-to-one matching. Let $\gamma_k(d)$ denote the probability that a user searches for item d when sending a query for key k, with $0 < \gamma_k(d) \leq 1$ for $d \in \mathcal{S}_k$, $\gamma_k(d) = 0$ for $d \notin \mathcal{S}_k$, and $\sum_{d \in \mathcal{S}_k} \gamma_k(d) = 1$. Then, the overall probability $\alpha'(d)$ for the access to data item d can be computed by:

$$\alpha'(d) = \sum_{k \in \mathcal{R}_d} \alpha(k)\gamma_k(d) \tag{5.1}$$

Obviously, we can define a new set of keys $\mathcal{K}' = \{1, 2, \ldots, d\}$ with access probabilities $\alpha'(d)$ given by Equation 5.1. That is, it holds $|\mathcal{D}| = |\mathcal{K}|$, $\mathcal{S}_k = \{k\}$ for $k \in \mathcal{K}'$, and $\mathcal{R}_d = \{d\}$ for $d \in \mathcal{D}$. Thus, a one-to-one matching between keys $k \in \mathcal{K}'$ and data items $d \in \mathcal{D}$ can be defined. Without loss of generality we assume a one-to-one matching between keys and values in the remainder of this section and denote both keys and values with $k \in \mathcal{K}$.

As generic system model, consider a MANET with a set $\mathcal{N}$ of mobile nodes that participate in an EID system, $|\mathcal{N}| = N$. In the remainder of this chapter, we use n and m with $1 \leq n, m \leq N$ to denote mobile nodes. Each node sends queries to retrieve data items. Furthermore, each node contributes a buffer with finite capacity

of B data items and LRU replacement as described in Section 3.2 to the system. The MANET spans an area $\mathcal{A} = a \times a$ of size $A = a \cdot a$. We assume that the mobile nodes are initially distributed within $\mathcal{A}$ according to an arbitrary distribution. To incorporate node mobility, which is essential for epidemic dissemination of information, following Section 4.1 we consider a mobility model given by a continuous-time-continuous-state stochastic process $(X_n(t), Y_n(t))$ that describes the position of node n at time t. The mobility model interacts with the EID system by determining feasible communication operations together with the radio propagation model. For the radio propagation model, we assume that a message transmission between two nodes n and m is successful at time t with probability $f(d)$ for d given by the Euclidian distance, i.e.,

$$d = \sqrt{(X_n(t) - X_m(t))^2 + (Y_n(t) - Y_m(t))^2}. \tag{5.2}$$

For ease of exposition, we assume that all nodes have a homogenous transmission range R and no errors disturb a transmission. That is, we have $f(d) = 1$ for $0 \leq d \leq R$ and $f(d) = 0$ for $d > R$. However, we point out that all approaches presented in this chapter can be easily extended to consider more sophisticated transmission models.

For deriving a stochastic model of an EID system, we do not consider the mobility model in detail, but make two assumptions:

1. the mobility model leads to a uniform steady-state spatial distribution of the mobile nodes within the area $\mathcal{A}$ and

2. the mobile nodes change positions substantially between two successive queries.

That is, the node positions can be considered to be chosen independently from the area $\mathcal{A}$ by a uniform distribution before each query. On the first sight, these assumptions seem to be very restrictive towards the mobility model, since most mobility models do not exhibit these properties. However, we show in Section 5.1.3 that the results determined by the stochastic model closely match the results of a detailed simulation study, even if the mobility model employed in the simulation model violates the assumptions (1) and (2). Note that using the assumptions on the mobility and the radio propagation model and neglecting border effects, the probability that two nodes n and m can communicate at time t can be approximated by the probability that n and m are located in each others transmission range. It is easy to see that this happens if node m is placed within a circular area with radius R around node n, resulting in a probability of $\frac{\pi R^2}{A} =: \rho$.

We assume that each node sends queries with interarrival times determined by a Poisson process with rate λ. From a global perspective, the global query stream

observed in the EID system can be described by a sequence of queries $q_1, q_2, \ldots$. Due to the assumptions of Poisson processes for generating the query interarrival times and the independent reference model for selecting the queries, the query stream q_i has some important properties:

1. The global query interarrival times are determined by a Poisson process with rate $N\lambda$.

2. The node n_i that sends a query q_i is chosen from $\mathcal{N}$ according to a uniform distribution.

3. For each query q_i, a key k is chosen with probability $\alpha(k)$.

Consistent with Chapter 3, we refer to a node that issues a query q_i as the *inquiring node* for query q_i.

It is reasonable to assume that for a given EID system, the inquiring node for a query q_i first searches the local buffer for a data item matching the query and subsequently sends a Query message to a set of other nodes. In general, this set is determined by the EID system and constitutes a subset of the set of all reachable nodes. Recall that node reachability is determined by the node positions given by the mobility model and the radio propagation model. When receiving a Query message, a node that stores the matching data item in its buffer generates a Response message that contains the item. Like a Query message, a Response message is sent to a subset of all reachable nodes. Again, this subset is determined by the EID system. Nodes that receive the Response messages extract the data item and insert it into their local buffer. Inserting an item might cause a replacement of an other item by the LRU replacement scheme. Note that under the assumption that items are only stored in the buffers of the mobile nodes, it may happen that an item d is replaced at all nodes. Subsequently, the item will be unavailable for all time. To avoid such situations, we assume that each item d is initially stored by exactly one mobile node. Consistent with Chapter 3, this node is denoted as the *origin node* of item d. The selection of the origin node for item d depends on the EID system and the application scenario. A data item is available from its origin node at all time, even if it is not stored in the buffer of any other node.

Recall that each mobile node maintains a finite buffer of size B with LRU replacement. Furthermore, using the assumptions on the mobility model, we do not have to consider node positions and the state of the system at time t can be completely described by the content of the buffers of all nodes. Such state description constitutes a generalization of the state description of a stand-alone LRU caching

system [Cof73]. Note that we consider an EID system in steady state and can assume without loss of generality that all buffers are filled. Thus, at time $t \geq 0$ the state $S_n(t)$ of the buffer of node n is a permutation of B elements taken from $\mathcal{K}$. That is, $S_n(t) = [l_{n,1}, l_{n,2}, \ldots, l_{n,B}]$, with $l_{n,i} \in \mathcal{K}$ and $l_{n,i} \neq l_{n,j}$ for $i \neq j$. The state of the entire system $S(t)$ is given by $S(t) = [S_1(t), S_2(t), \ldots, S_N(t)]$. Note that the system has $\binom{K}{B}^N$ states.

With this model of the state space, a state transition $S \rightarrow S'$ is triggered by queries and the successor state S' depends on the communication operations that result from a query. Consistent with Section 4.1, we assume that communication events are timeless events. Thus, all Query and Response messages are exchanged immediately after a query, leading to a single state transition, and all state changes are triggered by queries. Given a particular EID system, the state transition probabilities of the system are determined by

1. the content of the buffers in the current state,
2. the access probabilities $\alpha(k)$ for key k, and
3. the feasibility of communication events depending on the current node positions.

The behavior of the system can be described by a continuous-time-discrete-state stochastic process $\mathbf{S}_t = \{S(t) | t \geq 0\}$. In fact, $\mathbf{S}_t$ constitutes a continuous-time Markov chain (CTMC). To verify the Markov property of $\mathbf{S}_t$, recall that as consequence of the independent reference model the access probability $\alpha(k)$ for a key k in state $S(t)$ is independent from the history of states $S(t')$ for $t' < t$. Furthermore, the node positions are also independent from the history of states due to assumption (2) on the mobility model.

Consider an infinite sequence of instances of time $t_1, t_2, \ldots$, where t_i denotes the instance of time of query q_i. It is easy to see that a discrete-time Markov chain (DTMC) $\mathbf{S}_i = \{S_i | i \in I\!N\}$ with $S_i = S(t_i)$ can be embedded into the CTMC $\mathbf{S}_t$. By construction, the state probability distribution of $\mathbf{S}_i$ and $\mathbf{S}_t$ is identical for $t = t_i$. Furthermore, the next state probabilities are identical for $\mathbf{S}_i$ and $\mathbf{S}_t$. In the remainder of this chapter, we will use *hit rate*, i.e., the fraction of queries for which a response is successfully received, as major performance measure. The hit rate is determined by both the probability that query q_i is successful in the current state $S_i = S$ and the state probability $P(S_i = S)$. Thus, for the computation of the hit rate it does not matter whether the CTMC $\mathbf{S}_t$ or the DTMC $\mathbf{S}_i$ is used. For lower computational

complexity and ease of exposition, we will use the DTMC $\mathbf{S}_i$ for deriving system properties and performance measures in the remainder of this chapter.

Let $p\,(S, S', k)$ denote the k-step transition probability of the DTMC $\mathbf{S}_i$ from state S to state S', and $\mathbf{P}$ with $\mathbf{P}_{ij} = p(S_i, S_j, 1)$ denote the transition matrix of $S(i)$. In the remainder of this section we prove the existence of a stationary probability distribution based on the assumptions made above. That is, there exists a vector π with dimension $S := \binom{K}{B}^N$, $0 \leq \pi(i) \leq 1$ for $1 \leq i \leq S$, and $\sum_{i=1}^{S} \pi(i) = 1$, for which holds $\pi = \pi\mathbf{P}$.

For ease of exposition throughout the proof, we assume that there is a dedicated sever node, which acts as origin node for all data items. Each data item can be retrieved from the server node with probability ρ on each query. The server node does not maintain a buffer itself. Thus, reception of a Query message by the server node does not necessarily change the state of the system.

Theorem 5.1.1 *The DTMC $\mathbf{S}_i$ has a unique stationary probability distribution π. That is, it exists a unique vector π with dimension S, $\sum_{i=1}^{S} \pi(i) = 1$, and $\pi = \pi\mathbf{P}$.*

Proof This proof constitutes a generalization of the proof of existence of a unique steady state probability distribution of a stand-alone LRU caching system [Cof73]. According to [Tri02], for a formal proof of the existence of a stationary probability distribution π it is sufficient to show that the DTMC $\mathbf{S}_i$ is finite, aperiodic, and irreducible. Recall that the DTMC is finite by definition. To proof that the DTMC is aperiodic and irreducible, it is sufficient to show that for two arbitrary states S and S' holds $p(S, S, 1) > 0$ and $p(S, S', t) > 0$ for an integer $t > 0$, respectively.

Assume that the DTMC is in state $S = [S_1, S_2, \ldots, S_N]$ with $S_n = [l_{n,1}, l_{n,2}, \ldots, l_{n,B}]$. The system stays in state S, e.g., if a node n issues a query for key $l_{n,1}$ and no other node $m \neq n$ is reachable. An exception constitutes the server node, which may be reached by the Query message and, subsequently, may generate a Response, since this does not change the state of the system. Recall that based on the assumption on the query process the probability that a query is send by node n is $\frac{1}{N}$. Furthermore, the probability that any node $n \neq m$ is reachable by node n is given by ρ due to the assumption on the mobility and radio propagation models. Thus, it holds

$$p\,(S, S, 1) \geq \frac{1}{N}\alpha(l_{n,1})\,(1-\rho)^{N-1} > 0, \tag{5.3}$$

that is, the DTMC $\mathbf{S}_i$ is aperiodic.

Furthermore, consider two states $S = [S_1, S_2, \ldots, S_N]$ and $S' = [S'_1, S'_2, \ldots, S'_N]$ with $S_n = [l_{n,1}, l_{n,2}, \ldots, l_{n,B}]$ and $S'_n = \left[l'_{n,1}, l'_{n,2}, \ldots, l'_{n,B}\right]$. One possible sequence of

queries that leads to a state transition from S to S' is the following sequence of NB requests: First, node 1 sends queries for keys $l'_{1,B}, l'_{1,B-1}, \ldots, l'_{1,1}$. Then, node 2 sends queries for keys $l'_{2,B}, l'_{2,B-1}, \ldots, l'_{2,1}$ and so on. At each query by node n, no other node $m \neq n$ but the sever node is reachable. Given this request sequence, it holds

$$p\left(S, S', NB\right) \geq \left(\rho\left(1-\rho\right)^{N-1}\right)^{NB} \prod_{n=1}^{N} \prod_{i=1}^{B} \alpha(l'_{n,i}) > 0, \tag{5.4}$$

that is, the DTMC $\mathbf{S}_i$ is irreducible. ∎

5.1.1 Customizing the Markov Model for 7DS

This section uses 7DS as a first example for the customization of the DTMC model presented in Section 5.1 for a particular EID system. Recall from Section 2.4.2 that each mobile node running 7DS maintains a local cache, i.e., a directory on the local disk, for storing Web pages. To keep terminology consistent, we denote the cache as buffer in the remainder of this Chapter. To query the buffer of remote nodes, 7DS uses Query and Response messages. A node that searches for a Web page sends a Query message to all reachable nodes. On a hit, a remote node returns a Response message to the inquiring mobile node. The inquiring node subsequently transfers the Web page and stores it in the local cache. Without loss of generality, consistent with [PS01] we assume that a Web page is transfered to the inquiring node if at least on mobile node sends a Response message. For the exemplary customization of the DTMC, we do not consider the advanced 7DS features power conservation, server-client mode and query repetition. However, the impact of these features on the steady state performance of 7DS is analyzed in Section 5.2 using an approximate approach for the steady state analysis.

The design of 7DS presented in [PS01] assumes a buffer of infinite size and leaves it to the user to remove Web pages that are out-of-date or no longer of interest in order to free disk space. However, for mobile nodes such as PDAs or even Smart Phones, storage capacity is scarce. Thus, we model and analyze 7DS with a finite buffer with a capacity of B web pages and LRU replacement. Note that web pages may have different sizes and, thus, a more efficient way to manage the buffer is to limit the total buffer space occupied by all web pages rather than their count. However, for a performance analysis of 7DS with finite buffers, we use replacement based on the number of Web pages stored in a buffer. The results presented in this chapter hold for other management approaches if the buffer size B is carefully chosen.

A straightforward way to incorporate LRU buffer management into 7DS is to update the LRU stack on each access to the local buffer, i.e., either on a local or

remote query. The least recently used Web page is replaced, if on a local request a Web page is retrieved from a remote node and buffer capacity is exceeded. Note that a Query message that is received by a node n results in an update of the LRU stack only if the item matching the key in the Query message is stored in the buffer of node n. Furthermore, the LRU stack of the inquiring node is only updated if either the item matching the key in the Query message is stored in the local buffer or if the item is received from a remote node.

More formally, consider a state transition $S \rightarrow S'$ with $S = [S_1, S_2, \ldots, S_N]$, $S' = [S'_1, S'_2, \ldots, S'_N]$, $S_n = [l_{n,1}, l_{n,2}, \ldots, l_{n,B}]$ and $S'_n = \left[l'_{n,1}, l'_{n,2}, \ldots, l'_{n,B}\right]$. To compute the one-step transition probability between these states based on the observation on LRU stack updates, we define the following subsets of the set of mobile nodes $\mathcal{N}$.

- $\mathcal{DH}^*(k, S, S') = \{n | S_n = S'_n = [l_{n,1} = k, l_{n,2}, \ldots, l_{n,B}]\}$

  The set $\mathcal{DH}^*(k, S, S') \subseteq \mathcal{N}$ consists of all nodes that store the data item matching key k at the top stack position in both states S and S'. Note that the LRU stack of these nodes is not updated when receiving a Query message for key k.

- $\mathcal{DH}(k, S, S') = \{n | S_n = [l_{n,1}, l_{n,2}, \ldots, l_{n,B}], l_{n,1} \neq k$
  $\wedge \exists j > 1 : l_{n,j} = k \wedge S'_n = [l_{n,j}, l_{n,1}, l_{n,2}, \ldots, l_{n,j-1}, l_{n,j+1}, \ldots, l_{n,B}]\}$

  The set $\mathcal{DH}(k, S, S') \subseteq \mathcal{N}$ consists of all nodes that store the data item matching key k beyond the top stack position in state S and perform an LRU update on the transition to state S' triggered by a Query message for key k. Note that these nodes must receive the Query message.

- $\overline{\mathcal{DH}}(k, S, S') = \{n | S_n = S'_n = [l_{n,1}, l_{n,2}, \ldots, l_{n,B}], l_{n,1} \neq k$
  $\wedge \exists j > 1 : l_{n,j} = k\}$

  The set $\overline{\mathcal{DH}}(k, S, S') \subseteq \mathcal{N}$ consists of all nodes that store the data item matching key k beyond the top stack in state S and do not perform an LRU stack update on the transition to state S' triggered by a Query message for key k. Note that these nodes must not be reached by the Query message.

- $\mathcal{NDH}(k, S, S') = \{S_n = S'_n = [l_{n,1}, l_{n,2}, \ldots, l_{n,B}], \forall j : l_{n,j} \neq k\}$

  The set $\mathcal{NDH}(k, S, S') \subseteq \mathcal{N}$ contains all nodes that do not store the item matching key k in both states S and S'. These node may be reached by a Query message, since this does not result in an update of the LRU stack.

Note that for each pair of states $S \neq S'$ and $k \in \mathcal{K}$ holds $\mathcal{DH}^*(k,S,S') \cap \mathcal{DH}(k,S,S') \cap \overline{\mathcal{DH}}(k,S,S') \cap \mathcal{NDH}(k,S,S') = \varnothing$ if and only if a state transition $S \rightarrow S'$ is feasible. For the computation of the probability for a state transition $S \rightarrow S'$ four cases must be considered:

1. $S = S'$, that is $\mathcal{N} = \mathcal{DH}^*(k,S,S') \cap \overline{\mathcal{DH}}(k,S,S') \cap \mathcal{NDH}(k,S,S')$ and $\mathcal{DH}(k,S,S') = \varnothing$ for all $k \in \mathcal{K}$

   The state of the system does not change if on a request for k the inquiring node n either

   (a) requests the item on the top of its local buffer (i.e., $n \in \mathcal{DH}^*(k,S,S')$) and no other node $m \in \overline{\mathcal{DH}}(k,S,S')$ receives the Query message *or*

   (b) requests an item that is not stored in its local buffer (i.e., $n \in \mathcal{NDH}(k,S,S')$) and receives it neither from the origin node nor from any other node $m \in \mathcal{DH}^*(k,S,S') \cup \overline{\mathcal{DH}}(k,S,S')$.

   The probability for this state transition can be computed by:

$$\begin{aligned} p(S,S',1) &= \sum_{k \in \mathcal{K}} \alpha(k) \left(\frac{|\mathcal{DH}^*(k,S,S')|}{N} (1-\rho)^{|\overline{\mathcal{DH}}(k,S,S')|} \right. \\ &\quad \left. + \frac{|\mathcal{NDH}(k,S,S')|}{N} (1-\rho)^{|\mathcal{DH}^*(k,S,S')| + |\overline{\mathcal{DH}}(k,S,S')| + 1} \right) \end{aligned} \tag{5.5}$$

2. $\mathcal{DH}^*(k,S,S') \cup \mathcal{DH}(k,S,S') \cup \overline{\mathcal{DH}}(k,S,S') \cup \mathcal{NDH}(k,S,S') = \mathcal{N}$ with $\mathcal{DH}(k,S,S') \neq \varnothing$ for $k \in \mathcal{K}$

   This state transition occurs if on a query for key k the inquiring node n either

   (a) requests the item on the top of its local buffer (i.e., $n \in \mathcal{DH}^*(k,S,S')$), all nodes $m \in \mathcal{DH}(k,S,S')$ receive the Query message, and all nodes $m \in \overline{\mathcal{DH}}(k,S,S')$ do not receive the Query message *or*

   (b) stores the requested item on any other stack position than the top (i.e., $n \in \mathcal{DH}(k,S,S')$), all nodes $m \in \mathcal{DH}(k,S,S')$ with $m \neq n$ receive the Query message, and all nodes $m \in \overline{\mathcal{DH}}(k,S,S')$ do not receive the Query message.

