TKK Dissertations 259
Espoo 2010

DATA AVAILABILITY IN CHALLENGING NETWORKING ENVIRONMENTS IN PRESENCE OF FAILURES

Doctoral Dissertation

Mikko Pitkänen

Aalto University
School of Science and Technology
Faculty of Electronics, Communications and Automation
Department of Communications and Networking

TKK Dissertations 259
Espoo 2010

DATA AVAILABILITY IN CHALLENGING NETWORKING ENVIRONMENTS IN PRESENCE OF FAILURES

Doctoral Dissertation

Mikko Pitkänen

Doctoral dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Faculty of Electronics, Communications and Automation for public examination and debate in Auditorium S1 at the Aalto University School of Science and Technology (Espoo, Finland) on the 14th of January 2011 at 12 noon.

Aalto University
School of Science and Technology
Faculty of Electronics, Communications and Automation
Department of Communications and Networking

Aalto-yliopisto
Teknillinen korkeakoulu
Elektroniikan, tietoliikenteen ja automaation tiedekunta
Tietoliikenne- ja tietoverkkotekniikan laitos

Distribution:
Aalto University
School of Science and Technology
Faculty of Electronics, Communications and Automation
Department of Communications and Networking
P.O. Box 13000 (Otakaari 5)
FI - 00076 Aalto
FINLAND
URL: http://comnet.tkk.fi/
Tel. +358-9-470 22355
Fax +358-9-470 22474
E-mail: mikko.pitkanen@aalto.fi

ISBN 978-952-60-3507-9
ISBN 978-952-60-3508-6 (PDF)
ISSN 1795-2239
ISSN 1795-4584 (PDF)
URL: http://lib.tkk.fi/Diss/2011/isbn9789526035086/

TKK-DISS-2881

Aalto-Print
Helsinki 2010

Aalto University

ABSTRACT OF DOCTORAL DISSERTATION	AALTO UNIVERSITY SCHOOL OF SCIENCE AND TECHNOLOGY P.O. BOX 11000, FI-00076 AALTO http://www.aalto.fi
Author Mikko Juhani Pitkänen	
Name of the dissertation Data Availability in Challenging Networking Environments in Presence of Failures	
Manuscript submitted 18.06.2010	Manuscript revised 25.11.2010
Date of the defence 14.01.2011	
☐ Monograph	[✓ Article dissertation (summary + original articles)]

Faculty	Faculty of Electronics, Communications and Automation
Department	Department of Communications and Networking
Field of research	Networking Technology
Opponent(s)	Doctor Cecilia Mascolo
Supervisor	Professor Jörg Ott
Instructor	Professor Jörg Ott and Doctor Tapio Niemi

Abstract

This Doctoral thesis presents research on improving data availability in challenging networking environments where failures frequently occur. The thesis discusses the data retrieval and transfer mechanisms in challenging networks such as the Grid and the delay-tolerant networking (DTN). The Grid concept has gained adaptation as a solution to high-performance computing challenges that are faced in international research collaborations. Challenging networking is a novel research area in communications.

The first part of the thesis introduces the challenges of data availability in environment where resources are scarce. The focus is especially on the challenges faced in the Grid and in the challenging networking scenarios. A literature overview is given to explain the most important research findings and the state of the standardization work in the field.

The experimental part of the thesis consists of eight scientific publications and explains how they contribute to research in the field. Focus in on explaining how data transfer mechanisms have been improved from the application and networking layer points of views. Experimental methods for the Grid scenarios comprise of running a newly developed storage application on the existing research infrastructure. A network simulator is extended for the experimentation with challenging networking mechanisms in a network formed by mobile users. The simulator enables to investigate network behavior with a large number of nodes, and with conditions that are difficult to re-instantiate.

As a result, recommendations are given for data retrieval and transfer design for the Grid and mobile networks. These recommendations can guide both system architects and application developers in their work. In the case of the Grid research, the results give first indications on the applicability of the erasure correcting codes for data storage and retrieval with the existing Grid data storage tools. In the case of the challenging networks, the results show how an application-aware communication approach can be used to improve data retrieval and communications. Recommendations are presented to enable efficient transfer and management of data items that are large compared to available resources.