   The probability for this state transition can be computed by:

$$\begin{aligned} p(S,S',1) &= \alpha(k) \left(\frac{|\mathcal{DH}^*(k,S,S')|}{N} \rho^{|\mathcal{DH}(k,S,S')|} (1-\rho)^{|\overline{\mathcal{DH}}(k,S,S')|} \right. \\ &\quad \left. + \frac{|\mathcal{DH}(k,S,S')|}{N} \rho^{|\mathcal{DH}(k,S,S')|-1} (1-\rho)^{|\overline{\mathcal{DH}}(k,S,S')|} \right) \end{aligned} \tag{5.6}$$

3. $\mathcal{DH}^*(k,S,S') \cup \overline{\mathcal{DH}}(k,S,S') \cup \mathcal{NDH}(k,S,S') \cup \{n\} = \mathcal{N}$ with $\mathcal{DH}(k,S,S') = \varnothing$ for $n \in \mathcal{N}$ with $S_n = [l_{n,1}, l_{n,2}, \ldots, l_{n,B}]$ and $S'_n = [k, l_{n,1}, l_{n,2}, \ldots, l_{n,B-1}]$

   This state transition occurs if on a query for key k the inquiring node n does not store the matching data item but retrieves it either from the origin node or any other node $m \in \mathcal{DH}^*(k,S,S')$. Note that either the origin node or any node $m \in \mathcal{DH}^*(k,S,S')$ must receive the QUERY message. Furthermore, all nodes $m \in \overline{\mathcal{DH}}(k,S,S')$ must not receive the QUERY message. The probability for this state transition can be computed by:

$$\begin{aligned} p(S,S',1) &= \alpha(k)\frac{1}{N}\left(1-(1-\rho)^{|\mathcal{DH}^*(k,S,S')|+1}\right) \\ &\quad \cdot(1-\rho)^{|\overline{\mathcal{DH}}(k,S,S')|} \end{aligned} \tag{5.7}$$

4. $\mathcal{DH}^*(k,S,S') \cup \mathcal{DH}(k,S,S') \cup \overline{\mathcal{DH}}(k,S,S') \cup \mathcal{NDH}(k,S,S') \cup \{n\} = \mathcal{N}$ with $\mathcal{DH}(k,S,S') \neq \varnothing$ for $n \in \mathcal{N}$ with $S_n = [l_{n,1}, l_{n,2}, \ldots, l_{n,B}]$ and $S'_n = [k, l_{n,1}, l_{n,2}, \ldots, l_{n,B-1}]$

   This state transition occurs if on a query for key k the inquiring node n does not store the matching data item but retrieves it either from the origin node or any other node $m \in \mathcal{DH}^*(k,S,S') \cup \mathcal{DH}(k,S,S')$. Note that all nodes $m \in \mathcal{DH}(k,S,S')$ must receive the QUERY message, whereas the origin node and the nodes $m \in \mathcal{DH}^*(k,S,S')$ may receive the QUERY message. Furthermore, all nodes $m \in \overline{\mathcal{DH}}(k,S,S')$ must not receive the QUERY message. The probability for this state transition can be computed by:

$$p(S,S',1) = \alpha(k)\frac{1}{N}\rho^{|\mathcal{DH}(k,S,S')|}(1-\rho)^{|\overline{\mathcal{DH}}(k,S,S')|} \tag{5.8}$$

For all other state transitions $S \rightarrow S'$ holds $P(S,S',1) = 0$. To estimate the number of possible state transitions from a state $S = [S_1, S_2, \ldots, S_N]$, let $DH(k,S)$ denote the number of nodes n with $S_n = [l_{n,1}, l_{n,2}, \ldots, l_{n,B}]$, $l_{n,1} \neq k$, and $l_{n,j} = k$ for a given j with $1 < j \leq B$. Obviously, it holds $0 \leq DH(k,S) \leq N$, since all nodes may store an item on any position beyond the top of the stack. For all feasible state transitions that result from a query for key k, such nodes may either

- be located *inside* the transmission range of the inquiring node and update the LRU stack *or*
- be located *outside* the transmission range of the inquiring node and does not update the LRU stack.

Thus, there are about $2^{DH(k,S)}$ possible successor states on a query for key k in state S. That is, the number of successor states grows exponentially in the number of nodes. Note that this implies that there is no compact representation of the transition matrix $\mathbf{P}$ of the DTMC, e.g., by using data structures for sparse matrixes. That is, the DTMC model cannot be used for computational efficient determination of the steady state probability distribution π by numerical solution of $\pi = \pi\mathbf{P}$. Furthermore, it is not clear if the transition matrix $\mathbf{P}$ has any structured representation that allows a computational efficient solution. Motivated by these observations, we present an approximate steady-state analysis approach in Section 5.2.

Although it is not computationally efficient to explicitly compute all next state probabilities, an implicit approach can be used to determine the next state S' for a given state S with the appropriate probability $p(S, S', 1)$. This implicit approach is shown in Algorithm 5.1. In the algorithm, $Random(N)$ returns an integer n with $1 \leq n \leq N$ that is chosen according to a uniform distribution. $Key()$ returns k with probability $\alpha(k)$ for $1 \leq k \leq K$. For $0 \leq p < 1$, $Coin(p)$ returns *true* with probability p and *false* with probability $1 - p$. $Hit(n, k)$ returns *true* for n with $S_n = [l_{n,1}, l_{n,2}, \ldots, l_{n,B}]$ and $l_{n,j} = k$, for a given j with $1 \leq j \leq B$, and *false* otherwise. For n with $Hit(n, k) = true$, $Update(n, k)$ changes the state to $S'_n = [k, l_{n,1}, l_{n,2}, \ldots, l_{n,j-1}, l_{n,j+1}, \ldots, l_{n,B}]$. The implicit approach first selects the inquiring node and the key in the query (Lines 1–2). After that, the first `if` statement (Lines 4–5) determines whether the origin node is reached by the Query message. Within the `for` statement (Lines 6–9) it is checked whether any remote node that stores the item matching key k is reached by the Query message. For such nodes, the LRU

Algorithm 5.1 *NextState7ds*(S)

```
 1 n ← Random(N)
 2 k ← Key()
 3 Response ← false
 4 if Coin(ρ) then
 5    Response ← true
 6 for 1 ≤ m ≤ N with m ≠ n do
 7    if Hit(m, k) and Coin(ρ) then
 8       Update(m, k)
 9       Response ← true
10 if Response = true or Hit(n, k) then
11    Update(n, k)
```

stack is updated. Last, the second `if` statement (Lines 10–11) updates the LRU stack of the inquiring node in case of a local or remote hit. It is easy to verify that the implicit transition probabilities to the next states generated by Algorithm 5.1 are identical to the explicit transition probabilities computed above. We validate the implicit approach in Section 5.1.3.

5.1.2 Customizing the Markov Model for PDI

Recall from Chapter 3 that the major difference between PDI and 7DS constitutes the fact that with PDI a Response message is not only sent to the inquiring node, but to all nodes inside the transmission range of the responding node or even to more nodes if selective forwarding is enabled. Thus, a query for key k does not only update the LRU stacks of nodes that store the item matching key k in the local buffers, but also inserts this item into the buffers of other nodes than the inquiring node. It is easy to see that this functional difference increases the number of successor states for a state S. Thus, it is again computational efficient to explicitly compute the transition matrix $\mathbf{P}$ for the numerical evaluation of the DTMC for PDI. For this reason we do not derive the next state probabilities explicitely, but only present an implicit approach for determining the next state S' given a state S. This approach is shown in Algorithm 5.2. Beyond the definitions used in Algorithm 5.1, Algorithm 5.2 uses $Insert(n, K)$ to alter the state of node n from $S_n = [l_{n,1}, l_{n,2}, \ldots, l_{n,B}]$ with $l_{n,j} \neq k$ for $1 \leq j \leq B$ to $S'_n = [k, l_{n,1}, l_{n,2}, \ldots, l_{n,B-1}]$. Furthermore, P_{Query} denotes the probability that a node receives a Query message, $P_{Response}(r)$ denotes the probability that a node receives at least one Response message from r responding nodes and $P^*_{Response}(r)$ denotes the probability that a node receives at least one Response message from r responding nodes without receiving the Query message. Similar to Algorithm 5.1, Algorithm 5.2 determines the inquiring node n and the key to query k (Lines 1–2), determines if the origin node is reached by the Query (Lines 6–7), determines whether a Response message is generated by remote nodes (Lines 8–16), and updates the LRU stack of the inquiring node on a remote hit (Lines 17–18). While checking if remote nodes generate a Response message, those nodes that do not generate a Response message are classified into the set of data holders that do not receive the Query message $\mathcal{DH}_{NR}$ (Line 14) and the set of nodes that are not data holders $\mathcal{NDH}$ (Line 16). The LRU stack of nodes $n \in \mathcal{DH}_{NR}$ is updated, if the node is reached by at least one Response message generated by one of the nodes that receive the Query message (Lines 19–21). Furthermore, nodes $n \in \mathcal{NDH}$ also update the LRU stack, if they receive a Response message (Lines

22–24). Note that a node $n \in \mathcal{DH}_{NR}$ has a different update probability than a node $m \in \mathcal{NDH}$, since n must be located outside the transmission range of the enquiring node while m maybe located inside the transmission range.

The probabilities P_{Query}, $P_{Response}$, and $P^*_{Response}$ depend on the current node positions as well as on the configuration of the parameter TTL_{Query} that controls selective forwarding as described in Section 3.2.2. Depending on the choice of TTL_{Query}, the computation of these probabilities may be arbitrary complex. However, if selective forwarding is disabled, i.e., $TTL_{Query} = 1$, the probabilities can be approximated as follows. Without selective forwarding, the probability that a node receives a Query message is given by $P_{Query} = \rho$ similar to the 7DS case. Obviously, on a Query

Algorithm 5.2 $NextStatePdi(S)$

```
1  n ← Random(N)
2  k ← Key()
3  Response ← 0
4  DH_NR ← ∅
5  NDH ← ∅
6  if Coin(P_Query) then
7     Response ← Response + 1
8  for 1 ≤ m ≤ N with m ≠ n do
9     if Hit(m, k) then
10       if Coin(P_Query) then
11          Update(m, k)
12          Response ← Response + 1
13       else
14          DH_NR ← DH_NR ∪ {m}
15    else
16       NDH ← DH ∪ {m}
17 if Response > 0 or Hit(n, k) then
18    Update(n, k)
19 for m ∈ DH_NR do
20    if Coin(P*_Response(n)) then
21       Update(m, k)
22 for m ∈ NDH do
23    if Coin(P_Response(n)) then
24       Insert(m, k)
```

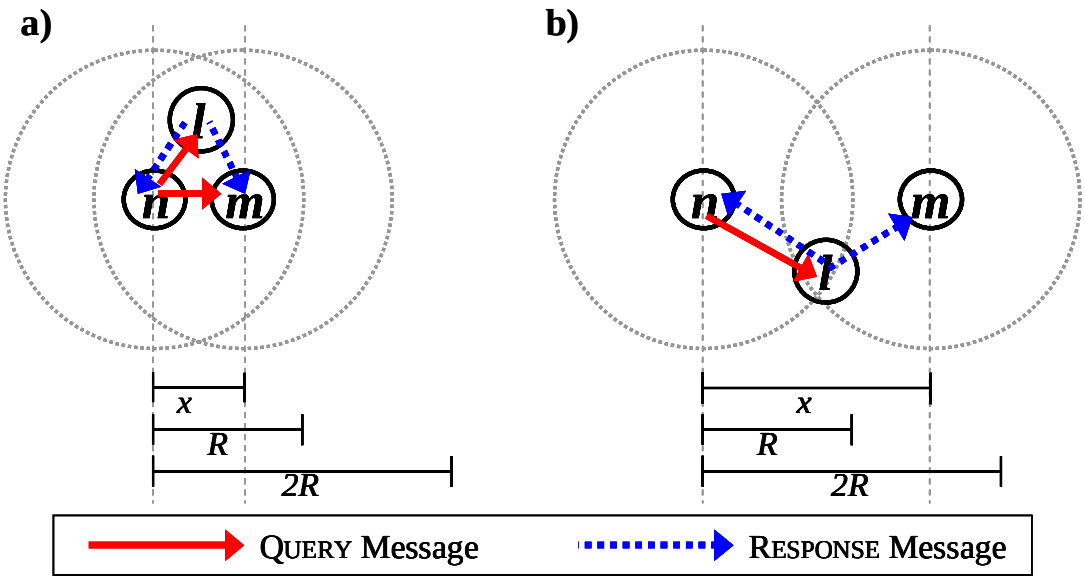

Figure 5.1. Illustration of situations for receiving a Response message

message for key k sent by node n, a node m receives a Response message if at least one node l that stores the item matching key k is located in the area that is covered by the transmission ranges of both nodes n and m. This is illustrated by Figure 5.1, where for the distance x between nodes n and m holds $0 \leq x \leq R$ in Figure 5.1 (a), and $R < x \leq 2R$ in Figure 5.1 (b). Note that node m receives both the Query and the Response message in Figure 5.1 (a), whereas it receives only the Response message in Figure 5.1 (b). Under the assumption that node positions are chosen uniformly from the area $\mathcal{A}$ and neglecting border effects, for the CDF $P(x)$ of the distance x between nodes n and m holds

$$P(x) = \rho = \frac{\pi x^2}{A}, \tag{5.9}$$

since the probability for a distance of at most x is equal to the probability for placing node m in a circular area with radius x around node n. Thus, the pdf $p(x)$ of the distance x is given by

$$p(x) = \frac{P(x)}{dx} = \frac{2\pi x}{A}. \tag{5.10}$$

Given a distance d between nodes n and m, it is easy to see that for the width w of the area covered by the transmission ranges of both nodes holds

$$x = 2R - w, \tag{5.11}$$

that is, $w = 2R - x$, and $R - w/2 = x/2$. The size of the covered area can be computed by integrating the function $f(x) = \sqrt{R^2 - x^2}$ that describes a semicircle

with radius R for $x/2 \leq t \leq R$ and multiplying the result by 4. Thus, we have

$$\begin{aligned} A(x) &= 4 \cdot \int_{x/2}^{R} \sqrt{R^2 - t^2} dt \\ &= \pi R^2 - \tfrac{x}{2}\sqrt{4R^2 - x^2} - 2R^2 \arcsin(\tfrac{x}{2R}) \end{aligned} \tag{5.12}$$

Using Equations 5.10 and 5.12, the probability $P_{Response}(r)$ can be computed by integrating the probability of placing nodes n and m within a distance of x with at least one node out of r data holders in the covered area for $0 \leq x \leq 2R$:

$$P_{Response}(r) = \int_0^{2R} p(x) \left(1 - \left(1 - \frac{A(x)}{A}\right)^r\right) dx \tag{5.13}$$

For computing $P^*_{Response}$, recall that a node m that does not receive a Query message must be located outside the transmission range of the inquiring node n as shown in Figure 5.1 (b). That is, node m can only be placed in the remaining area of size $A - \pi R^2$, changing Equations 5.9 and 5.10:

$$P'(x) = \frac{\pi x^2}{A - \pi R^2} \tag{5.14}$$

$$p'(x) = \frac{2\pi x}{A - \pi R^2} \tag{5.15}$$

Using these equations, the probability $P^*_{Response}(r)$ can be computed by replacing the expressions from Equation 5.10 in Equation 5.13 and integrating for $R < x \leq 2R$:

$$P^*_{Response}(r) = \int_R^{2R} p'(x) \left(1 - \left(1 - \frac{A(x)}{A}\right)^r\right) dx \tag{5.16}$$

Note that computing the probabilities $P_{Response}$ and $P^*_{Response}$ as shown above assumes that the node positions are chosen independently on each evaluation of Lines 20 and 23 of Algorithm 5.2, respectively. This is not conform to the assumptions on the mobility model that assumes fixed node positions for each query. However, we show in Section 5.1.3 that the approximation from Equations 5.13 and 5.16 provides reasonable approximations for the evaluation of the DTMC for PDI.

5.1.3 Validation of the Markov Models

Modeling Assumptions

Note that it is not clear how well the performance of 7DS and PDI is modeled by the DTMC models due to the assumptions on the mobility model and on node positions. Thus, in this section we validate the accuracy of the DTMC models for 7DS and PDI, respectively, against the results of a detailed simulator that considers

node mobility and message transmission. To perform the validation for a scenario that is realistic as well as relevant, we use assumptions identical to [PS01] as far as possible. Consistent with [PS01], we assume that N mobile nodes move in a square area of $1000m \times 1000m$. All mobile nodes have an identical transmission range R. Unfortunately, [PS01] presents a study of the transient behavior of 7DS considering the spread of a single data item among mobile nodes with buffers of infinite size. Thus, to evaluate the performance of EID systems with finite buffers and LRU replacement, we have to extend the system model used in [PS01].

In contrast to [PS01], we assume that $K = 1000$ popular data items are distributed in the buffers of the mobile nodes. [PS01] presents a performance study of a web browsing application using 7DS, that is, keys in 7DS constitute URLs that identify data items given by web documents. Thus, we have a one-to-one matching between keys and data items. [PS01] uses numbers of mobile nodes $N \in 5, 10, \cdots, 25$ to represent different levels of popularity for the single data item considered in each experiment. We assume a fixed number of mobile nodes, $N = 64$, in most experiments and represent the popularity of the data items by setting the access probabilities $\alpha(k) \cong k^{-\gamma}$ for $1 \leq k \leq K$. This is, similar to the P2P, IM, and DWS application presented in Section 3.3, we assume that the access probabilities follow a Zipf-like distribution. [CLP04] analyzes the object popularity distribution in a mobile web browsing application and reports a Zipf-like distribution with values of γ between 0.85 and 1. Thus, we use $\gamma = 0.9$ in most of our experiments. [CLP04] also reports that mobile nodes repeat a significant fraction of requests. Thus, the independent reference model constitutes a realistic assumption for a mobile web browsing application.

To validate the DTMC models, we implemented a discrete-event simulator based on these models, denoted as *DTMC simulator*. Furthermore, we implemented detailed simulators of the 7DS and PDI systems. In contrast to the DTMC simulator, the detailed simulators consider node mobility by implementing different mobility models. Query interarrival times for each node are chosen according to a Poisson process. Furthermore, on each query they model the exact exchange of Query and Response messages based on the current node positions. It is easy to see that the DTMC simulator is much more efficient than the detailed simulator, since it requires significantly fewer decisions based on stochastic events. Table 5.1 summarizes the default values for all model parameters. These values are used in all performance experiments if not stated otherwise.