Keywords data availability, Grid, failure tolerance, challenged networks, delay-tolerant networking	
ISBN (printed) 978-952-60-3507-9	ISSN (printed) 1795-2239
ISBN (pdf) 978-952-60-3508-6	ISSN (pdf) 1795-4584
Language English	Number of pages 163 p. + app. 96 p.
Publisher Department of Communications and Networking	
Print distribution Department of Communications and Networking	
[✓ The dissertation can be read at http://lib.tkk.fi/Diss/2011/isbn9789526035086/]	

Aalto-yliopisto

VÄITÖSKIRJAN TIIVISTELMÄ	AALTO-YLIOPISTO TEKNILLINEN KORKEAKOULU PL 11000, 00076 AALTO http://www.aalto.fi

Tekijä Mikko Juhani Pitkänen

Väitöskirjan nimi

Tiedon saatavuus haasteellisissa tietoverkoissa vikatilanteisiin varautuen

Käsikirjoituksen päivämäärä 18.06.2010	Korjatun käsikirjoituksen päivämäärä 25.11.2010
Väitöstilaisuuden ajankohta 14.01.2011	
☐ Monografia	☒ Yhdistelmäväitöskirja (yhteenveto + erillisartikkelit)

Tiedekunta	Elektroniikan, tietoliikenteen ja automaation tiedekunta
Laitos	Tietoliikenne- ja tietoverkkotekniikan laitos
Tutkimusala	Tietoverkkotekniikka
Vastaväittäjä(t)	Tohtori Cecilia Mascolo
Työn valvoja	Professori Jörg Ott
Työn ohjaaja	Professori Jörg Ott ja Tohtori Tapio Niemi

Tiivistelmä

Tässä väitoskirjatyössä tutkitaan tiedon saatavuuteen liittyviä haasteita tietoverkoissa, joissa virhetilanteisiin tulee erityisesti varautua. Väitöskirja käsittelee näitä haasteita käyttäen esimerkkeinä hilaverkkojen (Grid) tallennusjärjestelmiä sekä viivesietoisten tietoverkkojen (DTN) tiedonhaku- ja siirtomenenetelmiä. Hilaverkot ovat laajalti käytetty teknologia kansainvälisten tiedeyhteisöjen harjoittamassa suurteholaskennassa. Haastelliset tietoverkot puolestaan ovat uusi tutkimusalue mobiiliviestinnän saralla.

Työn alkuosassa esitetään tiedon saatavuuden ongelma resurssien suhteen niukassa ympäristossä. Erityisesti keskitytään hilaverkkojen ja haastellisten tietoverkkojen käyttötapauksissa esiintyviin niukkojen ja epäluotettavien resurssien ongelmaan. Kirjallisuuskatsauksen keinoin selvitetään alan merkittävimmät tutkimustulokset ja annetaan katsaus tutkimuksen ja standardoinnin nykytilaan.

Työn kokeellisessa osuudessa käsitellään tähän työhön sisältyvät kahdeksan tieteellistä julkaisua, ja esitetään näihin liittyvää tutkimuksta. Erityisesti täsmennetään, miten tiedonsiirtoa on pystytty tehostamaan uusin ratkaisuin sekä käyttäjäsovelluksen että verkon (muodostaman systeemin) kannalta. Kokeellisen osuuden tutkimusvälineinä on käytetty hilaverkkojen tapauksessa olemassa olevaa tutkimusinfrastruktuuria ja tätä työtä varten tarkoituksellisesti kehitettyä tallennusohjelmistoa. Haasteellisten tietoverkkojen tutkimuksessa käytetään menetelmänä verkkosimulaatiota, jonka avulla voidaan tarkastella suurienkin verkkojen käyttäytymistä vaikeasti toistettavissa olosuhteissa.