To avoid inaccuracies due to border effects when mobile nodes are located close to edges of the simulation area, we use the toroidal distance model as proposed

in [Bet02]. That is, the flat square-like area becomes a torus, so that mobile nodes located at one border of the simulation area can contact mobile nodes at the opposite border. The mobile nodes move according to one of the following three mobility models:

1. *Random placement model* (*RP*): New positions of the nodes are chosen from the simulation area $\mathcal{A}$ by a uniform distribution between two successive queries.

2. *Random waypoint mobility model* (*RWP*): Using the approach for perfect simulation of the RWP mobility model, the nodes move with a maximum speed $v \in (0, v_{max}]$ and rest for a maximum time T_{pause} at their destination points as described in Section 4.2.

3. *Reference point group mobility* (*RPG*, [HGPC99]): The nodes move in G groups with speed v_g, radius r_g, and a displacement radius r_n as described in Section 3.4.

Note that the RP mobility model is not realistic, but matches closely the assumptions of the stochastic modeling approach. The RWP mobility model is quite realistic for mimicking the movement of individual pedestrians. However, it has been shown that the RWP model leads to a non-uniform distribution of mobile nodes while they are moving [BRS03], [Bou04]. Thus, it violates a key assumption of the modeling approach. Additionally, the RPG mobility model violates the assumption of a substantially change of the vicinity of a mobile node between two successive queries.

Since we consider a EID system in operation, the mobile nodes start with full buffers in the initial state of the simulation. The initial distribution of the items in the buffers depends heavily on the application scenario. Thus, beside initially empty buffers, we consider a worst case and a best case initial distribution in the simulation experiments:

Parameter	**Value**
Number of mobile nodes N	64
Transmission range R	$115m$
Simulation area A	$1000m \times 1000m$
Query rate λ	$1/60$
Zipf-parameter of query locality γ	0.9
Number of keys K	1000
Buffer size B	256

Table 5.1. Default values for model parameters

1. each buffer is filled with B data items chosen from $\mathcal{K}$ according to a uniform distribution. That is, each item is equally likely to be contained in a buffer, which constitutes a worst-case scenario since the popularity of the data items is not considered.

2. each buffer is filled with B data items chosen from $\mathcal{K}$ according to a Zipf-like distribution with parameter γ similar to the parameter of the request probabilities $\alpha(k)$. This constitutes a best-case scenario, since the popularity of the data items is reflected in the initial content of each buffer.

To determine the performance of the EID systems with all simulation models, we count the number of responses received for each query and calculate the average hit rate. Note that the hit rate can be computed by counting the number of queries that triggers an update of the buffer of the inquiring node in Line 11 of Algorithm 5.1 and Line 18 of Algorithm 5.2, respectively. To cut off the initial transient phase, we perform a warmup phase of 30,000 queries. After that, we conduct 30 batches each comprising of 10,000 queries and compute the 99% confidence interval for the hit rate using the batch means.

Transient Behavior

Before validating the Markov models of PDI and 7DS, we use the detailed simulator of 7DS to provide evidence that a steady state analysis is important for evaluating the performance of EID systems with finite buffers. For this purpose, we measure the duration of the transient phase for a 7DS system starting with full buffers. The initial buffer content is chosen according to both the worst case and the best case model described above. Figure 5.2 plots the transient phase of the average fraction of dataholders, i.e., the fraction of nodes that store a particular data item, for the data items with rank 1, 50, and 100, respectively. Note that with the Zipf-like access probabilities, the popularity rank of the item matching key k is k. The average fraction of dataholders for each item is calculated in intervals of 60 seconds. For all data items, the steady state fraction of dataholders is shown as a dashed horizontal line. For the experiments, we use the RWP mobility model with parameters $v_{max} = 2m/s$ and $T_{hold} = 30s$. Figure 5.2 shows that even in the worst case scenario with a uniform distribution of buffer content, the fraction of dataholders reaches the steady state for a popular data item in less than one hour. For less popular data items, the fraction of dataholders is close to the steady state values in less than 2 hours. If the initial buffer content is chosen according to a Zipf-like distribution, the fraction of dataholders is close to the steady state fraction almost immediately. Note that due

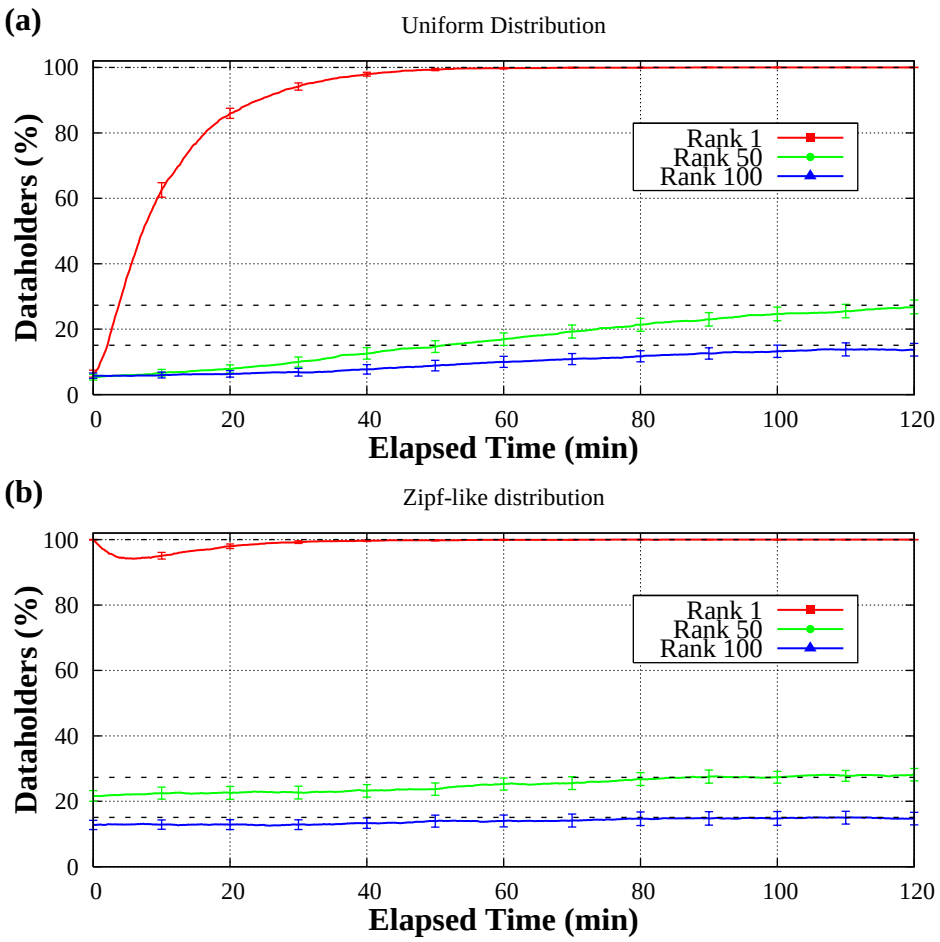

Figure 5.2. Transient phases of 7DS with initial distribution of buffer content drawn from (a) a uniform distribution and (b) a Zipf-like distribution

to the Zipf-like distribution of access probabilities, the hit rate of 7DS is determined by the number of hits to popular data items. Thus, the hit rate is close to the steady state in below one hour. We conclude from Figure 5.2 that a steady state analysis constitutes an important tool for the performance evaluation of EID systems with finite buffers, since a 7DS system with finite buffer capacity reaches steady state in a short amount of time.

Validation of the 7DS Model

In the next set of experiments, we validate the DTMC simulator of 7DS against the results of the detailed simulator. For the detailed simulator, we use the RP mobility model as well as the RWP mobility model with three different parameter settings. Table 5.2 compares the results for both simulators and different mobility models for

various buffer sizes. The table includes the 99% confidence intervals for the simulation results. Table 5.2 shows that the steady state hit rates for all mobility models are almost identical for a given buffer size. The results of the DTMC simulator almost always lie within the 99% confidence intervals of the detailed simulation results except for small buffer sizes. However, the maximum difference between the mean values is below 3%, even for the more realistic simulations based on the RWP mobility model. We conclude from Table 5.2 that the steady state hit rate of the 7DS system is almost independent of the mobility model and is accurately predicted by the DTMC simulator.

To gain further insight into the impact of the mobility model, we validate the DTMC simulator against a detailed simulation study that uses the RPG mobility model. Figure 5.3 plots the hit rate as a function of the number of groups G for $v_{max} = 2m/s$, $r_g = 200m$, and $r_n = 50m$. The nodes are distributed to the groups in a round robin fashion. The server node is also assigned to one of the groups. Recall that the server node is the origin node for all data items. Hence, in scenarios with few groups, hit rate is higher than predicted by the DTMC simulator, since almost all nodes in the group of the server node receive a Response message for each query. With an increasing number of groups, the probability for reaching the server node converges against ρ and the DTMC simulator closely matches the simulation results. We conclude from Figure 5.3 that the DTMC simulator can be applied even in scenarios with many small groups of mobile nodes.

We have performed various validation experiments for the DTMC simulator of 7DS that consider different system sizes, different transmission ranges and different degrees of query locality. All experiments showed an excellent agreement between the results of the DTMC simulator and the detailed simulator.

Buffer Size	Markov Model		Random Placement		Random Waypoint $v_{max} = 2m/s$ $T_{hold} = 30s$		Random Waypoint $v_{max} = 32m/s$ $T_{hold} = 30s$		Random Waypoint $v_{max} = 2m/s$ $T_{hold} = 1200s$	
	Mean	*Conf. Int.*	*Mean*	*Conf. Int.*	*Mean*	*Conf. Int.*	*Mean*	*Conf. Int.*	*Mean*	*Conf. Int.*
32	0.511	[0.509, 0.514]	0.515	[0.512, 0.519]	0.523	[0.518, 0.528]	0.524	[0.520, 0.528]	0.516	[0.511, 0.522]
64	0.606	[0.604, 0.608]	0.608	[0.605, 0.611]	0.616	[0.607, 0.624]	0.618	[0.615, 0.621]	0.610	[0.606, 0.613]
96	0.770	[0.668, 0.671]	0.669	[0.666, 0.673]	0.681	[0.677, 0.686]	0.677	[0.672, 0.681]	0.671	[0.667, 0.675]
128	0.712	[0.710, 0.714]	0.707	[0.704, 0.710]	0.721	[0.717, 0.725]	0.722	[0.719, 0.725]	0.720	[0.716, 0.725]
160	0.748	[0.745, 0.751]	0.745	[0.741, 0.749]	0.755	[0.751, 0.758]	0.757	[0.753, 0.760]	0.750	[0.747, 0.754]
192	0.778	[0.775, 0.780]	0.775	[0.773, 0.778]	0.794	[0.790, 0.798]	0.786	[0.782, 0.790]	0.781	[0.777, 0.785]
224	0.803	[0.800, 0.806]	0.800	[0.797, 0.802]	0.817	[0.814, 0.820]	0.806	[0.802, 0.810]	0.807	[0.804, 0.810]
256	0.826	[0.823, 0.829]	0.821	[0.818, 0.824]	0.828	[0.825, 0.832]	0.829	[0.826, 0.832]	0.827	[0.823, 0.830]

Table 5.2. Validation of the DTMC model for 7DS

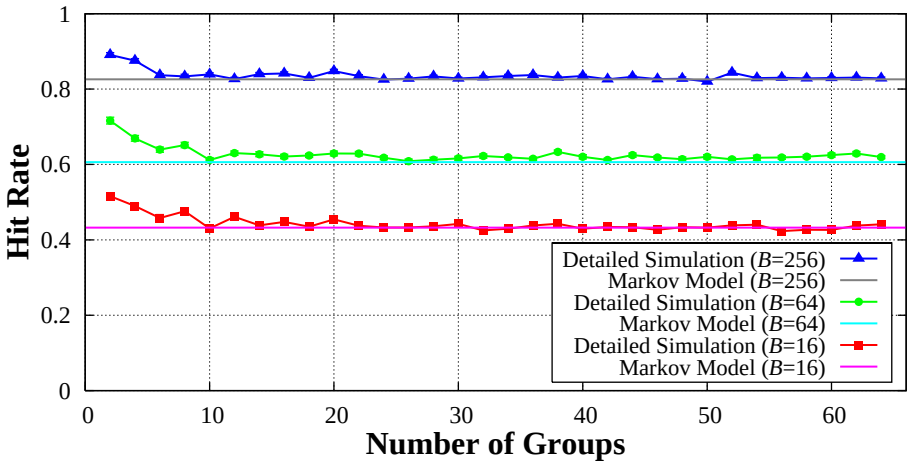

Figure 5.3. Validation of the DTMC model for different degrees of group mobility

Validation of the PDI Model

In a last set of experiments, we validate the DTMC simulator for PDI against the detailed simulator. Figure 5.4 plots the hit rate for the DTMC simulator and the detailed simulator, respectively, for different numbers of nodes N, transmission ranges R and degrees of query locality γ, respectively, as well as different buffer sizes B. Recall that the DTMC simulator for PDI does not consider exact node positions, but uses an average value for the number of node that can be reached by both the inquiring node and the node that receives a response message. However, the validation experiments show an excellent agreement between the results of the DTMC simulator and the detailed simulator.

5.2 An Approximate Steady State Analysis

Recall that for both 7DS and PDI the DTMC models do not allow the efficient numerical computation of the stationary probability distribution. Nevertheless, the efficient approximate computation of steady-state hit rate is possible for a certain class of EID systems. This section presents an approximate approach for computing the steady-state hit rate and applies the approach to 7DS.

5.2.1 Approximate Analysis of LRU Buffer Hit Rate

Obviously, the buffer hit rate is the key quantity for the long-term performance of an EID system. Thus, performance modeling of EID systems requires the derivation of

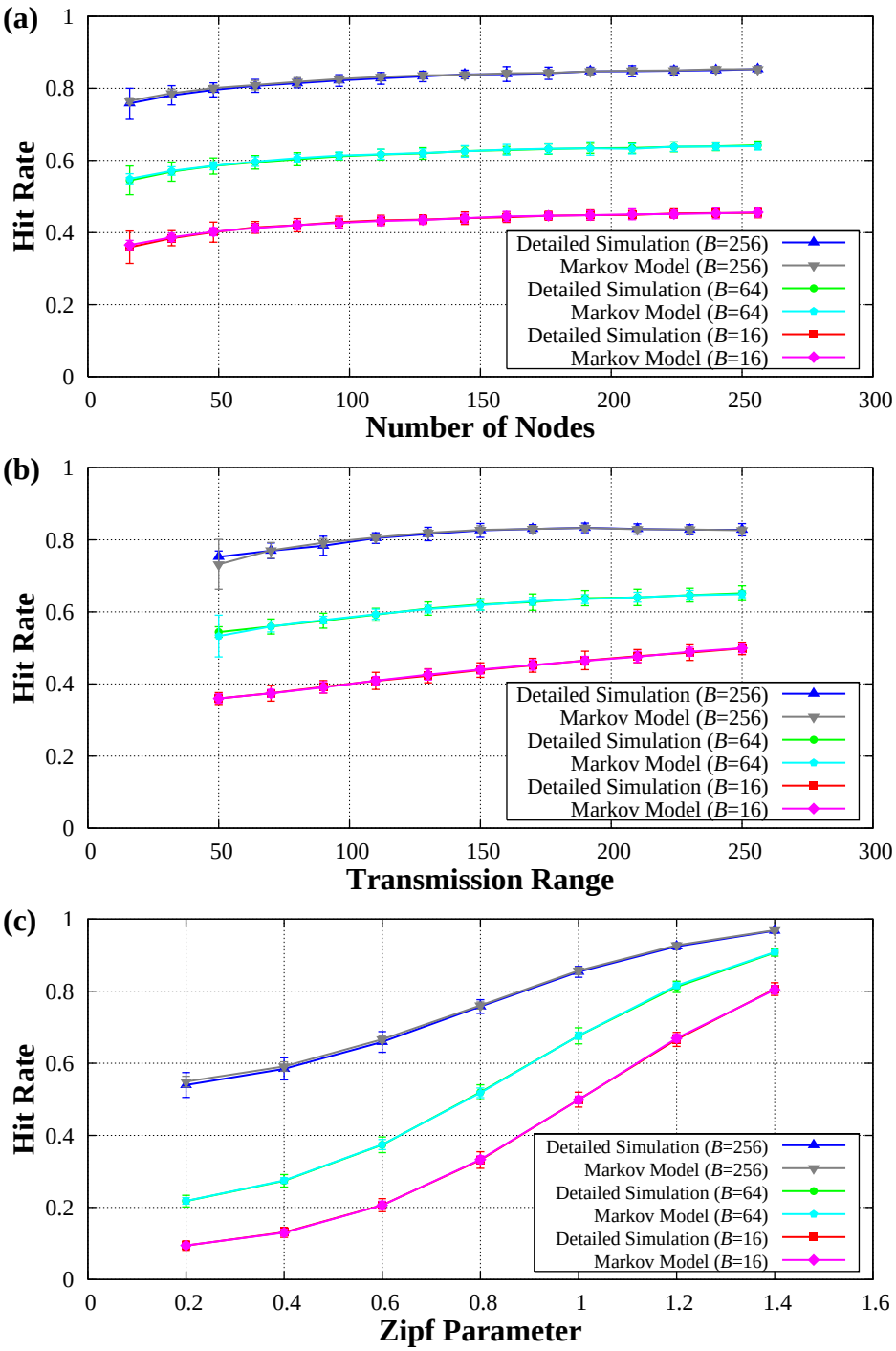

Figure 5.4. Validation of the Markov Model for PDI for (a) different system sizes, (b) different transmission ranges, and (c) different degrees of query locality

LRU buffer hit rate in steady-state. We recall a computationally efficient approximate approach for determining the steady state buffer hit rate for LRU buffers introduced by Dan and Towsley [DT90]. The approach cannot only be employed for stand-alone caches, but has also been applied for distributed data-sharing environments in wired networks [DDY90].

Consider an application that retrieves data items by keys using a buffer with LRU replacement. The assumptions on the data items and keys are similar to the assumptions for the DTMC model presented in Section 5.1. The approach presented in [DT90] does not require a one-to-one matching of keys and values. In fact, for increasing the computational efficency, keys $k, l \in \mathcal{K}$ with the same access probability $\alpha(k) = \alpha(l)$ are combined to a single key k'. Thus, without loss of generality we can assume a one-to-many matching between keys and data items, i.e., $\mathcal{S}_k \geq 1$ and $\mathcal{R}_d = 1$. To keep notation consistent with [DDY90], we define the fraction $\beta(k)$ of data items matched by key k as $\beta(k) := \frac{|\mathcal{S}_k|}{|\mathcal{D}|}$. Thus, the set $\mathcal{D}$ can be divided into K partitions $D(k)$, each of size $\beta(k)D$ with $\sum_{k=1}^{K} \beta(k) = 1$. Recall that consistent with Section 5.1 queries are issued according to a Poisson process with rate λ and follow the independent reference model with access probabilities $\alpha(k)$.