Tutkimuksen perustella on laadittu suosituksia, joiden avulla tiedonsiirto- ja hallintamenetelmiä voidaan suunnitella sopiviksi hila- ja mobiiliverkkoympäristöihin. Nämä suositukset auttavat sekä järjestelmäsuunnittelijoita että sovelluskehittäjiä tulevaisuuden kehitystyössä. Hilaverkkojen tapauksessa tulokset antavat näyttöjä virheenkorjauskoodien soveltuvuudesta tiedon siirtämiseen ja tallentamiseen hilaverkkojen tietovarastoissa. Haastellisten tietoverkkojen parissa työn tulokset osoittavat, miten sovellustason tietoa voidaan käyttää tehostamaan verkkotason toimintaa. Tuloksena esitetään suosituksia verkon resursseihin nähden suurien tietomäärien tehokkaaseen siirtoon ja hakemiseen.

Asiasanat tiedon saatavuus, vikasietoisuus, hilaverkot, haasteelliset tietoverkot, viivesietoiset tietoverkot

ISBN (painettu) 978-952-60-3507-9	ISSN (painettu) 1795-2239
ISBN (pdf) 978-952-60-3508-6	ISSN (pdf) 1795-4584
Kieli englanti	Sivumäärä 163 s. + liitteet 96 s.

Julkaisija Tietoliikenne- ja tietoverkkotekniikan laitos

Painetun väitöskirjan jakelu Tietoliikenne- ja tietoverkkotekniikan laitos

☒ Luettavissa verkossa osoitteessa http://lib.tkk.fi/Diss/2011/isbn9789526035086/

Preface

This thesis and the related postgraduate studies have been carried as a part of my work in the Technology Programme in the Helsinki Institute of Physics and in the Department of Communications and Networking in the Aalto University. Academy of Finland has funded my research through the Grid for Scientists, DISTANCE, and RESMAN projects. Grants from the Valdemar von Frenckell's Foundation and the Nokia Foundation have financially supported my work.

First, I want to thank the supervisor of this thesis, Jörg Ott, for his guidance, feedback, and truly uncompromised enthusiasm throughout the work. Tapio Niemi gave me important advice on research work and kept me on track, especially during the first years of my postgraduate studies. Miika Tuisku enabled and supported my research visit in Geneva between 2005-2008. Ari-Pekka Hameri was an inspiring academic example during the work and ensured funding to complete this thesis. I would also like to thank my colleagues in the Helsinki Institute of Physics and in the Aalto University for a nice atmosphere.

Comments from my two pre-examiners, Vania Conan and Kevin Almeroth, helped me to give the final touch to the outline of this thesis. Several colleagues, Rim Moussa, Tapio Niemi, Martin Swany, Juho Karppinen, Ari Keränen, Teemu Kärkkäinen, Janico Greifenberg, and Jörg Ott have been co-authors for the publications presented in this thesis. It has truly been a pleasure to work with these hard-working, smart, and inspiring people.

Finally, my dear family and Noora deserve my whole-hearted gratitude for the support without which this would not have been fun.

Espoo, November 2010

Mikko Pitkänen

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Mikko Pitkänen, Rim Moussa, Marting Swany, and Tapio Niemi, "Erasure Codes for Increasing Availability of the Grid Data Storage", Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW 2006), ISBN 0-7695-2522-9, pages 185–194, February, 2006.

II Mikko Pitkänen, Juho Karppinen, and Martin Swany, "GridBlocks DISK - Distributed Inexpensive Storage with K-availability", Proceedings of the 15th International IEEE Symposium on High Performance Parallel and Distributed Computing (HDPC) Workshop on Next-Generation Distributed Data Management, pages 1–6, June, 2006.

III Jörg Ott and Mikko Pitkänen, "DTN-based Content Storage and Retrieval", Proceedings of the First WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC), ISBN 978-1-4244-0993-8, pages 1–7, June, 2007.

IV Mikko Pitkänen and Jörg Ott, "Redundancy and Distributed Caching in Mobile DTNs", MobiArch '07: Proceedings of 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture, ISBN 978-1-59593-784-8, pages 1–7, ACM, August, 2007.

V Mikko Pitkänen and Jörg Ott, "Enabling Opportunistic Storage for Mobile DTNs", Journal of Pervasive and Mobile Communications, vol 4, no. 5, pages 579–594, Elsevier, October, 2008.

VI Mikko Pitkänen, Ari Keränen, and Jörg Ott, "Message Fragmentation in Opportunistic DTNs", Proceedings of the Second WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC), ISBN 978-1-4244-2099-5, pages 1–7, June, 2008.