As key concept, the approach of Dan and Towsley approximately determines the expected number of items of partition $D(k)$ in the top j positions of the LRU stack, denoted by $b(k, j)$, rather than determining the exact content of the buffer. When a query is issued for key k, the hit probability in the top j stack positions is given by $\frac{b(k,j)}{\beta(k)D}$. Let $r(k, j)$ denote the rate for pushing down items of partition $D(k)$ from stack position j to stack position $j + 1$ where $1 \leq k \leq K$ and $1 \leq j \leq B$. [DT90] argues that under steady-state-conditions the long-term rate for pushing down an item from position j to stack position $j+1$ is equal to the rate for inserting the item into the top j stack positions. Thus, with the assumption that on a miss an item can always be inserted into the buffer, the rate $r(k, j)$ is equal to the miss rate for key k in the top j stack positions. This miss rate is given by the product of the query rate λ, the access probability for key k, and the miss probability for key k. Thus, we have:

$$r(k, j) = \lambda\alpha(k)\left(1 - \frac{b(k, j)}{\beta(k)D}\right) \tag{5.17}$$

This observation is denoted as *conservation of flow* in [DDY90].

Let $p(k, j)$ denote the probability that an item of partition $D(k)$ is located at position j in the LRU stack with $1 \leq k \leq K$ and $1 \leq j \leq B$. As shown in [DT90], the probability $p(k, j)$ can be derived by exploiting the fact that if an item is pushed down from stack position j in the LRU scheme, then it moves to position $j+1$. Furthermore,

if no item is pushed down to stack position j, the item on this position obviously remains unchanged. Hence, the probability $p(k,j)$ is given by the probability for pushing down an item from position $j-1$ to position j under the condition that the current query triggers such a buffer movement from position $j-1$ to j in the LRU stack. The probability $p(k,j)$ can be approximated closely by the ratio between the rate for a buffer movement of an item of $D(k)$ from position $j-1$ to j and the rate for such a buffer movement of any item from $\mathcal{D}$. Thus, using Equation 5.17 we have:

$$\begin{aligned} p(k,j) &\approx \frac{r(k,j-1)}{\sum_{n=1}^{K} r(n,j-1)} \\ &= \frac{\alpha(k)\left(1-\frac{b(k,j-1)}{\beta(k)D}\right)}{\sum_{n=1}^{K}\alpha(n)\left(1-\frac{b(n,j-1)}{\beta(n)D}\right)} \end{aligned} \tag{5.18}$$

This observation is denoted as *relative push down rate* in [DDY90]. Note that the probability $p(k,j)$ is independent of the query rate λ.

Subsequently, for each partition $D(k)$, the expected number of items of $D(k)$ in the top j positions, $b(k,j)$, can be determined by the summation of the probabilities for finding an item of $D(k)$ at a position less or equal to j in the buffer. Thus, we have:

$$b(k,j) = \sum_{n=1}^{j} p(k,n), k = 1,\ldots,K, j = 1,\ldots,B \tag{5.19}$$

Combining Equation 5.18 and Equation 5.19 together with $b(k,1) = \alpha(k)$ for $1 \leq k \leq K$ leads to an iterative scheme for the approximate computation of $b(k,j)$ for $k = 1,\ldots,K$, and $j = 2,\ldots,B$. This iterative scheme has the computational complexity $O(KB)$. As noted in [DT90], due to the approximative computation, values of $b(k,j)$ may exceed $\beta(k)D$ for some k and j. In this case, all $r(k,n)$ are set to 0 for $j < n \leq B$. That is, it is assumed that all items of partition k are located in the top j locations of the buffer and are never pushed down beyond position j. Having computed the quantities $b(k,B)$ using Equation 5.19, the hit rate of an LRU buffer is given by:

$$HR_{LRU} \approx \sum_{k=1}^{K} \frac{\alpha(k)b(k,B)}{\beta(k)D} \tag{5.20}$$

In the next section, we show how to generalize Equation 5.17 to Equation 5.20 to EID systems on a MANET with finite LRU buffers.

5.2.2 A Generalization for Epidemic Information Dissemination Systems

Recall that the conservation of flow property used in Equation 5.17 assumes that on a miss an item is always inserted into the buffer. This assumption does not hold for EID systems in a MANET, since in such a system on a miss an item is only inserted if either

1. the item can be retrieved from the (remote) origin node *or*
2. the item can be retrieved from the buffer of any other remote node.

Thus, to compute the hit rate in a EID system, we have to consider that the rate for inserting an item into a buffer depends on the content of other remote buffers. We incorporate the impact of remote buffers into the approach presented by Dan and Towsley by assuming that the content of all buffers is stochastically *independent identical distributed* (*IID*), i.e., the hit-probability for data items of partition $D(k)$ is given by $\frac{b(k,B)}{\beta(k)D}$ for each buffer regardless of the content of other buffers. Using this assumption we can compute the hit rate of the system by computing the hit rate of a single buffer.

Before showing how to compute the hit rate from remote buffers, we discuss how reasonable the assumption of IID buffer content is for different EID systems. Note that a small dependency among the content of the buffers exists for every EID system, since a node can retrieve and insert an item into the local buffer with higher probability if it is stored in many remote buffers. Nevertheless, most of the dependencies among the content of different buffers is caused by the EID system. Recall from Section 3.4 that with PDI hit rate reaches saturation or even drops if either the number of nodes N, the transmission range R, or the time-to-live for selective forwarding TTL_{query} is large. The reason for this behavior is that all nodes that receive a Response message store all extracted items, making the content of their buffers very similar. Thus, at a given time t, the content of the buffers is obviously not IID. On the other hand, recall from Section 2.4.2 that with 7DS an item in a Response message is only inserted into the buffer of the inquiring node. Thus, each query changes only the hit rate of a single buffer, implying that the assumption of independent IID buffer content is more reasonable for 7DS than for PDI.

To gain more insight into the distribution of the buffer content we compare the measured number of data holders for an item from partition $D(k)$ for 7DS and PDI to the number of data holders predicted based on the assumption of IID buffer content.

Note that the assumption implies that the distribution of the number of data holders follows a Binomial distribution, since whether the item is buffered at each node can be determined by an independent Bernoulli trial with success probability $p := \frac{b(k,B)}{\beta(k)D}$. With the assumption that consistently with Section 5.1 data items are supplied by a dedicated server node, the pmf $dh_k(n)$ and the CDF $DH_k(n)$ of the number of data holders of an item from partition $D(k)$ for $0 \leq n \leq N$ is given by:

$$dh_k(n) = \binom{N}{n} p^n (1-p)^{N-n} \tag{5.21}$$

$$DH_k(n) = \sum_{i=0}^{n} \binom{N}{n} p^i (1-p)^{N-i} \tag{5.22}$$

We compute the pmf and CDF of the number of data holders for a given data item experimentally by using the detailed simulator of the 7DS and PDI system described in Sections 5.1.1 and 5.1.2. Again, we use a one-to-one matching between keys and values, i.e., $\beta(k)D = 1$. All other parameters are chosen according to Table 5.1, with exception of the transmission range R that is set to $R = 230$ to increase the dependency among the buffers for PDI. Nodes move according to the RP mobility model. We perform simulation runs of 300.000 queries and measure the number of data holders for each data item in batches of 300 queries. 50 independent replicates of the simulation are performed and the average distributions together with 99% confidence intervals are computed.

Note that the parameter p of the Binominal distribution can be determined from the simulation results by observing that the probability for finding an item in an arbitrary buffer at a given time t is equal to the probability for picking the buffer of one of the data-holders, i.e.,

$$p = \sum_{n=0}^{N} dh_k(n) \frac{n}{N}. \tag{5.23}$$

Thus, for the mean $\bar{m}$ of the Binomial distribution holds

$$\bar{m} = pN = \sum_{n=0}^{N} dh_k(n) \cdot n, \tag{5.24}$$

that is, the mean of the binominal distributions is equal to the mean of the measured distribution. Nevertheless, the predicted variances V_p differ from the measured variances V_m as shown in Table 5.3 for the items with the popularity rank 100 and 200, respectively. Note that for PDI the measured variances differ more significantly from the predicted variances than for 7DS. To provide further insight, Figure 5.5 shows the measured CDFs and the predicted CDFs for the items with rank 100 and 200,

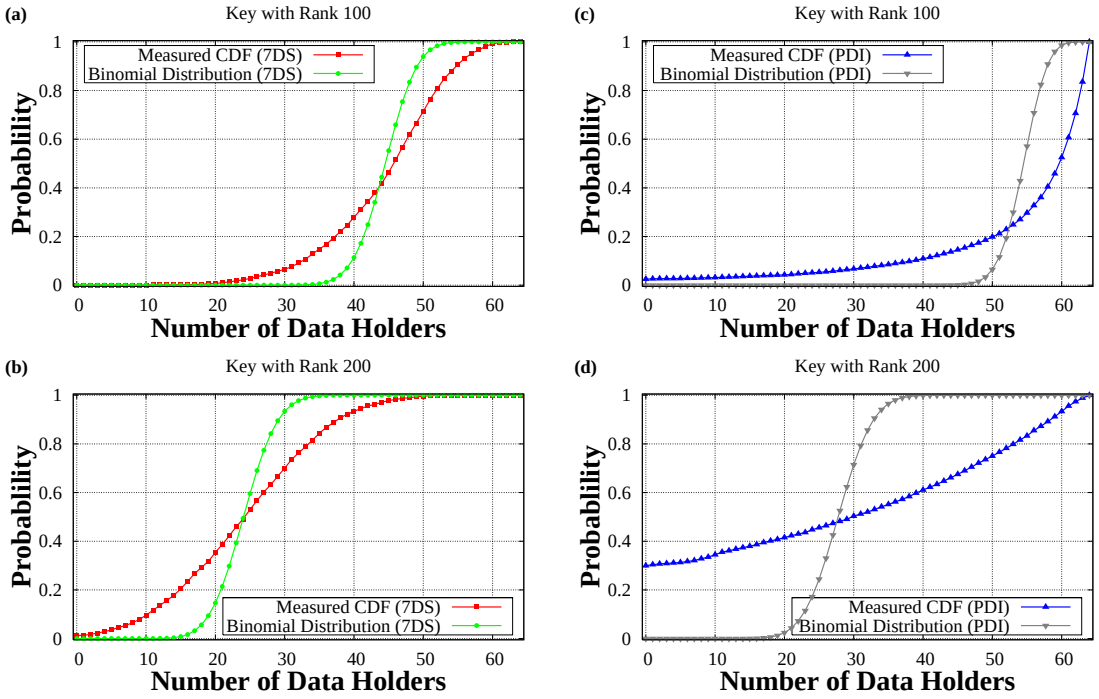

Figure 5.5. Comparison of the measured distribution of data holders and the binomial distribution for two EID systems

respectively. We find that the distributions of the number of data holders for 7DS has roughly the shape of the binomial distribution, while the exact distributions differ due to the different variances. Recall that a slight dependency among buffer content is given for every EID system, leading to the difference in the exact distributions. For PDI, the shape of the measured distributions is totally different from the shape of the Binominal distributions. Confirming the conjecture from Section 3.4 that the content of all buffers is very similar, Figure 5.5 (c) shows that the probability for caching the item with rank 100 in 60 or more buffers is more than 50%. Note that this implies that if a query cannot be serviced by four remote nodes, there is hardly any other node that can service it. Figure 5.5 (d) shows that the item with rank 200 is stored in none of the buffers with a probability of more than 30%, further reducing hit rate.

	7DS				PDI			
Rank	p	$\bar{m}$	V_p	V_m	p	$\bar{m}$	V_p	V_m
100	0.702	44.94	13.38	75.86	0.858	54.88	7.82	185.75
200	0.384	24.60	15.14	111.16	0.442	28.26	15.78	542.03

Table 5.3. Means and variances for predicted and measured distributions

We conclude from Table 5.3 and Figure 5.5 that the assumption of IID buffer content obviously does not hold for PDI with the given system parameters. In experiments not shown here, we found that the assumption of statistically independent buffers is more reasonable for PDI systems with low node density and/or small transmission range, i.e., $N \leq 64$ and $R \leq 200$. However, we focus on EID systems like 7DS in the remainder of this Chapter, since the assumption is reasonable for all values of N and R with 7DS. As validated in Section 5.3.1, the assumption of IID buffer content provides very accurate results for 7DS.

We note that for EID systems with IID buffer content the number of data items of partition $D(k)$ in the top j positions depends on both the number of data items of all partitions in the top $j-1$ positions of the local buffer, $b(k, j-1)$, and on the number of such items in remote buffers at any position, $b(k, B)$. Analogous to Equation 5.19, for $1 \leq k \leq K$, $1 \leq j \leq B$ we can write:

$$\begin{aligned} b(k,j) &= f_k(b(1,j-1), b(2,j-1), \ldots, b(K,j-1), \\ &\quad b(1,B), b(2,B), \ldots, b(K,B)\ ) \end{aligned} \tag{5.25}$$

The f_k in Equation 5.25 in general are non-linear functions in $b(k,j)$ which opposed to Equation 5.19 cannot be solved by simple back-substitution. In the following, we propose an effective iterative scheme for the approximate computation of $b(k,B)$, $1 \leq k \leq K$. For ease of exposition, we define $b(k,0) = 0$. We start with the uniform assignment of the remote buffers:

$$b^{(0)}(k,B) = \frac{B}{K}, 1 \leq k \leq K. \tag{5.26}$$

Then, for $j = 1, \ldots, B$ and $i = 1, 2, \ldots$ we iteratively compute

$$\begin{aligned} b^{(i)}(k,j) &= f_k(b^{(i-1)}(1,j-1), b^{(i-1)}(2,j-1), \ldots, b^{(i-1)}(K,j-1), \\ &\quad b^{(i-1)}(1,B), b^{(i-1)}(2,B), \ldots, b^{(i-1)}(K,B)\ ) \end{aligned} \tag{5.27}$$

until the termination condition

$$|b^{(i)}(k,B) - b^{(i-1)}(k,B)| < \varepsilon \tag{5.28}$$

is satisfied for $1 \leq k \leq K$. Note that customizing the iterative scheme to a particular EID system requires derivation of f_k. To prove the convergence of the iterative scheme, note that Equation 5.27 describes a simple fix point iteration. We can easily use the f_k to construct a function $F : \mathbb{R}^d \rightarrow \mathbb{R}^d$ with $d = (B+1)K$. The function operates on a vector $\mathbf{b} \in \mathbb{R}^d$ with $\mathbf{b}(jK+k) = b(k,j)$. Following the contraction mapping principle [Gau97], the iteration $\mathbf{b}^{(i+1)} = F(\mathbf{b}^{(i)})$ for $i = 1, 2, \ldots$ converges, if holds:

$$||F(\mathbf{b}) - F(\mathbf{b}^*)||_\infty \leq c||\mathbf{b} - \mathbf{b}^*||_\infty \tag{5.29}$$

for some constant c with $0 < c < 1$. The convergence of Equation 5.28 must be shown for each F given by the functions f_k for a particular EID system.

5.2.3 Customizing the Approach for 7DS

For customizing the approach presented in Section 5.2.2 to the 7DS system, we have to derive an expression for the push down rate of a data item of partition $D(k)$ from position j to position $j+1$ in the LRU stacks of 7DS. Using the conversation of flow observation, this is equal to finding an expression for the rate of inserting an item of partition $D(k)$ into the top j positions. In 7DS, an item can be inserted into the top j positions of the buffer either on a local or a remote request. That is:

1. on a local request, the item is not already located in the top j positions and it is either located in the bottom $B-j$ positions of the local buffer or it can be retrieved from the origin node or from any other remote node.

2. on a remote request, the item is not already located in the top j positions of the local buffer, but is located in the bottom $B-j$ positions.

To derive an expression for the relative push down rate for 7DS, we introduce additional measures in Table 5.4 for $1 \leq k \leq K$ and $1 \leq j \leq B$. Subsequently, we split the push down rate $r(k,j)$ into two rates $r_{local}(k,j)$ and $r_{remote}(k,j)$ and derive these rates as follows:

1. *Local request*: Since all nodes issue queries according to a Poisson process with identical rate λ, the rate of local requests by a node n is λ. Using the notation

Expression	Meaning
$p_{local}(k,j)$	Probability for a hit for key k in the top j positions of the local buffer.
$\bar{p}_{local}(k,j)$	Conditional probability for a hit for key k in the bottom $B-j$ positions of the local buffer given that no hit for key k occurs in the top j positions.
$p_{origin}(k)$	Probability for retrieving an item from partition $D(k)$ from the origin node.
$p_{remote}(k)$	Probability for retrieving an item from partition $D(k)$ from a remote node other than the origin node.
$r_{local}(k,j)$	Rate for pushing down items from partition $D(k)$ from stack position j to $j+1$ due to query responses from the local buffer.
$r_{remote}(k,j)$	Rate for pushing down items from partition $D(k)$ from stack position j to $j+1$ due to query responses from remote buffer.

Table 5.4. Definitions of key measures for modeling the 7DS system

introduced in Table 5.4 to derive the rate for pushing down an item on a local request, we have:

$$\begin{aligned} r_{local}(k,j) &= \alpha(k)\lambda\,(1-p_{local}(k,j)) \\ &\quad \cdot (1-(1-\bar{p}_{local}(k,j)) \\ &\quad \cdot (1-p_{origin}(k))\,(1-p_{remote}(k))) \end{aligned} \tag{5.30}$$

2. *Remote request*: Under the assumption that a node sends a query message even if it stores a matching item in the local buffer, analogous to case (1) the rate of remote requests is $(N-1)\lambda$. For the rate for pushing down an item k on a remote request, we have:

$$r_{remote} = \alpha(k)(N-1)\lambda\rho\,(1-p_{local}(k,j))\,\bar{p}_{local}(k,j) \tag{5.31}$$

Using $r_{local}(k,j)$ and $r_{remote}(k,j)$, the overall push down rate for an item of partition $D(k)$ for $1 \leq k \leq K$ and $1 \leq j \leq B$ is given by:

$$r(k,j) = r_{local}(k,j) + r_{remote}(k,j) \tag{5.32}$$

Subsequently, we show how to derive the individual terms of Equation 5.32. Similar to [DT90], the probability for a hit in the top j positions of the local LRU buffer is given by:

$$p_{local}(k,j) = \frac{b(k,j)}{\beta(k)D} \tag{5.33}$$

For computing the probability for a hit in the bottom $B-j$ positions, recall that in Equation 5.30 and Equation 5.31 this probability is used under the condition that no hit has occurred in the top j stack positions. Thus, we derive the conditional probability as:

$$\bar{p}_{local}(k,j) = \frac{p_{local}(k,B) - p_{local}(k,j)}{1-p_{local}(k,j)} \tag{5.34}$$

Neglecting border effects in the simulation area similar to Section 5.1, the probability for a successful transmission to a particular node is given by ρ. Instead of assuming that data items are supplied to the system by a dedicated server node, we make the more realistic assumption that all nodes supply some of the data items, where items are assigned to nodes in a round-robin fashion. Thus, the origin node of a data item from partition $D(k)$ is the local node with probability $1/N$ and an arbitrary remote node with probability $(N-1)/N$, the probability for retrieving an item from the origin node is given by:

$$p_{origin}(k) = \frac{1}{N} + \frac{N-1}{N}\rho \tag{5.35}$$

Using the assumption of IID buffer content discussed in Section 5.2.2, the computation of the probability for receiving an item from a remote buffer can be broken down to the probability that exactly n remote nodes receive the QUERY message and at least one of them stores a matching data item in its local buffer:

$$\begin{aligned} p_{remote}(k) &= \sum_{n=1}^{N-1} \binom{N-1}{n} (\rho)^n \\ &\quad \cdot (1-\rho)^{N-1-n} \left(1 - (1 - p_{local}(k,B))^n\right) \end{aligned} \tag{5.36}$$

Putting Equation 5.33 to Equation 5.36 into Equation 5.32 completely specifies the overall push down rate for an item of partition $D(k)$. Subsequently, following Equation 5.18 we derive the probabilities:

$$p(k,i) \approx \frac{r_{local}(k,j-1) + r_{remote}(k,j-1)}{\sum_{l=1}^{K} \left(r_{local}(l,j-1) + r_{remote}(l,j-1)\right)} \tag{5.37}$$

Note that as in Equation 5.18 the rate λ of the query arrival process cancels out in Equation 5.37. Subsequently, the $b(k,j)$, can be determined by summation of the probabilities $p(k,j)$ according to Equation 5.19.