VII Mikko Pitkänen, Teemu Kärkkäinen, Janico Greifenberg, and Jörg Ott, "Searching for Content in Mobile DTNs", Proceedings of the 7th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom), ISBN 978-1-4244-3304-9, pages 1–10, March, 2009.

VIII Mikko Pitkänen, Teemu Kärkkäinen, and Jörg Ott, "Opportunistic Web Access via WLAN Hotspots", Proceedings of the 8th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom), ISBN 978-1-4244-5329-0, pages 20–30, March, 2010.

List of Abbreviations

AAP	Application-aware Processing
API	Application Programming Interface
ARC	Advanced Resource Connector
CAN	Content Addressable Network
DRS	Data Replication Service
DRS	Data Replication Service
DTN	Delay–tolerant Networking
DTNRG	Delay–tolerant Networking Research Group
EDR	Enhanced Data Rate
EID	Endpoint Identifier
GCP	Grid Copy
GSI	Globus Security Infrastructure
GPS	Global Positioning System
FC	First Contact
FE	Front-end
FEC	Forward Error Correcting
FIFO	First In First Out
HCS	Helsinki City Scenario
HEP	High Energy Physics
ICMP	Internet Control Message Protocol
IAB	Internet Architecture Board
IDA	Information Dispersal Algorithm
IETF	Internet Engineering Task Force
IRTF	Internet Research Task Force
IP	Internet Protocol

LAN	Local Area Network
LDPC	Low-Density Parity-Check
LFN	Logical File Name
LRC	Local Replica Catalog
MANET	Mobile Ad-hoc
MIME	Multipurpose Internet Mail Extensions
MTU	Maximum Transfer Unit
NAPTR	Name Authority Pointer
PFN	Physical File Name
PSN	Pocket Switched Network
RAID	Redundant Array of Inexpensive Disks
RFC	Request for Comments
RG	Research Group
RFT	Reliable File Transfer
RLI	Replica Location Index
RLS	Replica Location Service
RS	Reed Solomon
RWP	Random Way Point
SE	Storage Element
URI	Uniform Resource Identifier
S/MIME	Secure Multipurpose Internet Mail Extensions
SSO	Single-sing-on
SSP	Scheme-specific Part
TCP	Transmission Control Protocol
TTL	Time to Live
VANET	Vehicular Ad Hoc Network
WAN	Wide Area Network
WLAN	Wireless Local Area Network
WG	Working Group
WSRF	Web Service Resource Framework

1 Introduction

Accessing digital content in the Internet is one of the biggest trends in personal information management. Web users frequently consume and create data, such as digital pictures and web pages, by using their computers and mobile devices. The applications have evolved so that they commonly access the Internet to provide a richer set of services to users. The modern mobile applications (for example, Flickr, Facebook, RSS readers), are geared towards sharing and consuming data in the web, which makes the Internet even more important source of data than the local resources on the user's device. This means that the applications are increasingly dependent on the reliability of the network services and the availability of the data stored in the services.

At the same time the Internet services experience growth pains as they must reliably serve increasing amounts of data for a growing number of users. An often-used medication is to build a distributed system, which shares the work of serving the users between multiple service elements. However, when the number of required elements increases, it becomes more likely that some of them are not instantly available [63]. Unavailability of the elements can be caused, for example, due to a temporal maintenance break, network outage, or a more permanent failure. Thus, building a distributed system requires mechanisms to cope with both the expected and unexpected failures of the infrastructure elements. Large storage systems used by the Internet search engine [56] or cloud computing providers [103] often use duplication to protect against failures. In these scenarios, the increase in usage usually leads to an increase in income to the service providers, for example, through online advertisement, and thus the amount of storage resources can be increased. However, there are many scenarios where the available budget does not increase in proportion to amount of stored data. In these settings, for example, in scientific data preservation [64], redundancy mechanisms that can ensure data availability with minimum amount of storage space and minimal cost are needed.