Finally, the overall hit rate HR_{7DS} of the 7DS system can be determined using the probabilities that upon a query for key k the items matching k can be retrieved either from the local buffer, the origin node, or the buffer of a remote node. That is:

$$\begin{aligned} HR_{7DS} &= \sum_{k=1}^{K} \alpha(k) \left(1 - (1 - p_{local}(k,B))\right. \\ &\quad \left. \cdot (1 - p_{origin}(k)) \left(1 - p_{remote}(k)\right)\right) \end{aligned} \tag{5.38}$$

Each iteration of the iterative scheme stated in Equation 5.27 requires the computation of $b(k,j)$ for $k = 1, \cdots, K$ and $j = 1, \cdots, B$. Furthermore, each iteration of Equation 5.27 requires one evaluation of Equation 5.36 with $N-1$ summations for $k = 1, \cdots, K$. Thus, the complexity of each iteration of the scheme is $O(KN+KB)$. Canceling the iteration of Equation 5.27 when either the termination condition Equation 5.28 is satisfied or a maximum number of c iterations is performed, the iterative scheme for calculating the hit rate of the local buffer has an overall complexity of $O(KN + KB)$. For all 7DS system models analyzed in Section 5, the corresponding scheme converges in less than 40 iterations. The subsequent computation of the overall hit rate of the EID system using Equation 5.38 has the additional complexity $O(KN)$. Putting it altogether, the overall computational complexity of the iterative scheme is $O(KN + KB)$.

We omit a formal proof of the convergence following Equation 5.29 but state that convergence has been reached for all experiments performed in Section 5.3. Note that the contraction mapping principle used in Equation 5.29 is sufficient for

the existence of a unique solution of the non-linear equation system in Equation 5.27, but not necessary. That is, the fact that convergence is reached in the performance experiments does not imply that the system has a unique solution. However, for an experimental validation we performed a series of experiments with arbitrary initial values $b^{(0)(k,B)}$ and found that the system converged against the same solution for all values.

5.2.4 System Models for 7DS Variants

In this section, we show how to extend the basic model presented in Section 5.2.3 to represent the different variants of 7DS introduced in [PS01]. As first extension of the model, we include server-client (S-C) scenarios using fixed and mobile infostations. For a 7DS variant with fixed infostations (FIS), we assume that the data is not provided by the mobile nodes, but by N_{FIS} fixed infostations that are placed within the area A with an optimal spatial distribution. That is, the area covered by the infostations does not overlap. Note that retrieving a data item from an infostation requires bi-directional communication to exchange query and response messages. Thus, the infostations have an effective transmission range that is equal to the transmission range of the mobile nodes, R, even if it is possible to operate infostations with much higher transmission ranges in practical applications due to a much better power supply. Opposed to the infostations used in Section 3.3, we assume an optimal spatial distribution, each infostation linearly increases the probability for retrieving an item from the origin node. Thus, Equation 5.35 changes to:

$$p_{origin}(k) = N_{FIS} \cdot \rho \tag{5.39}$$

The assumption of an optimal spatial distribution does not hold for mobile infostations (MIS). Therefore, with N_{MIS} mobile infostations a data item can be retrieved if any of the mobile infostations is reachable, where the radio coverage of the mobile infostations may overlap. In this case, Equation 5.35 changes to:

$$p_{origin}(k) = N_{MIS} \cdot \left(1 - (1 - \rho)^{N_{MIS}}\right) \tag{5.40}$$

Consistent with [PS01], we use $N_{FIS} = N_{MIS} = 1$ in our performance studies. Note that in this case 5.39 is equal to 5.40.

When using 7DS in a P2P variant instead of a S-C variant, the mobile nodes must supply data items. For example, in a mobile Web browsing application, a mobile node may download Web pages using a fixed network connection before joining the 7DS system. Subsequently, the mobile node supplies the Web pages to other nodes. In

such scenario, it is not realistic to assume that each data item is stored by exactly one mobile node. We rather assume that the number of origin nodes for a data items from partition $D(k)$ reflects the popularity of the items $\alpha(k)$, $1 \leq k \leq K$. Thus, Equation 5.35 must be changed to include the increased quantities, while ensuring that each data item is available from at least one mobile node by taking the ceiling for values smaller than 1:

$$p_{origin}(k) = \frac{\lceil \alpha(k)N \rceil}{N} + \left(1 - \frac{\lceil \alpha(k)N \rceil}{N}\right)\left(1 - (1-\rho)^{\lceil \alpha(k)N \rceil}\right) \tag{5.41}$$

Beside the S-C and P2P variants without power conservation denoted as FIS-NP and NP, respectively, [PS01] considers 7DS variants that use power conservation. These variants are denoted as FIS-P and P, respectively. Recall from Section 2.4.2 that with power conservation, the mobile nodes divide the operation cycle in ON periods of duration P_{ON} and OFF periods of duration P_{OFF}. During an OFF period, the mobile node switches off the wireless communication interface and does not respond to any queries. Since the ON and OFF periods are not synchronized among the mobile nodes, the probability for contacting a mobile node in an ON period is $P_{ON}/(P_{ON}+P_{OFF})$. To model power conservation in P2P mode, Equations 5.36 and 5.41 can easily be extended:

$$\begin{aligned} p_{remote}(k) &= \sum_{n=1}^{N-1} \binom{N-1}{n} \left(\frac{\pi R^2}{A}\right)^n \left(1 - \frac{\pi R^2}{A}\right)^{N-1-n} \\ &\quad \cdot \left(1 - \left(1 - \frac{P_{on}}{P_{on}+P_{off}} p_{local}(k,B)\right)^n\right) \end{aligned} \tag{5.42}$$

$$\begin{aligned} p_{origin}(k) &= \frac{\lceil \alpha(k)N \rceil}{N} + \left(1 - \frac{\lceil \alpha(k)N \rceil}{N}\right) \\ &\quad \cdot \left(1 - \left(1 - \frac{P_{on}}{P_{on}+P_{off}}\rho\right)^{\lceil \alpha(k)N \rceil}\right) \end{aligned} \tag{5.43}$$

As last feature, [PS01] considers periodical query repetition in the case that a data item is not retrieved. Note that this feature is essential for the performance analysis of the transient behavior of a 7DS system with a single data item presented in [PS01], since without query repetition no node will be interested in the item after some time and dissemination will stop. In realistic applications, a limited number of N_{RT} repetitions with an exponential back-off time is obviously preferable than unlimited query repetitions. For ease of exposition, we count the initial query as first repetition. It is easy to see that the number of repetitions changes the probability for an access to a key k, $\alpha(k)$, since a query for key k of partition $D(k)$ is likely to be repeated, if the probability for a miss is high. The probability for a hit $p_1(k)$ in the

first repetition of a query for an item from partition $D(k)$ is given by the probability that an item can be either retrieved from the local buffer, the origin or a remote buffer, i.e.,

$$p_1(k) = 1 - (1 - p_{local}(k, B))\,(1 - p_{origin}(k))\,(1 - p_{remote}(k)) \tag{5.44}$$

In contrast, the item will not be in the local buffer on successive repetitions. Thus, the probability for a hit $p_2(k)$ is given by:

$$p_2(k) = 1 - (1 - p_{origin}(k))\,(1 - p_{remote}(k)) \tag{5.45}$$

Using Equation 5.44 and 5.45, the expected number of repetitions of a query for an item of partition $D(k)$ is given by:

$$\begin{aligned} R(k) &= p_1(k) + (1 - p_1(k)) \Big(1 + \textstyle\sum_{n=1}^{N_{rt}-2} \left(n \left((1 - p_2(k))^{n-1} p_2(k)\right)\right) \\ &\quad + (N_{RT} - 1)\,(1 - p_2(k))^{N_{RT}-2}\Big) \\ &= p_1(k) + (1 - p_1(k)) \left(1 + \frac{1 - (1 - p_2(k))^{N_{RT}-1}}{p_{2(k)}}\right) \end{aligned} \tag{5.46}$$

The effective access probability to partition $D(k)$ with N_{RT} queries can be determined by:

$$\hat{\alpha}(k) = \frac{R(k)\alpha(k)}{\sum_{l=1}^{K} R(l)\alpha(l)} \tag{5.47}$$

To model the 7DS variants considered in [PS01], we use Equation 5.41 to model a P2P scenario without power conservation (NP) as well as Equation 5.42 and Equation 5.43 to model a P2P scenario with power conservation (P). For the S-C scenarios, we use Equation 5.39 to model a scenario without power conservation (FIS-NP) as well as Equation 5.39 and Equation 5.42 to model a scenario with power conservation (FIS-P). Note that [PS01] considers passive querying in S-C scenarios, i.e., a mobile node only sends queries when it receives an advertisement from an infostation. Thus, the hit rate is always equal to 1. We will rather consider the non-trivial case of using infostations together with active queries. To avoid confusion, we will denote the resulting scenarios as FIS-NP* and FIS-P*, respectively. The impact of query repetition is only analyzed for the NP variant by using Equation 5.41 and Equation 5.47.

5.3 A Steady State Performance Study of 7DS

5.3.1 Validation of the Approximate Model

In this section, we validate the approximate model for the basic variant of 7DS as derived in Section 5.2.3 by comparison to the results obtained by the detailed simulator used in Section 5.1.3. Recall that the approximate model assumes that data items are not contributed to the system by a dedicated server node, but by all nodes participating in the system. Hence, the detailed simulator is modified to consider this functional difference and the results obtained in this section may differ from the results presented in Section 5.1.3. Based on the observations from Section 5.1.3, we use the RWP mobility model with the parameters $v_{max} = 2m/s$ and $T_{hold} = 30s$ in all simulation studies. All further assumptions are consistent to Sections 5.1.3 and 5.2, respectively.

In a first experiment, we validate the approximate model for different numbers of mobile nodes N and different buffer sizes B in Figure 5.6 (a). The results derived by the approximate model closely match the simulation results, with an average relative difference of 0.73% for $B = 64$ and 0.25% for $B = 256$. Again, for small buffer sizes, i.e., $B = 64$, the approximate model underestimates the hit rate with an average relative difference of 2.07% and a maximum of 4.3%. Note that [DT90] also reports a maximum error of about 3% for small buffers, which results from the approximation used in Equation 5.18. Figure 5.6 (b) shows a validation of the approximate model for different transmission ranges R ranging from $50m$ to $250m$. We observe average relative differences between the simulation results and the approximate model that range from 0.59% for $B = 256$ to 1.09% for $B = 16$. Even for small buffer sizes, the curve shows an excellent agreement between approximate model and simulation results. To validate the approximate model for different degrees of locality in the query stream, we plot the hit rate as a function of the Zipf parameter γ in Figure 5.6 (c). Again, we observe a slight underestimation of hit rate for small locality and small buffers, resulting in an average relative difference of 3.71% with a maximum of 8.54%. For larger buffers, the average relative difference decreases to 0.48%. Figure 5.6 shows that the approximate model provides reasonable estimates of the hit rate for various parameter settings.

We performed various experiments to validate the approximate model for other system configuration as well as for the 7DS variants NP, P, FIS-NP*, and FIS-P*. We do not include these results but state that all validations show an excellent agreement between the approximate model and the simulation results.

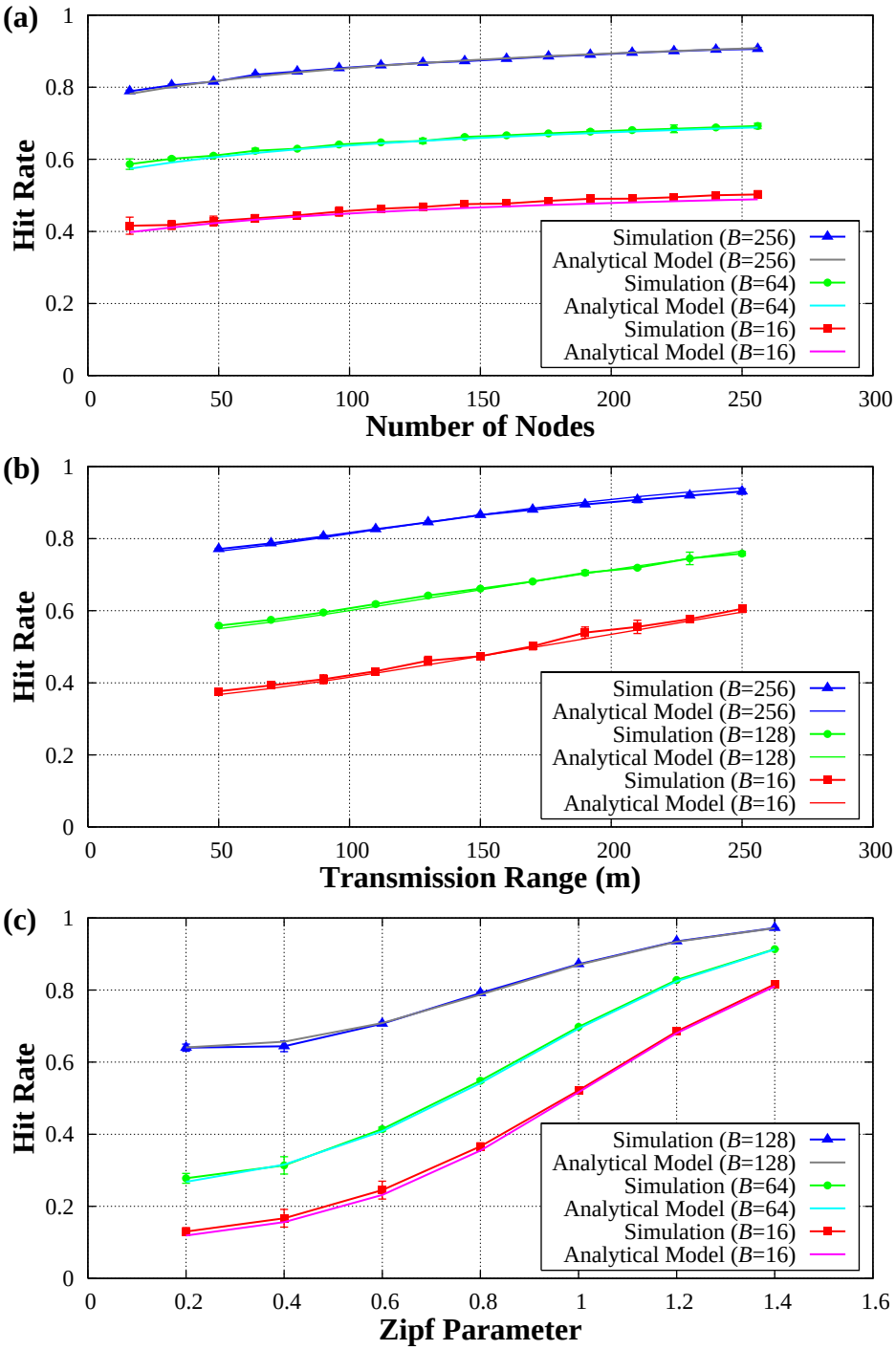

Figure 5.6. Validation of the approximate model for (a) different system sizes, (b) different transmission ranges, and (c) different degrees of query locality

5.3.2 Comparative Evaluation of 7DS Variants

In the remainder of this section, we compare the results that are derived by transient analysis of 7DS in [PS01] to the long-run results derived by the approximate steady state model. Similar to [PS01], we consider three different transmission ranges, i.e., a high transmission range of $R = 230m$, a medium transmission range of $R = 115m$, and a low transmission range of$R = 57.5m$. In many experiments, we consider two different buffer sizes $B = 64$ and $B = 256$. Recall that in a typical 7DS application data items constitute Web objects. Assuming an average object size of $20KB$, 256 data items constitute about $5MB$ of data, which is the size of a typical Web browser cache that can be easily handled by a notebook. 64 data items constitute about $1.25MB$, a data volume that can be handled by a modern PDA.

To evaluate the impact of the different 7DS variants on data dissemination in the long run, we analyze the average fraction of data holders for each data item. Figure 5.7 plots the average fraction for each data item for the selected variant of 7DS, i.e., NP, P, FIS-NP*, and FIS-P*, respectively. The data items are associated with their popularity rank. For both variants using power conservation, i.e., P and FIS-P*, we set $P_{ON} = P_{OFF}$, that is, for the *relative duration of the off period* holds $\frac{P_{OFF}}{P_{ON}+P_{OFF}} = 0.5$. [PS01] reports significant differences in the fraction of data holders for data items with a different popularity during the transient phase. Supplementing this result, we find that all 7DS variants perform almost equal in the long run. This shows that both power conservation and infostations only affect the speed of the dissemination process in the transient phase, but not the long run behavior. [PS01] considers only popular data items, which are comparable to the items with rank 1 to 5 in our study. The experiments presented in [PS01] show that these items are stored by a significant fraction of the mobile nodes after a transient phase of 25 minutes. Extending the observations of [PS01], we find that even in the long run an unpopular item is hardly stored by any mobile node. E.g., with small buffers more than 80% of the data items are stored by less than 5% of the mobile nodes. Comparing the individual 7DS variants, we find that NP and FIS-NP* store slightly more unpopular data items than P and FIS-P*. However, since this observation is not very significant, we conclude from Figure 5.7 that neither the transmission power nor the selected variant of 7DS has significant impact on the long run behavior.

Recall that the performance of 7DS not only depends on the hit rate of the local buffer, but also on the hit rate of remote buffers. Since the long-term remote buffer hit rate cannot be analyzed using transient simulation, this component is omitted in [PS01]. We plot the hit rate as a function of the buffer size for all 7DS variants and three transmission ranges. Figure 5.8 shows that the hit rate grows with the buffer

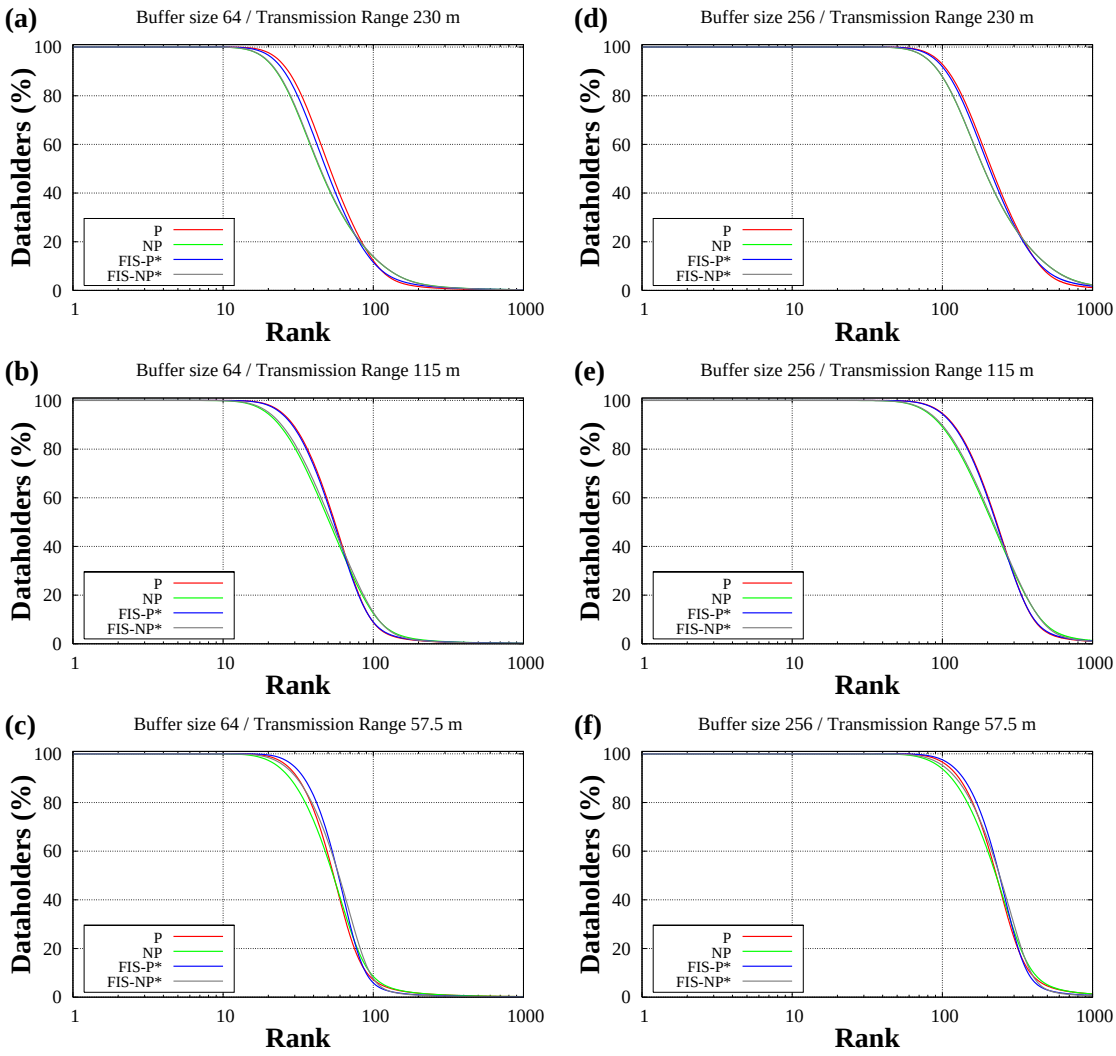

Figure 5.7. Long run behavior of data item dissemination for different 7DS variants

size in a log-like fashion with a minimum of 0.48 and a maximum of 0.92 depending on the 7DS variant. Decreasing the transmission range from $230m$ to $57.5m$ results in hit rates between 0.37 and 0.77. That is a reduction of about 0.2 for NP and FIS-NP*, and by about 0.1 for P and FIS-P*. This confirms the conclusion of [PS01] that the transmission range has a significant impact on 7DS performance. Nevertheless, comparing Figures 5.7 and 5.8 shows a different rational behind this conclusion: the reduction of the hit rate is due to a reduced number of hits at remote buffers for low transmission ranges, and not due to a significant different distribution of data items in the buffers of the mobile nodes. As a consequence, using the variant P and FIS-P* instead of NP and FIS-NP*, i.e., enabling power conservation, reduces the

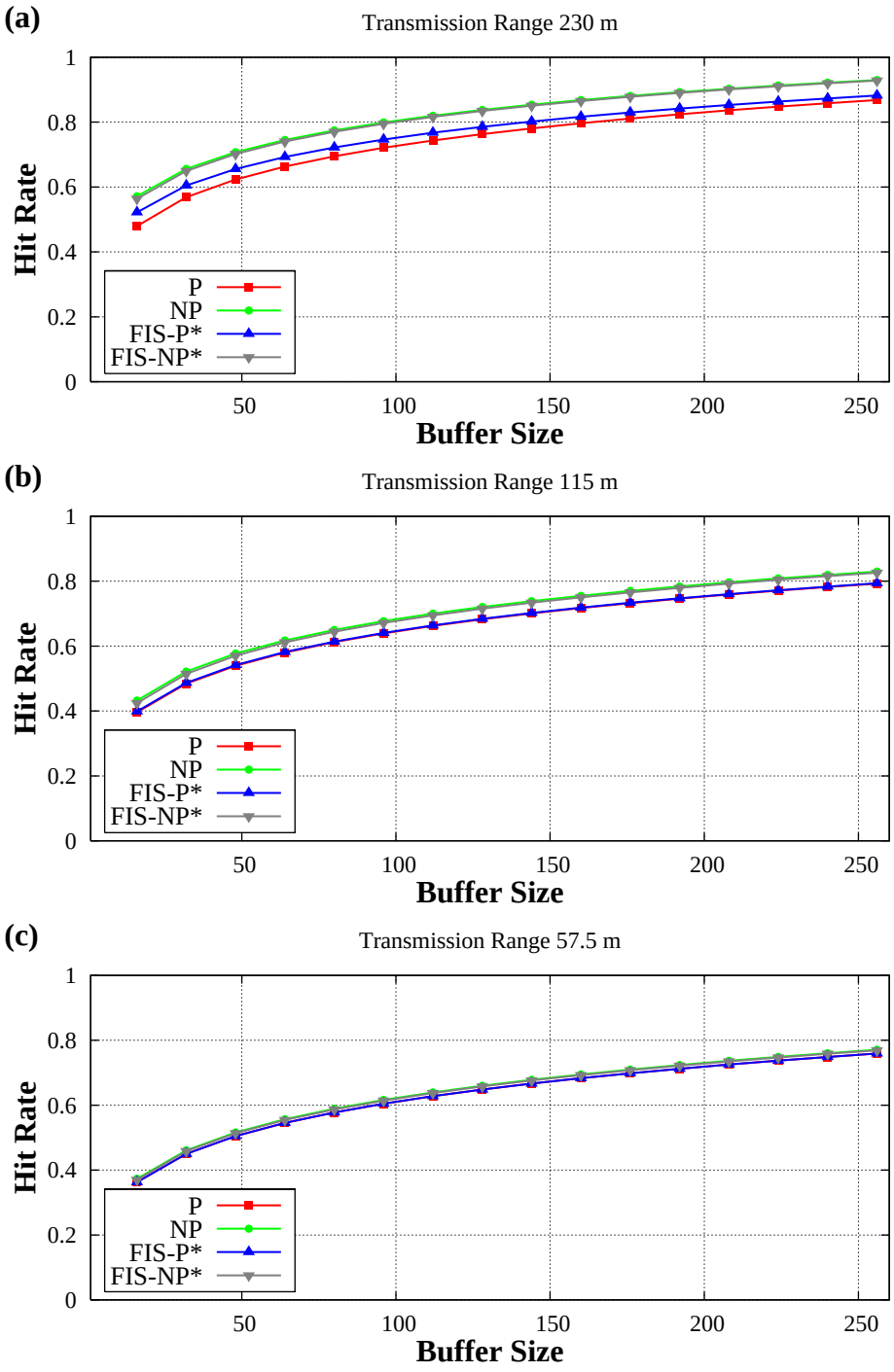

Figure 5.8. Comparison of hit rates achieved by different 7DS variants

hit rate by up to 0.1 or 18% for high transmission ranges as shown in Figure 5.8 (a). Recall that power conservation reduces the probability for retrieving a data item from a remote buffer, since remote mobile nodes respond to queries only 50% of the time. Infostation-based 7DS variants are less sensitive to power conservation, since the infostations do not employ power conservation. The performance gain due to infostations is largest for very small buffers, where the hit rate is increased by 0.05. An additional evidence for the impact of remote hits on the system performance is given by Figures 5.8 (b) and (c). These figures indicate that limiting the transmission power reduces the performance loss caused by power conservation. In particular for a low transmission power, the selected variant of 7DS has no impact on the system performance.

5.3.3 Sensitivity to Power Conservation

In the next experiment, we compute the hit rate as a function of the relative duration of the OFF period $\frac{P_{OFF}}{P_{ON}+P_{OFF}}$. The results for different buffer sizes and transmission ranges are shown in Figure 5.9. Confirming the conclusions from Figure 5.8, we find that power conservation affects the long-run performance of 7DS only if the transmission range is high. In these scenarios, aggressive power conservation degrades the hit rate by 22% for a pure P2P system and by 16% if infostations are used. This extends the results from [PS01], which reports a significant impact of power conservation for all transmission ranges in the transient phase. Consistently for both buffer sizes, Figure 5.9 shows that FIS-P* outperforms P for high transmission ranges, in particular if power conservation is used aggressively. Comparing Figure 5.9 (a) to (b) and (d) to (e), respectively, we find that an OFF period of 90% yields approximately the same hit rate as reducing the transmission range from $230m$ to $115m$. That is, using OFF periods of 50%, we have a hit rate of 0.66 for high transmission ranges and small buffers with the 7DS variant P. Extending the relative duration of the OFF period to 90% even reduces the hit rate to 0.57. In contrast, using a medium transmission range and no power conservation yields a hit rate of 0.62. Applying the radio propagation model and network interface setting used in [PS01], this reduction of the transmission rage equals a reduction of transmission power from $281.8mW$ to $17.6mW$, i.e., a reduction of 93.7%. Power conservation as proposed in [PS01] with an OFF period of 0.9 reduces both the energy required to listen to remote queries and the number of generated response messages by 90%. Nevertheless, the node has to send an identical number of query messages. Since a reduced transmission power saves 93.7% of the energy consumption for each message, reducing the transmis-

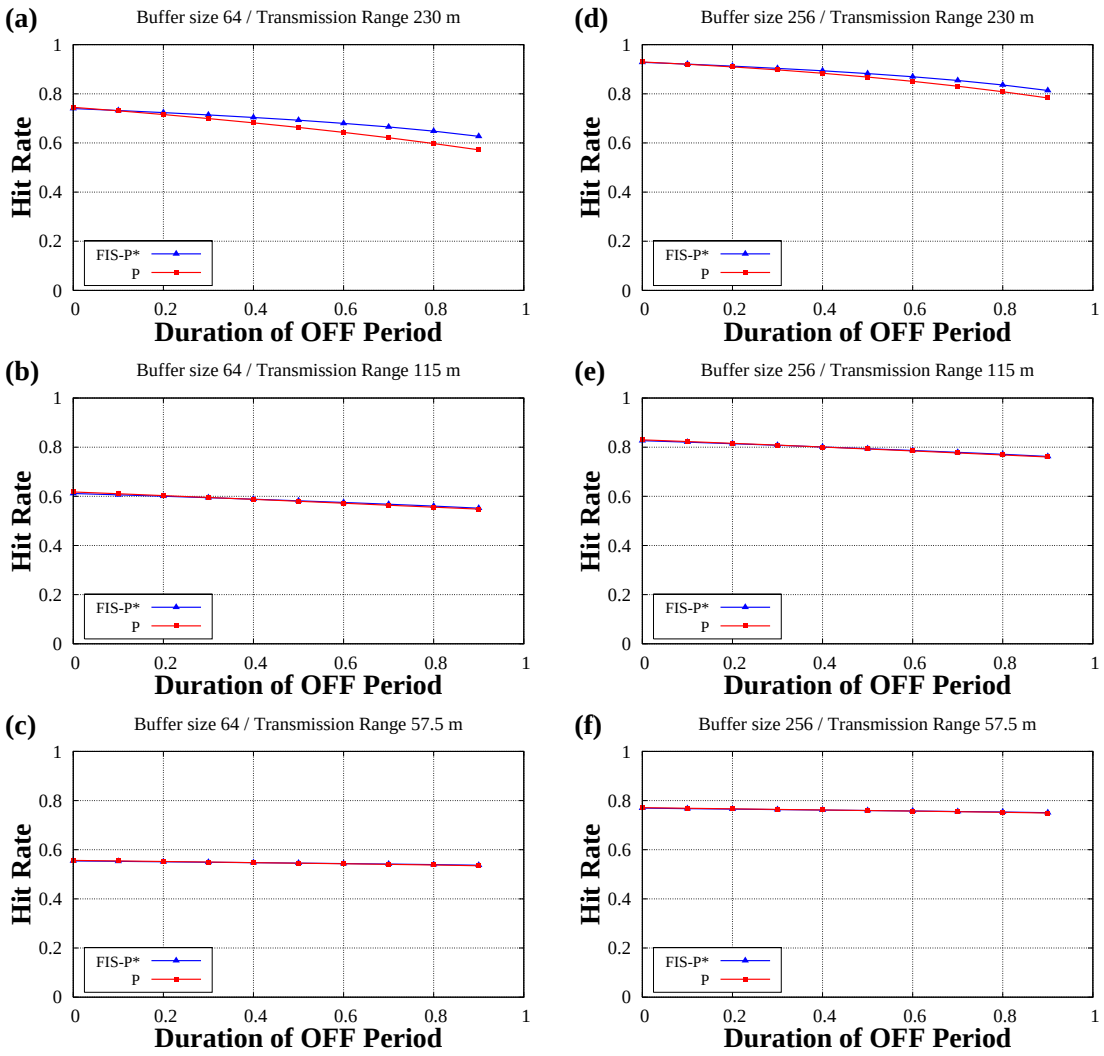

Figure 5.9. Impact of OFF period on the performance of the 7DS variants P and FIS-P*

sion range saves significantly more power than using long OFF periods. Thus, we conclude from Figure 5.9 that reducing the transmission power yields a much more efficient approach for reducing the power consumption than switching off the wireless interface.

5.3.4 Impact of Query Repetition

In a last set of experiments, we extend the results of [PS01] by computing the average number of query repetitions in the long run. We consider a maximum number of repetitions of three, five, and seven. In this experiment, we use the 7DS variant

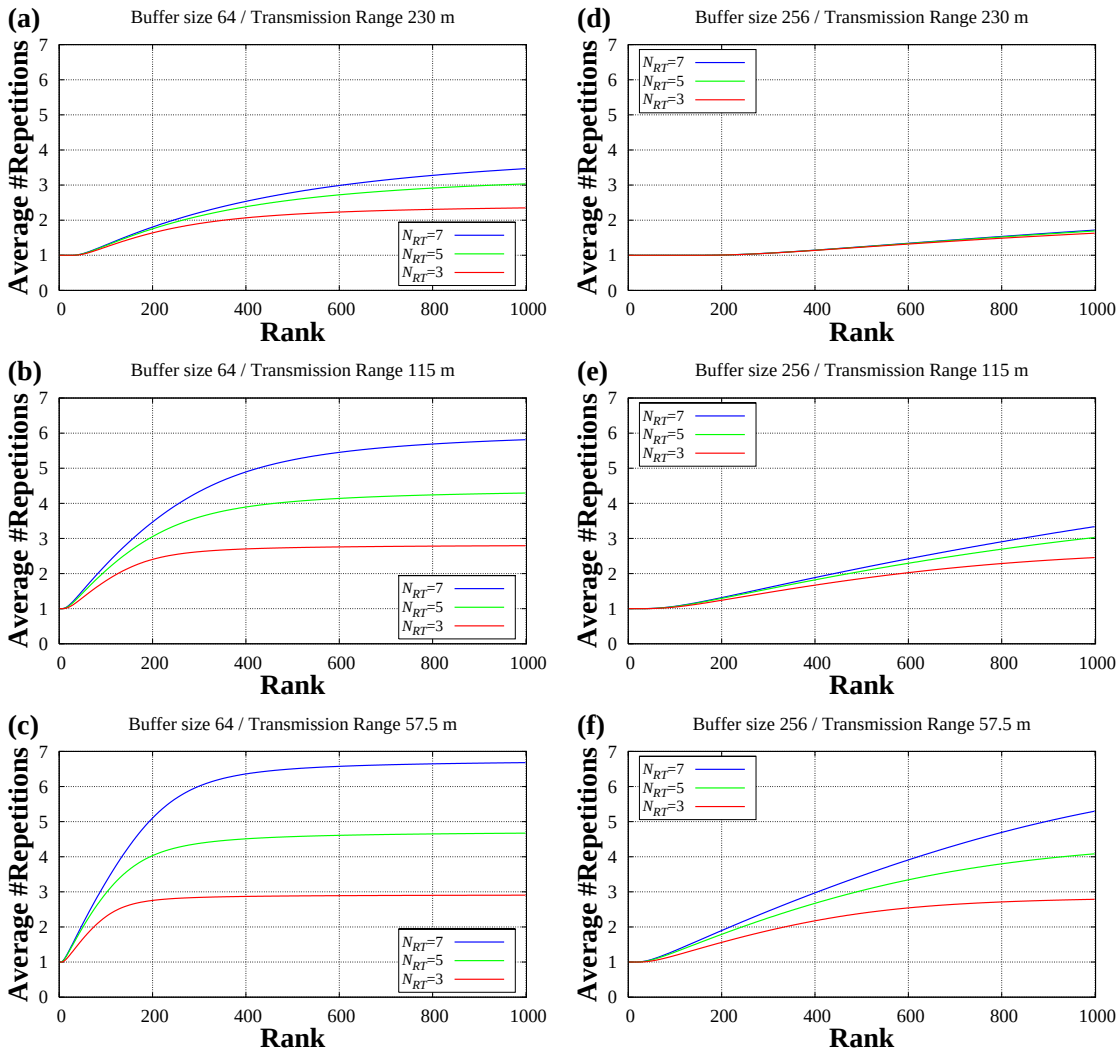

Figure 5.10. Average number of query repetitions for each data item

NP, which achieves the best performance as shown in Figure 5.8. The results for all data items, small and large buffers, and high, medium, and low transmission ranges are shown in Figure 5.10. Figures 5.10 (a) and (d) show that for high transmission ranges the average number of query repetitions is significantly below the maximum number regardless of the buffer size and the popularity of a data item. For low transmission ranges and small buffers, the number of maximum repetitions is almost reached for 60% to 80% of the data items as shown in Figure 5.10 (c). Recall from Section 5.3.2 that in these scenarios, 80% of the items are stored in less than 5% of the buffers. Larger buffers significantly reduce the average number of repetitions at low transmission ranges as shown in Figure 5.10 (f). Here, the average number of repetitions is below six even for unpopular data items if the maximum number of

repetitions is set to seven.

Figure 5.10 leads to the conclusion that query repetition is particular useful for small buffers. To gain further insight into the tradeoff between buffer size and the maximum number of query repetitions, we compute the hit rate as a function of the buffer size for high, medium and low transmission range. Here, a query is counted as a hit if a response is received within the maximum number of repetitions. We find that for small buffer sizes, aggressive query repetition increases the hit rate by up to 61% depending on transmission range. However, hit rate is a concave function of the number of repetitions, so increasing the number of repetitions further than seven does not yield a significant performance gain. Figure 5.11 (a) confirms the conjecture from Figure 5.10 that aggressive repetitions does not increase the performance of 7DS for high transmission ranges and large buffer sizes. That is, if the buffer size exceeds 120, a maximum number of three repetitions perform almost equal to five or more repetitions. To achieve a reasonable hit rate beyond 0.9, we recommend a maximum number of five repetitions for high transmission ranges and buffers smaller than 120 entries. For larger buffers, three repetitions suffice. If the transmission range is medium to small, a hit rate of 0.9 is not reached regardless of the maximum number of repetitions. Here, the buffer size must be larger than 100 items for a medium transmission ranges and larger than 155 for a small transmission range. In both cases, a maximum of five to seven repetitions is required to achieve a reasonable performance.