This thesis investigates mechanisms to support information access in the presence of failures and especially in scenarios where hardware is a scarce resource because of a limited budget. The thesis starts by discussing mechanisms to cope with failures of the storage elements in the distributed data storage infrastructures. Then, the discussion continues to cover even more challenging settings, where all network components, including nodes and links between them, are considered unreliable. Throughout the thesis, two mechanisms have important roles. First, increasing *redundancy* of data entries enables using multiple data sources to provide the stored data when part of the service infrastructure is not available. Second, allowing nodes that are in the proximity of a client to opportunistically serve data requests increases *locality* of data access. Thus, in the scope of this thesis increasing locality decreases dependency on accessing data from pre-determined and possibly distant locations. Locality is beneficial especially in challenging scenarios where large part of the infrastructure can become unreachable and content copies in the proximity of the requesting node may be the only reachable sources for the requested data. This thesis investigates the first mechanism in large storage infrastructures where data is spread over multiple storage elements, which can become partially unavailable over time. The second mechanism is beneficial in even more challenging scenarios where the network components, including links, are not constantly available and thus the service infrastructure is only partially connected. The latter scenario is characterized by high intensity of failures and high scarcity of resources, thus the proposed mechanisms cannot rely on any assumptions about resource availability. Instead, they must be able to benefit even from individual resources that are opportunistically available and can be reached. Opportunistic network formed between mobile phones carried by human users is an example of such scenarios.

Practical use cases for the above mechanisms can be found in several areas in the real world. Large information archives distribute data over multiple data storage elements in distinct locations to increase reliability and access performance of the storage [3]. A replication–based approach is commonly used, for example, in large scientific data storages [95]. Moreover, locality is used in large-scale content distribution systems [97]. For example, when a user in Europe is watching a popular sports event from another continent, a nearby copy in Europe serves as a data source rather than accessing a server in the other continent. Similar challenges happen, for example, in rural conditions, where the communication is possible only with the nodes in the local proximity [34] and a close by data copy can be the only available source to download the data.

The redundancy and locality concepts, which are discussed in this thesis are not entirely new in the distributed systems domain. Intelligent and space efficient redundancy schemes, for example, a RAID disk array [106] or information dispersal algorithm [115] have been proposed already in the 1980s and efficient caching mechanisms for the Internet content in the 1990s [14]. Furthermore, replication and locality are used to serve hundreds of billions web requests every year in the Internet [97]. Cooperation between the Internet users has enabled large content access infrastructures, for example, BitTorrent [28], which use striping of files. Moreover, recent research has proposed re-architecting the Internet architecture [83, 35] to support data-oriented operation and using redirection to near by data items, but the work mainly focuses on fixed network settings. This thesis, on the other hand, investigates data access also for mobile users that have limited network connectivity. While the way the networks are used and deployed evolves constantly, adapting the network mechanisms to these conditions continues to offer many venues for research. This thesis covers some of them and focuses especially to scenarios where the resources are scarce and failures occur frequently.

1.1 Problem Statement

The Internet architecture is designed to support end-to-end communication between nodes that are well connected by reliable network links. This model is suitable, for example, to a computer administrator whose task is to operate a mainframe computer through a remote terminal session. A communication session between the specific endpoints is established and rapid interaction is required. However, in this model communication happens inside an end-to-end session that is unique per user. Sharing of resources is difficult because the users individually connect to a predetermined service and are served inside a dedicated sessions per user. Furthermore, when used for data access, the end-to-end session is established between the client and a particular location where a copy of the data exists. This makes the data access dependent on the availability of the particular storage location and the connection used to reach it. Any failure in the path or end-points can render the data access impossible.

The dependency on a path to a predetermined location can prevent reaching a particular resource and is thus especially bad in challenging network conditions where failures frequently occur. This thesis investigates how to efficiently use the network and storage resources to enable information retrieval in the presence of failures. The discussion focuses on scenarios where data is once written and later read multiple times by the different users. This is a common use case in information retrieval, and helps to limit research problem because questions related to, for example, simultaneous updates and versioning can be limited out of scope. Thus, research problem in this thesis can be formulated by the following two research questions:

How to store data items reliably when storage elements are likely to fail in the system? This thesis investigates mechanisms for managing the redundancy of data to enable data retrieval even when part of the system is unavailable. In particular, mechanisms are investigated that require smaller amount of storage space than the traditional approach that uses replication. The applicability of the proposed methods is evaluated in two example storage infrastructures, which are well connected by modern networking infrastructure.

How to access data when connectivity of the network infrastructure is challenged? Challenged connectivity can be caused, for example, by network partitions or large round trip times to distant locations. This thesis investigates methods for operating with data so that the operations can be moved in the network in a semantically self-contained manner. This enables the storage applications and the supporting infrastructure to benefit from the self-contained nature of the data access operations. Then, the ability of network nodes to interpret the data access operations is used to control redundancy of data items in the network. Fragmentation is introduced to overcome limited transfer capacity of the contacts forming the network and to enable transferring large data items over scarce communication resources. Furthermore, mechanisms that enable close by nodes to serve data based on search terms are investigated. Finally, using partial infrastructure support for web access and using location awareness to improve response delivery are discussed.

These two questions share a common goal, since both investigate scenarios in which part of the storage resources are unreachable at the time of data access. In the first case, reliable network infrastructure enables to reach the majority of the infrastructure elements, and it is sufficient to cope with failures or unreachability of the individual storage elements, which are available according to a predictable failure model. The second case can be considered as an extension to this as the failure of the network itself can partition the storage infrastructure, and assumptions about the properties of individual storage nodes are not enough to ensure availability of the stored data. Instead, the near by data items are used to provide the access to sought resources.

1.2 Context and Methodology

This thesis takes a systems approach to investigate the efficiency of data retrieval and transfer in the challenging environments. Two challenging networking architectures are introduced to motivate the challenges in data access, and an overview on the current literature in the field is provided. The first architecture is the Grid [53], which is used in well-connected distributed systems. The Grid storages often store large data items over multiple storage elements, and thus chances increase that some of the elements fail during the operation of the data storage [63]. The second architecture is the delay-tolerant networking (DTN) architecture [20], which is designed for challenging conditions where the network nodes and links are not reliable.

The Grid and the delay-tolerant networking architectures discussed in this thesis are different in many ways. An obvious distinction is the higher performance and better connectivity that the elements in the Grid infrastructure often posses. However, these architectures also share several characteristics that are relevant to mechanisms investigated in the following sections. First, resources are scarce, either limited by budget in the Grid or unreliable conditions in the DTN scenarios. Second, both architectures can benefit from a write once and read many approach, which is applicable approach for data sharing and distribution in other systems too. Third, operating on self-contained data lends itself for decentralized operation, since data can be accessed independent of indices and operations can be executed on independent data items. Thus, the proposed mechanisms can be applicable also in other types of scenarios where these characteristics hold. An example of such a scenario is a green computing architecture where storage unavailability can be caused, for example, by a scheduled site downtime due to high electricity usage during daytime heat. Similarly, access to close by data replicas of data can save electricity in the transmission when data does not have to be transferred between distant locations.

The first experiments, presented in Publications I and II, use a distributed storage application that is designed and implemented as part of the research conducted for this thesis. The application is adapted to *the Grid*, which has been defined by Foster et *al.* [52] as a system that: *"...coordinates resources that are not subject to centralized control using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service"*. The scientific computing community has been one of the most influential user groups of the Grid since the beginning. Since many scientific applications either consume or produce large amounts of data, data management has become an important topic in the Grid architecture research and development [24, 95]. This thesis uses the Grid data storage infrastructure as an environment to evaluate a storage application design that uses erasure coding to increase the fault tolerance of the distributed storage. The storage application is implemented in software and its performance is assessed in two distinct real world testbeds. The application is first installed on multiple geographic locations to establish a testbed for performance evaluations. Then, the storage application is extended to support simultaneous access for multiple clients and a testbed is formed in a local area network. The testbed approach enables to evaluate performance in a realistic, but uncontrolled environment that works on top of the Internet. Furthermore, the used settings allow a systems view on operation in presence of real hardware and network conditions.