5.4 Summary

To provide insight into the driving forces of epidemic information dissemination, in this chapter we presented stochastic models for EID systems with finite buffers and LRU replacement. We showed how to model such system as discrete-time Markov chain (DTMC) with a stationary probability distribution. We customized the DTMC model for two state-of-the-art EID systems, i.e., Seven Degrees of Separation (7DS) and Passive Distributed Indexing (PDI). Unfortunately, the state space of the DTMCs grows exponentially in the number of nodes and the number of data items in the system, and even the number of successor states of a given state grows exponentially in the number of nodes. Thus, for systems of relevant size, the DTMC model cannot be employed to compute the steady state stationary probability distribution numerically. Nevertheless, it enables efficient simulation of the EID systems, with an excellent agreement between the simulation results derived by the DTMC simulator and a detailed simulator that considers realistic mobility models and detailed

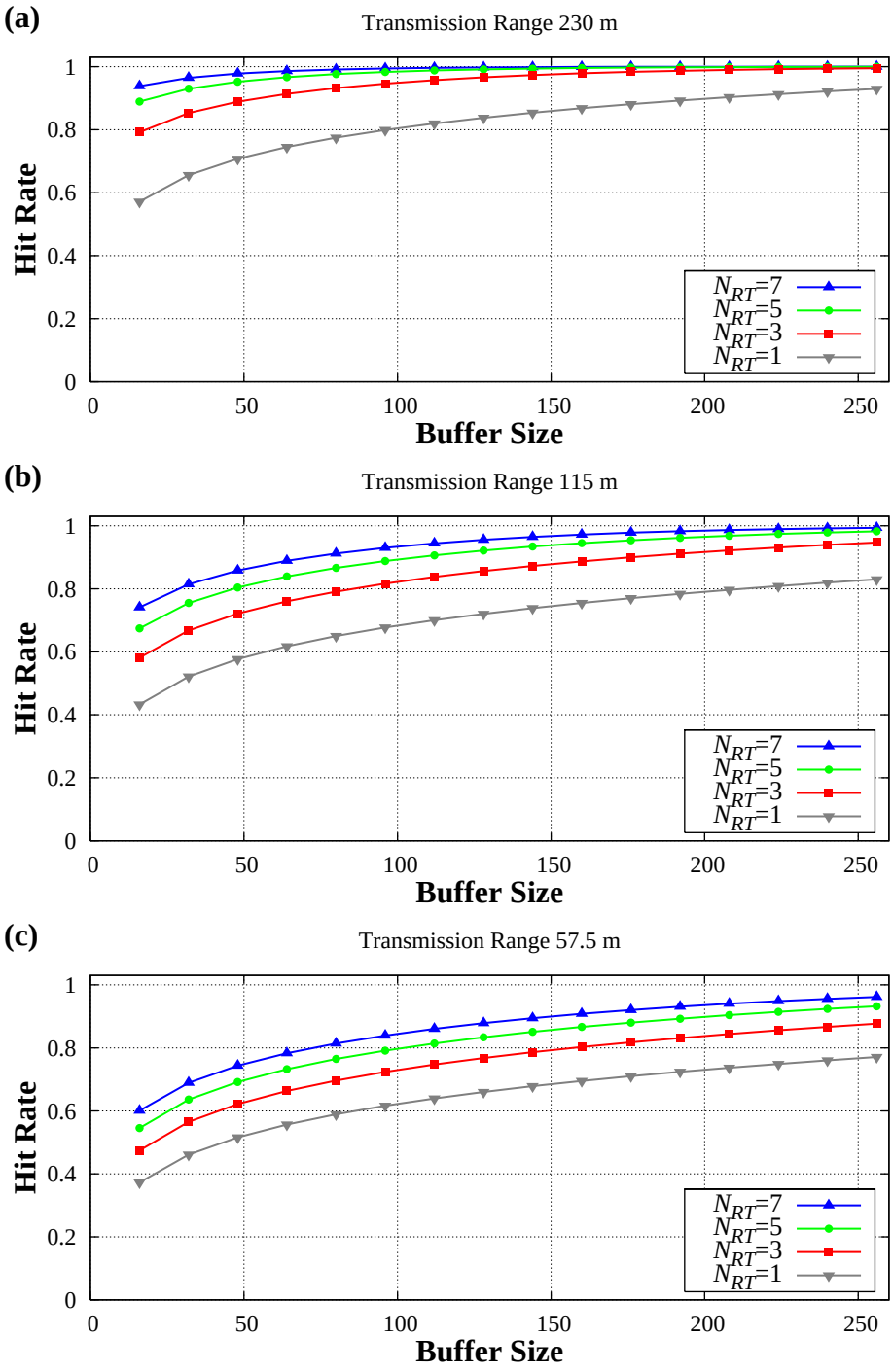

Figure 5.11. Hit rate of 7DS system with query repetition

message transmissions.

To enable analytical computation of performance measures despite of the size of the state space, we presented an approximate analysis approach for analyzing the performance of EID systems with finite buffers. The modeling approach explicitly represents the spread of multiple data items and finite LRU buffers at mobile devices. Previous work just considered the transient behavior for a single data item, assumed buffers of infinite size and required off-line simulation for determining model parameters. The approach is an extention of an approximate model for the steady-state hit rate of a stand-alone LRU buffer presented by Dan and Towsley [DT90]. The extention exploits the fact that the content of the buffers of all nodes is independent identical distributed, an assumption that holds quite well for 7DS, but does not hold for PDI as revealed by simulation results.

Using the approximate approach, we showed how to derive performance models for four variants of 7DS [PS01], a well-known system for P2P data sharing in MANET. Using these performance models, we presented a comparative evaluation of four 7DS variants P, NP, FIS-P* and FIS-NP*. Furthermore, we investigated the impact of power conservation and query repetition. We found that neither the transmission range nor the selected variant of 7DS has a significant impact on the fraction of dataholders in the long run. However, for high transmission ranges the selected variant of 7DS has a significant impact on the hit rate. Depending on the 7DS variant and the buffer size, hit rates between 0.48 and 0.92 can be achieved. 7DS variants NP and FIS-NP* outperform variants P and FIS-P* by up to 18%. For low transmission ranges, the hit rate lies between 0.37 and 0.77 regardless of the 7DS variant. Enabling aggressive power conservation in scenarios with high transmission ranges degrades the hit rate by 22% for a pure P2P system, and by 16%, if infostations are additionally deployed. Furthermore, a reduced transmission range of $115m$ yields higher hit rates than using long OFF periods at $230m$ transmission range. Repeating queries up to seven times increases hit rate by up to 61% for small buffers and low transmission ranges. Only for high transmission ranges and large buffers, query repetition yields marginal performance gains. Thus, the derivation of an optimized power management strategy for 7DS requires the comprehensive study of the trade-off between the duration of OFF periods, transmission range, and the number of query repetitions.

Chapter 6

Concluding Remarks

This thesis explores novel ways for implementing distributed applications on mobile devices such as notebooks, personal digital assistants, or smart phones. Equipped with network technologies for short- to medium-range wireless communication such devices can form Mobile Ad Hoc Networks (MANET) rather than communicate via a cellular infrastructure. In MANET, all mobile devices act as routers and forward packets on behalf of other devices. Thus, communication is feasible by using multi-hop routes of intermediate devices even in case devices are not located in each others transmission range.

MANET are inherently self-organizing, since they cannot rely on any fixed infrastructure. Since MANET share this property with Peer-to-Peer (P2P) systems, the P2P paradigm is much more attractive for building distributed MANET application than the Client/Server paradigm. Unfortunately, most state-of-the-art P2P systems for the Internet construct and maintain an overlay network. This thesis showed that the straight-forward deployment of overlay-based P2P systems in a MANET generates high overhead due to node mobility and intermittent connectivity. Thus, more sophisticated methods to implement P2P applications in MANET are required. Recent work in this area can be subdivided in two domains: (1) *P2P systems using cross-layer information transfer* that maintain an overlay network in conformance with the physical topology of the MANET by a close cooperation of protocols and applications running on different layers of the TCP/IP protocol stack. (2) *P2P systems using epidemic information dissemination* that exploit node mobility and transmit messages when nodes encounter, similar to the spread of an infectious disease among individuals. While P2P systems using cross-layer information transfer are beyond the scope of this thesis, a three-fold contribution is made in the domain of P2P systems using epidemic information dissemination. Within this domain, the contribution can

be subdivided into the fields of epidemic dissemination of index information, epidemic dissemination of presence information, and stochastic modeling of epidemic information dissemination systems.

Epidemic Dissemination of Index Information

As a first contribution of this thesis, a distributed lookup service for mobile applications denoted as *Passive Distributed Indexing* (*PDI*) was introduced. Beyond all previous work, PDI can be employed for key-to-value lookups in arbitrary mobile applications, explicitly uses buffers of finite capacity with Least Recently Used (LRU) replacement, and comprises mechanisms for providing coherence of disseminated information. As key concept, PDI employs epidemic dissemination of index entries in form of (key, value) pairs using index caches at the mobile devices. To foster information dissemination for devices with limited transmission range, PDI incorporates a bandwidth-efficient message relaying mechanism denoted as *selective forwarding*. Coherence of the disseminated information is provide by an implicit invalidation mechanism denoted as *value timeouts* and an explicit invalidation mechanism denoted as *lazy invalidation caches*. Both value timeouts and lazy invalidation caches rely on epidemic dissemination of control information.

A comprehensive simulation study illustrated that PDI is well-suited to implement consistent lookup operations in various mobile applications. For simulation purposes, a framework for characterizing the workload generated by a mobile application was presented and customized for four different mobile applications, i.e., mobile P2P file sharing, mobile instant messaging, a mobile information portal, and disconnected Web search. Using this framework, it was shown that with the suitable configuration of index cache size and selective forwarding PDI can achieve hit rates of more than 90% and a result coherence of more than 95% for the mobile P2P file sharing application. For the mobile instant messaging application, the PDI invalidation mechanisms can easily cope with the high rate of value expirations and even increase hit rate, since with invalidation no index cache space is wasted for stale index entries. For the mobile information portal, increasing the number of nodes up to 300 does not significantly increase hit rate, since the content of the index caches is identical in many cases. For the disconnected Web search application, a moderate number of infostations provides fair hit rates. A comprehensive description of the functionality of PDI can be found in a draft specification. Prototype implementations that conform to the specification have been successfully developed for P2P file sharing, mobile instant messaging, and disconnected web search.

Epidemic Dissemination of Presence Information

As a second contribution of this thesis, a comprehensive analysis of approaches for disseminating frequently changing presence information in highly partitioned MANET was presented. Beyond all previous work, the analysis introduced *sustained consistency* as a novel measure for the coherence of the disseminated presence information, evaluated different alternative dissemination approaches, and proposed the *System for Presence information Exchange by Epidemic Dissemination* (*SPEED*) based on the evaluation results.

For comparing different alternative approaches for disseminating presence information, a high-level stochastic model of the IM system was presented. This model was employed to derive performance bounds by discrete-event simulation. In particular, upper bounds for sustained consistency and lower bounds for the control traffic were derived. The bounds revealed that an approach using epidemic dissemination of presence information by one-time pushing of state information to all reachable nodes and successive periodical polling for up-to-date state information of all contacts constitutes the method of choice.

The performance bounds motivated the development of SPEED that comprises an optimized mix of controlled flooding, caching, and epidemic dissemination of information. In a detailed simulation study, the impact of different design parameters on the performance of SPEED was illustrated. Furthermore, the adaptive selection of the poll interval, the scope for flooding poll messages, and the information timeout based on the number of nodes located in the transmission range were proposed. A comparison to an optimized approach based on periodical flooding of presence information revealed that for transmission ranges up to $115m$ SPEED achieves up to 20% higher sustained consistency than the flooding-based approach while for larger transmission ranges, epidemic information dissemination saves up to 48% control traffic. A comparison between SPEED and the theoretical bounds revealed that the presented configuration of SPEED achieves high traffic efficiency at the expense of reduced sustained consistency. However, the sensitivity studies showed that SPEED can be easily configured to achieve higher sustained consistency at the expense of increased control traffic.

Stochastic Modeling of Epidemic Information Dissemination Systems

As a third contribution of this thesis, a stochastic modeling approach for EID systems was presented. Beyond all previous work, the approach explicitly considers the dissemination of multiple data items as well as buffers with finite capacity and LRU

replacement, conducts steady-state analysis rather than transient analysis, and does not require offline simulations to determine model parameters.

As first step towards the modeling approach, it was shown that EID systems with finite buffers can be modeled by discrete-time Markov chains (DTMCs) with a stationary probability distribution. A customization of the DTMC model for the EID systems Seven Degrees of Separation (7DS) and PDI revealed that the state space of the DTMCs grows exponentially in the number of nodes and the number of data items in the system and even the maximum number of successor states for a given state grows exponentially in the number of nodes. Thus, the DTMC model cannot be employed for numerical computation of the steady state stationary probability distribution when analyzing EID systems with a relevant size. Nevertheless, it enables efficient simulation of the EID systems.

To cope with the complexity of the stochastic model, an approximate approach for analyzing the performance of EID systems was presented. The modeling approach is an extention of an approximate model for the steady-state hit rate of a stand-alone LRU buffer. The extention exploits the fact that the content of the buffers of all nodes is independent identical distributed, an assumption that holds quite well for 7DS, but does not hold for PDI as revealed by simulation results. The approximate modeling approach was applied to derive performance models for four variants of 7DS, P, NP, FIS-P* and FIS-NP* as well as to investigate the impact of power conservation and query repetition. Steady-state performance results show that depending on the 7DS variant and the buffer size, hit rates between 0.48 and 0.92 can be achieved. Enabling aggressive power conservation in scenarios with high transmission ranges degrades the hit rate by 22% for a pure P2P system, and by 16% in case infostations are additionally deployed, while a reduced transmission range of $115m$ yields higher hit rates than using long OFF periods at $230m$ transmission range. Repeating queries up to seven times increases the hit rate by up to 61% for small buffers and low transmission ranges.

Future Research Directions

In both P2P systems and MANET a user spends resources, e.g., bandwidth and energy for forwarding messages or memory capacity for storing information, on behalf of other users. This thesis assumes a *cooperative model*, that is, all users cooperate without any restrictions. However, in real world systems users will behave rational or even selfish and, thus, may hesitate to spend resources on behalf other users without any personal benefit. *Incentive models* provide means for stimulating cooperation in mobile environments. In recent work, first attempts have been made to provide

incentive models for caching content on mobile devices [GLMT04]. Nevertheless, it has been shown that developing effective incentive models for stimulating both routing and forwarding in MANET is not an easy task due to mobility and intermittent connectivity [ZLLY05]. On the one hand, the real-world deployment of EID systems requires appropriate incentive models that constitute extension of [GLMT04]. On the other hand, epidemic information dissemination can also provide means for the development of efficient incentive models in mobile environments by providing effective communication mechanisms that can cope with the problems identified in [ZLLY05]. Both incentive models for EID systems and implementing incentive models by EID constitute promising research directions beyond the results of this thesis.

This thesis considers "isolated" MANET without a connection to the Internet. However, extending the coverage of the Internet is a promising application for MANET technology. The most prominent examples for such approaches are *ad hoc extensions of the Internet* and *wireless mesh networks*. Ad hoc extensions of the Internet consist of mobile devices that provide connectivity to an Internet gateway by using multi-hop routes based on MANET technology. Wireless mesh networks use MANET-like routing protocols to construct a self-organizing backbone of fixed wireless routers that provides Internet access for fixed or mobile clients [AWW05]. Although these technologies differ from the scenarios considered in this thesis, both basically constitute extentions of the MANET concept. Thus, extending the results of this thesis for these emerging network technologies is a promising research direction.

The approximate modeling approach presented in this thesis can only be applied for a limited, though relevant, class of EID systems. How to extend the approximate analysis approach to systems without independent identical distributed buffer content is an open research problem. Furthermore, it is not clear if other approaches for steady-state analysis can be derived, e.g., by exploiting structures in the DTMC model. A promising research direction beyond the results of this thesis constitutes the development of performance models of more general EID systems. Such performance models can definitely foster the design of EID systems for MANET and emerging network technologies by providing means for evaluating and optimizing novel approaches.

References

[APHS02] K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt. Improving Data Access in P2P Systems. *IEEE Internet Computing*, 6(1):58–67, 2002.

[Are03] M. Arens. Entwicklung einer Software-Umgebung für Simulation und Betrieb von P2P Systemen in mobilen Ad-hoc Netzen. Diplomarbeit, Universität Dortmund, 2003.

[AWW05] I. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a Survey. *Computer Networks*, 47(4):445–487, 2005.

[BÖ3] D. Bölting. Entwicklung eines verteilten Indexing-Verfahrens für P2P Systeme in mobilen Ad-hoc Netzen. Diplomarbeit, Universität Dortmund, 2003.

[Bai75] N. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Hafner, New York, NY, 2nd edition, 1975.

[BCC01] F. Brauer and C. Catillo-Chávez. *Mathematical Models in Population Biology and Epidemiology*. Springer, New York, NY, 2001.

[BCF+99] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Schenker. Web-Caching and Zipf-like Distributions: Evidence and Implications. In *Proc. IEEE Infocom 99*, pages 126–134, New York, NY, 1999.

[Bet02] C. Bettstetter. On the Minimum Node Degree and Connectivity of a Wireless Multihop Network. In *Proc. 3rd ACM Int. Symp. on Mobile Ad Hoc Networking & Computing (MobiHoc 2002)*, pages 80–91, Lausanne, Switzerland, 2002.

[Bih04] H. Bihr. Entwicklung eines Instant Messaging Systems für mobile Ad-hoc Netze. Diplomarbeit, Universität Dortmund, 2004.

[Bis01] C. Bisdikian. An Overview of the Bluetooth Wireless Technology. *IEEE Communications*, 39(12):86–94, 2001.

[BMJ+98] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proc. 4th ACM Int. Conf. on Mobile Computing & Networking (MobiCom 98)*, pages 85–97, Dallas, TX, 1998.

[Bou04] J.-Y. Le Boudec. Understanding the Simulation of Mobility Models with Palm Calculus. Technical Report IC/2004/53, EPF Lausanne, 2004.

[BRS03] C. Bettstetter, G. Resta, and P. Santi. The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks. *IEEE Trans. on Mobile Computing*, 2(3):257–269, 2003.

[BV05] J.-Y. Le Boudec and M. Vojnovic. Perfect Simulation and Stationarity of a Class of Mobility Models. In *Proc. IEEE Infocom 2005*, Miami, FL, 2005.

[CGM+01] P. Castro, B. Greenstein, R. Muntz, C. Biskidiuan, R. Kermani, and M. Papadopouli. Locating Application Data across Service Discovery Domains. In *Proc. 7th ACM Int. Conf. on Mobile Computing & Networking (MobiCom 2001)*, pages 28–42, Rome, Italy, 2001.