Publications III to VIII use the delay-tolerant networking architecture as a platform for investigating how to improve efficiency of content retrieval when the network is unreliable and an end-to-end path between the communicating parties does not exist or cannot be considered reliable. The DTN architecture is researched in the Internet Research Task Force's (IRTF) DTN research group (RG). The DTN model makes opportunistic communication possible in cases where the IP-based end-to-end communication would fail [20]. The DTN nodes communicate by using *bundles* that are forwarded on hop-by-hop basis. This work leverages ability of the bundles to carry messages that are by themselves meaningful for applications and adds a small amount of application information to the bundles to allow intermediate nodes to cooperate for data storage operations. The benefits of the proposed approach are evaluated by using network simulators, for which the necessary storage and protocol extensions are designed and implemented. The simulations are used as the main method, because of difficulty of performing repeatable and large-scale experiments with human users carrying mobile devices. The complexity of the investigated scenarios makes it also impossible to capture numerous factors affecting the system performance by using analytical models. Moreover, human mobility patterns from real world [44] have been used together with state-of-the-art mobility models [45] to make the simulations to model the real world human behavior as closely as possible. The simulations are also used to investigate how to efficiently fragment large application data units, how to support information retrieval based on search terms, and how to leverage partial infrastructure support for web access. In addition, implementing parts of the system as software that runs on existing modern smartphones, as discussed in Publication VII and in Publication VIII, has helped to gain realistic understanding of the the system properties that were then used as an input to the simulation models.

This thesis proposes methods for increasing data access reliability that are novel in the scope of the Grid and the DTN architectures. Following the systems approach, a realistic model of the complete system is built to evaluate the feasibility of the proposed mechanism. The model is then evaluated in a network test-bed and by using network simulation tools. Finally, the evaluation results are analyzed and the findings are discussed in relation to existing work in the field to gain understanding on the effects of the proposed mechanisms in a system level.

1.3 Contributions

This thesis consists of eight peer-reviewed scientific publications. The following summarizes their contribution to the state-of-the-art in the field.

Publication I presents a novel design for a storage application that uses erasure coding to distribute files to storage services in the Grid. The presented approach overcomes high storage space requirements that are traditionally experienced in replication based distributed storage infrastructures. Moreover, it enables efficient data access even when data is partly accessed from storage locations with poor data delivery capacity. The applicability of the design is illustrated by performing experimental evaluation in a real world testbed, which combines several existing Grid storages in Europe and the United States. Publication II continues this work by adapting the presented storage architecture for a local area networking environment. The publication presents design and performance evaluation for parallel data access to illustrate how small communication latency in a local network can be leveraged to increase performance of the proposed design.

Publication III discusses how cooperative sharing of communication and storage resources can make information retrieval more efficient in opportunistic networks. The publication formalizes the retrieval and storage operations for self-contained information bundles in DTNs, and gives preliminary results on feasibility of using the operations. Publication IV extends the work by introducing protocol support for the caching and storage operations, and outlines application–level erasure coding for the bundles. The publication uses increasingly realistic scenarios to evaluate the applicability of the design in a network consisting of mobile users. Publication V combines the findings from the publications III and IV and extends the performance evaluation to more realistic scenarios, which model user behavior in a realistic manner. The publication applies a realistic content request model and illustrates the liveliness properties of cached resources in an opportunistic mobile network that is modeled according to real human mobility.

Publication VI investigates how fragmentation of bundles can be used to overcome limited transfer capacity of opportunistic contacts in DTNs. The publication explains different fragmentation models and evaluates their performance together with several well-known DTN routing protocols. The publication also discusses harmful effects of fragmentation and recommends that the messages should be forwarded without fragmentation when possible. Moreover, the publication evaluates and discusses the benefits of using predefined fragmentation boundaries for the bundles.

Publication VII discusses how to retrieve content by using search terms in unstructured networks and investigates the effect of various query termination mechanisms in opportunistic networks that use DTN for communications. Publication VIII investigates how to leverage partial infrastructure support via wireless local area networks (WLANs) for opportunistic web access in an urban area. The publication outlines the applicability of DTN mechanisms in presence of the partial infrastructure support and shows how partial information about the network structure can be used to control redundancy and leverage locality for response routing.

1.4 Author's Contributions

Publication I is a joint work of the authors. The publication was written and the research problem formulated jointly together with Tapio Niemi, Rim Moussa, and Martin Swany. The present author carried out the evaluation part.

Publication II is a joint work of the authors. The publication was written and the research problem formulated jointly together with Juho Karppinen and Martin Swany. Evaluations were carried out jointly with Juho Karppinen, who also participated on their design.