[CGMT03] M. Conti, S. Giordano, G. Maselli, and G. Turi. MobileMAN: Mobile Metropolitan Ad hoc Networks. In *Proc. 8th Int. Conf. On Personal Wireless Communications (PWC 2003)*, pages 169–174, Venice, Italy, 2003.

[CJ03] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). Request for Comments 3626, Internet Engeneering Task Force, 2003.

[CL98] P. Cao and C. Liu. Maintaining Strong Cache Consistency in the World-Wide Web. *IEEE Trans. on Computers*, 47(4):445–457, 1998.

[Cli01] clip2. *The Gnutella Protocol Specification v0.4*, 2001. *http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf*.

[CLP04] F. Chinchilla, M. Lindsey, and M. Papadopouli. Analysis of Wireless Information Locality and Association Patterns in a Campus. In *Proc. IEEE Infocom 2004*, pages 906–917, Hong Kong, 2004.

[CMTG04] M. Conti, G. Maelli, G. Turi, and S. Giordano. Cross Layering in Mobile Ad Hoc Network Design. *IEEE Computer*, 37(2):48–51, 2004.

[Cof73] E. Coffman. *Operating System Theory*, chapter 6. Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[CSN02] 4. K. Chen, S. Shah, and K. Nahrstedt. Cross-layer Design for Data Accessibility in Mobile Ad Hoc Networks. *Wireless Personal Communications*, 21(1):49–76, 2002.

[DDY90] A. Dan, D. Dias, and S. Yu. The Effect of Data Access on Buffer Hits and Data Contention in a Data Sharing Environment. In *Proc. 16th Int. Conf. on Very Large Data Bases (VLDB 1990)*, pages 419–431, Brisbane, Australia, 1990.

[DFGV03] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli. Age matters: efficient route discovery in mobile ad hoc networks using encounter ages. In *Proc. 4th ACM Int. Symp. on Mobile Ad Hoc Networking & Computing (MobiHoc 2003)*, pages 257–266, Annapolis, MD, 2003.

[DGH+87] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic Algorithms for Replicated Database Maintenance. In *Proc. 6th Symp. on Principles of Distributed Computing (PODC 1987)*, pages 1–12, Vancouver, Canada, 1987.

[DRS00] M. Day, J. Rosenberg, and H. Sugano. A Model for Presence and Instant Messaging. Request for Comments 2778, Internet Engeneering Task Force, 2000.

[DT90] A. Dan and D. Towsley. An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes. In *Proc. ACM. Int. Conf. on Measurement & Modeling of Computer Systems (ACM SIGMETRICS 90)*, pages 143–152, Boulder, CO, 1990.

[dWG05] C. de Waal and M. Gerharz. BonnMotion - A Mobility Scenario Generation and Analysis Tool, 2005. *http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/*.

[EGKM04] P. Eugster, R. Guerraoui, A-M. Kermarrec, and L. Massoulie. Epidemic Information Dissemination in Distributed Systems. *IEEE Computer*, 37(5):60–67, 2004.

[ERS02] D. Eastlake, J. Reagle, and D. Solo. (Extensible Markup Language) XML-Signature Syntax and Processing. Request for Comments 3275, Internet Engeneering Task Force, 2002.

[Fe05] K. Fall and K. Varadhan (editors). *The ns-2 Manual*. The VINT Project, UC Berkeley, LBL and Xerox PARC, 2005.

[FGS04] R. Friedman, M. Gradinariu, and G. Simon. Locating Cache Proxies in MANET. In *Proc. 5th ACM Int. Symp. on Mobile Ad Hoc Networking & Computing (MobiHoc 2004)*, pages 175–186, Roppongi, Japan, 2004.

[For] Gnutella Developer Forum. Gnutella - A Protocol for a Revolution. *http://rfc-gnutella.sourceforge.net*.

[Gau97] W. Gautschi. *Numerical Analysis*. Birkhäuser, Boston, MA, 1997.

[GBMY97] D.J. Goodman, J. Borras, N.B. Mandayam, and R.D. Yates. Infostations: A New system for Data and Messaging Services. In *Proc. IEEE Vehicular Technology Conference (VTC-Fall 97)*, pages 969–973, 1997.

[GLMT04] M. Goesmans, L. Li, V. Mirrokni, and M. Thottan. Market Sharing Games Applied to Content Distribution in Ad-Hoc Networks. In *Proc. 5th ACM Int. Symp. on Mobile Ad Hoc Networking & Computing (MobiHoc 2004)*, pages 55–66, Tokyo, Japan, 2004.

[GPVD99] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol. Reqest for Comments 2608, Internet Engeneering Task Force, 1999.

[GSXL02] S. Goel, M. Singh, D. Xu, and B. Li. Efficient Peer-to-Peer Data Dissemination in Mobile Ad-Hoc Networks. In *Proc. Int. Workshop on Ad Hoc Networking (IWAHN 2002)*, Vancouver, BC, 2002.

[GT02] M. Grossglauser and D. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. on Networking*, 10(4):477–486, 2002.

[GV03] M. Grossglauser and M. Vetterli. Locating Nodes with EASE: Mobility Diffusion of Last Encounters in Ad Hoc Networks. In *Proc. IEEE Infocom 2003*, pages 1954–1964, San Francisco, CA, 2003.

[HBR03] J. Hähner, C. Becker, and K. Rothermel. A Protocol for Data Dissemination in Frequently Partitioned Ad-Hoc Networks. In *Proc. IEEE Int. Symp. on Computers and Communications (ISCC 2003)*, pages 633–640, Antalya, Turkey, 2003.

[HGPC99] X. Hong, M. Gerla, G. Pei, and C. Chiang. A Group Mobility Model for Ad Hoc Wireless Networks. In *Proc. ACM Int. Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 99)*, pages 53–60, Seattle, WA, 1999.

[HKB99] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In *Proc 5th ACM Int. Conf. on Mobile Computing & Networking (MobiCom 99)*, pages 174–185, Seatle, WA, 1999.

[HLM03] K. Hanna, B. Levine, and R. Manmatha. Mobile Distributed Information Retrieval For Highly-Partitioned Networks. In *Proc. 11th IEEE Int. Conf. on Network Protocols (ICPN 2003)*, pages 38–47, Atlanta, GA, 2003.

[IEE97] IEEE Computer Society LAN MAN Standards Committee. *IEEE 802.11-1997: Standard Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1997.

[JHMJ01] J. Jetcheva, Y. Hu, D. Maltz, and D. Johnson. A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks. Internet Draft (work in progress), Internet Engeneering Task Force, July 2001. *http://www.ietf.org/proceedings/01dec/I-D/draft-ietf-manet-simple-mbcast-01.txt.*

[JMH04] D. Johnson, D. Maltz, and Y. Hu. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). Internet Draft (work in progress), Internet Engeneering Task Force, July 2004. *http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt.*

[KAL+05] H. Kaaranen, A. Ahtiainen, L. Laitinen, S. Naghian, and V. Niemi. *UMTS Networks: Architecture, Mobility and Services.* John Wiley & Sons, New York, NY, 2nd edition, 2005.

[KaZ] Sharman Networks. *KaZaA Hompage. http://www.kazaa.com.*

[KBTR02] A. Khelil, C. Becker, J. Tian, and K. Rothermel. An Epidemic Model for Information Diffusion in MANETs. In *Proc. 5th ACM Int. Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2002)*, pages 54–60, Atlanta, GA, 2002.

[Kin71] W. King. Analysis of paging algorithms. In *Proc. IFIP Congress*, pages 485–490, Ljublanjana, Yugoslavia, 1971.

[Kle05] A. Klemm. *Measurement Methodology and Tools for the Characterization of the Internet*. PhD thesis, Universität Dortmund, 2005. *Forthcoming.*

[KLVW04] A. Klemm, C. Lindemann, M. Vernon, and O. Waldhorst. Characterizing the Query Behavior in Peer-to-Peer File Sharing Systems. In *Proc. ACM Internet Measurement Conference (IMC 2004)*, pages 55–67, Taormina, Italy, 2004.

[KLW03] A. Klemm, C. Lindemann, and O. Waldhorst. A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks. In *Proc. 58th IEEE Vehicular Technology Conference (VTC-Fall 2003)*, pages 2758–2763, Orlando, FL, 2003.

[KLW04a] A. Klemm, C. Lindemann, and O. Waldhorst. Peer-to-peer Computing in Mobile Ad Hoc Networks. In E. Gelenbe and M. Calzarossa (Eds.), editors, *Performance Tools and Applications to Networked Systems*, number 2965 in Lecture Notes in Computer Science, pages 187–208. Springer-Verlag, 2004.

[KLW04b] A. Klemm, C. Lindemann, and O. Waldhorst. Relating Query Popularity and File Replication in the Gnutella Peer-to-Peer Network. In *Proc. 12th GI/ITG Conf. on Measuring, Modeling and Evaluation of Computer and Communication Systems (MMB 2004)*, pages 305–314, Dresden, Germany, 2004.

[LCC+02] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proc. 16th ACM Int. Conf. on Supercomputing*, pages 84–95, New York, NY, 2002.

[LCD00] D. Li, P. Cao, and M. Dahlin. WCIP: Web Cache Invalidation Protocol. Internet Draft (work in progress), Internet Engeneering Task Force, November 2000. *http://www.wrec.org/Drafts/draft-danli-wrec-wcip-00.txt.*

[LCP+05] E. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, 2005.

[LEH03] J. Luo, P. Eugster, and J. Hubaux. Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks. In *Proc. IEEE Infocom 2003*, pages 2229–2239, San Francisco, CA, 2003.

[LEH04] J. Luo, P. Eugster, and J. Hubaux. Pilot: Probabilistic Lightweight Group Communication System for Ad Hoc Networks. *IEEE Trans. on Mobile Computing*, 3(2):164–179, 2004.

[LH86] K. Li and P. Hudak. Memory Coherence in Shared Virtual Memory Systems. In *Proc. 5th ACM Annual Symp. on Principles of Distributed Computing (PODC 86)*, pages 229–239, Calgary, Canada, 1986.

[LLW05] C. Lambert, C. Lindemann, and O. Waldhorst. Effective Dissemination of Presence Information in Highly Partitioned Mobile Ad Hoc Networks. Technical Report 799, University of Dortmund, Dep. of Computer Science, 2005.

[LTK+00] C. Lindemann, A. Thümmler, A. Klemm, M. Lohmann, and O. Waldhorst. Quantitative System Evaluation with DSPNexpress 2000. In *Proc. 2nd Int. Workshop on Software and Performance (WOSP 2000)*, pages 12–17, Ottawa, Canada, 2000.

[LTK+02] C. Lindemann, A. Thümmler, A. Klemm, M. Lohmann, and O. Waldhorst. Performance Analysis of Time-enhanced UML Diagrams Based on Stochastic Processes. In *Proc. 3rd Int. Workshop on Software and Performance (WOSP 2002)*, pages 25–34, Rome, Italy, 2002.

[LW01] C. Lindemann and O. Waldhorst. Evaluating Cooperative Web Caching Protocols for Emerging Network Technologies. In *Proc. Workshop on Caching, Coherence and Consistency (WC3 '01)*, pages 1–9, Sorrento, Italy, 2001.

[LW02a] C. Lindemann and O. Waldhorst. A Distributed Search Service for Peer-to-Peer File Sharing in Mobile Applications. In *Proc. 2nd IEEE Conf. on Peer-to-Peer Computing (P2P 2002)*, pages 71–78, Linköping, Sweden, 2002.

[LW02b] C. Lindemann and O. Waldhorst. Evaluating the Impact of Different Document Types on the Performance of Web Cache Replacement Schemes. In *Proc. Int. Conf. on Dependable Systems and Networks (DSN 2002)*, pages 717–726, Washington, DC, 2002.

[LW03] C. Lindemann and O. Waldhorst. Consistency Mechanisms for a Distributed Lookup Service supporting Mobile Applications. In *Proc. 3rd Int. ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE 2003)*, pages 61–68, San Diego, CA, 2003.

[LW04a] C. Lindemann and O. Waldhorst. Exploiting Epidemic Data Dissemination for Consistent Lookup Operations in Mobile Applications. *ACM SigMobile Mobile Computing and Communication Review (MC2R) Special Issue on Mobile Data Management*, 8(3):44–56, 2004.

[LW04b] C. Lindemann and O. Waldhorst. *Passive Distributed Indexing (PDI) v1.0*, 2004. *http://mobicom.cs.uni-dortmund.de/pdi/draft-lindemann-waldhorst-pdi-00.html.*

[LW05a] C. Lindemann and O. Waldhorst. Epidemic Dissemination of Presence Information in Mobile Instant Messaging Systems. In *Proc. 14. GI / ITG Fachtagung Kommunikation in Verteilten Systemen (KiVS 2005)*, pages 29–40, Kaiserslautern, Germany, 2005.

[LW05b] C. Lindemann and O. Waldhorst. Modeling Epidemic Information Dissemination on Mobile Devices with Finite Buffers. In *Proc. ACM. Int. Conf. on Measurement & Modeling of Computer Systems (ACM SIGMETRICS 2005)*, pages 121–132, Banff, Canada, 2005.

[MAN] IETF Working Group Mobile Ad hoc Networks (MANET). *Mobile Ad-hoc Networks (MANET) Charter. http://www.ietf.org/html.charters/manet-charter.html.*

[Moc87] P. Mockapetris. Domain Names - Concepts and Facilities. Request for Comments 1034, Internet Engeneering Task Force, 1987.

[Nag04] P. Nagelschmidt. Entwicklung eines Verfahrens zur mobilen Suche in WWW-basierten Informationsportalen. Diplomarbeit, Universität Dortmund, 2004.

[Nap] Napster, LLC. *Napster Hompage. http://www.napster.com.*

[NKR+02] 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.

[NTCS99] S. Ni, Y. Tseng, Y. Chen, and J. Sheu. The Broadcast Strom Problem in a Mobile Ad Hoc Network. In *Proc. 5th ACM Int. Conf. on Mobile Computing & Networking (MobiCom 99)*, pages 151–162, Seatel, WA, 1999.

[OTB89] A. Ovchinnikov, S. Timashev, and A. Beleyy. *Kinetics of Diffusion Controlled Chemical Processes.* Nova Science Publishers, 1989.

[PRD03] C. Perkins, E. Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. Request for Comments 3561, Internet Engeneering Task Force, 2003. *ftp://ftp.rfc-editor.org/in-notes/rfc3561.txt.*

[PS01] M. Papadopouli and H. Schulzrinne. Effects of Power Conservation, Wireless Coverage and Cooperation on Data Dissemination among Mobile Devices. In *Proc. 2th ACM Int. Symp. on Mobile Ad Hoc Networking & Computing (MobiHoc 2001)*, pages 117–127, Long Beach, CA, 2001.

[PSW01] M. Parameswaran, A. Susarla, and A. Whinston. P2P Networking: An Information-Sharing Alternative. *IEEE Computer*, 34(7):31–38, 2001.

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proc. ACM SIGCOMM 2001*, pages 149–160, San Diego, CA, 2001.

[RIF02] M. Ripeanu, A. Imnitchi, and I. Foster. Mapping the Gnutella Network. *IEEE Internet Computing*, 6(1):50–57, 2002.

[SA04a] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. Request for Comments 3921, Internet Engeneering Task Force, 2004.

[SA04b] P. Saint-Andre. XMPP - Core Protocol. Request For Comments 3920, Internet Engeneering Task Force, 2004.

[SGM03] Q. Sun and H. Garcia-Molina. Partial Lookup Services. In *Proc. 23rd Int. Conf. On Distributed Computing Systems (ICDCS 2003)*, pages 58–67, Providence, RI, 2003.

[SGN03] R. Schollmeier, I. Gruber, and F. Niethammer. Protocol for Peer-to-Peer Networking in Mobile Environments. In *Proc. IEEE 12th Int. Conf. on Computer Communications and Networks (ICCCN'03)*, Dallas, TX, 2003.

[SH03] T. Small and Z. Haas. The Shared Wireless Infostation Model: A New Ad Hoc Networking Paradigm (or Where there is a Whale, there is a Way). In *Proc. 4th ACM Int. Symp. on Mobile Ad Hoc Networking & Computing (MobiHoc 2003)*, pages 233–244, Annapolis, MA, 2003.

[SMK+01] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proc. ACM SIGCOMM 2001*, pages 149–160, San Diego, CA, 2001.

[Sri01] K. Sripanidkulchai. The Popularity of Gnutella Queries and its Implications on Scalability. Featured on O'Reilly's www.openp2p.com website, 2001. *http://www-2.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html.*

[TCG05] G. Turi, M. Conti, and E. Gregori. A Cross Layer Optimization of Gnutella for Mobile Ad hoc Networks. In *Proc. 6th ACM Int. Symp. on Mobile Ad Hoc Networking & Computing (MobiHoc 2005)*, pages 343–354, Urbana-Champaign, IL, 2005.

[Thu03] A. Thuemmler. *Stochastic Modeling and Analysis of 3G Mobile Communication Systems.* PhD thesis, Universität Dortmund, 2003.

[TK05] S. Tewari and L. Kleinrock. Analysis of search and replication in unstructured peer-to-peer networks. Technical Report UCLA-CSD-TR050006, University of California, Los Angeles, 2005.

[TNS03] Y. Tseng, S. Ni, and E. Shih. Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multihop Mobile Ad Hoc Network. *IEEE Trans. on Computers*, 52(5):545–557, 2003.

[Tri02] K. Trivedi. *Probability and Statistics with Reliablility, Queuing, and Computer Science Applications.* John Wiley & Sons, New York, NY, 2nd edition, 2002.

[W3C03] World Wide Web Consortium. *XML Schema*, 2003.

[Wes05] C. Westphal. On Maximizing the Lifetime of Distributed Information in Ad-Hoc Networks with Individual Constraints. In *Proc. 6th ACM Int. Symp. on Mobile Ad Hoc Networking & Computing (MobiHoc 2005)*, pages 26–33, Urbana-Champaign, IL, 2005.

[XO02] Y. Xie and D. O'Hallaron. Locality in Search Engine Queries and Its Implications for Caching. In *Proc. IEEE Infocom 2002*, pages 1238–1247, New York, NY, 2002.

[YGK03] Y. Yi, M. Gerla, and T. Kwon. Efficient Flooding in Ad hoc Networks: a Comparative Performance Study. In *Proc. IEEE Int. Conf. on Communication (ICC 2003)*, pages 1059–1063, Anchorage, AK, 2003.

[YGM01] B. Yang and H. Garcia-Molina. Comparing Hybrid Peer-to-Peer Systems. In *Proc. 27th Int. Conf. on Very Large Data Bases (VLDB 2001)*, pages 561–570, Rome, Italy, 2001.

[Zip49] G. Zipf. *Human Behaviour and the Principle of Least-Effort.* Addison-Wesley, Cambridge, MA, 1949.

[ZLLY05] S. Zhong, L. Li, Y. Liu, and Y. Yang. On Designing Incentive-Compatible Routing and Forwarding Protocols in Wireless Ad-Hoc Networks. In *Proc. 11th ACM Int. Conf. on Mobile Computing & Networking (MobiCom 2005)*, Cologne, Germany, pages 117–131, 2005.