Publication III is a joint work of the authors. The publication was written and the research problem formulated jointly together with Jörg Ott. The present author conducted the experimentations.

Publication IV is a joint work of the authors. The publication was written and the research problem formulated jointly together with Jörg Ott. The present author conducted the evaluations.

Publication V is a joint work of the authors. The publication was written and the research problem formulated jointly together with Jörg Ott. The present author conducted the experiments.

Publication VI is a joint work of the authors. The publication was written and the research problem formulated jointly together with Jörg Ott. Evaluation part was carried out jointly with Ari Keränen, who also participated on the design of the experiments.

Publication VII is a joint work of the authors. The publication was written and the research problem formulated jointly together with Jörg Ott, Teemu Kärkkäinen, and Janico Greifenberg. Evaluation part was carried out jointly with Mr. Kärkkäinen, who also participated on the design of the experiments.

Publication VIII is a joint work of the authors. The publication was written and the research problem formulated jointly together with Jörg Ott and Teemu Kärkkäinen. Evaluation part was carried out jointly with Mr. Kärkkäinen, who also participated on the design of the experiments.

1.5 Structure of the Thesis

This thesis is structured as follows. Section 2 discusses data access in storage infrastructures in which the individual storage elements are the main source for failures. The Grid data storage infrastructure is used as an example of such environment and a redundancy scheme is presented that can protect against failures with minimal storage space requirements. The performance of the proposed scheme is evaluated in two distributed network testbeds. Section 3 discusses data access in presence of even more challenging failures, which can cause the network to fail and become partitioned. The delay-tolerant networking architecture is used for enabling communication in such conditions, and several mechanisms are presented to improve data access in the presence of failures. A large set of evaluation results from measurement and simulations are shown to illustrate the applicability of the proposed mechanisms Section 4 concludes the results of this thesis with a discussion and gives hints on possible future research directions. Finally, eight peer-reviewed scientific publications are listed that present the results obtained during the course of the work conducted for this thesis.

2 Coping with Storages Element Failures

This section discusses methods for ensuring high availability of data in presence of failing infrastructure elements in a distributed data storage infrastructure. The discussion begins with motivating an access scenario to a distributed storage and then continues by an overview on erasure codes and their usage as a method to protect against storage element failures. Then, the system level functionality of a modern distributed data storage is discussed by using the replication based Grid infrastructure as an example. To overcome the high storage space requirements of the replication, a coding scheme is presented that can provide the level of redundancy required in the operational environment. To illustrate the applicability of the presented scheme, a storage application is designed, implemented in software, and its performance is evaluated in real computing hardware and network testbeds. The results of the evaluation illustrate that the approach is applicable in both geographically and locally distributed storage infrastructures. In case of the geographically distributed storage infrastructure, the evaluation shows that the presented scheme works efficiently in a setting where connection to some of the storage elements have low transfer capacity, for example, due to distant locations. In case of the locally distributed setting, the evaluation shows that the storage entities performing computing intensive coding tasks and network intensive access tasks can execute in parallel in a setting where resources are shared between adjacent nodes. The work presented in this section summarizes the results obtained in publications I and II. The publications contain further details about the design, the used experimental set-up, and the results.

2.1 Background

Creating multiple copies of data, in other words, *replicating* the data, is a widely used mechanism to secure against data loss [56, 24]. The mechanism creates verbatim copies of the file, called *replicas* that can be distributed to different locations in the network. If some of the locations become unavailable, a copy of data can be accessed from an alternative location. Replication has a desirable property that accessing data is straightforward, because the access does not require any pre-processing, for example, decoding. However, the increased processing power of modern computers motivates to investigate how computational methods, such as erasure coding, can be used to overcome the high storage space requirement faced with the replication based approach. The benefit of using erasure codes is that they can provide redundancy with a smaller amount of storage space when some extra processing is allowed. In other words, using erasure codes enables higher fault tolerance guarantees than would be possible by replication when the amount of storage space remains the same. The challenge of limited storage resources is currently faced, for example, in the large-scale scientific data analysis tasks where the amount of data grows more rapidly than the budget for the supporting infrastructure [64].

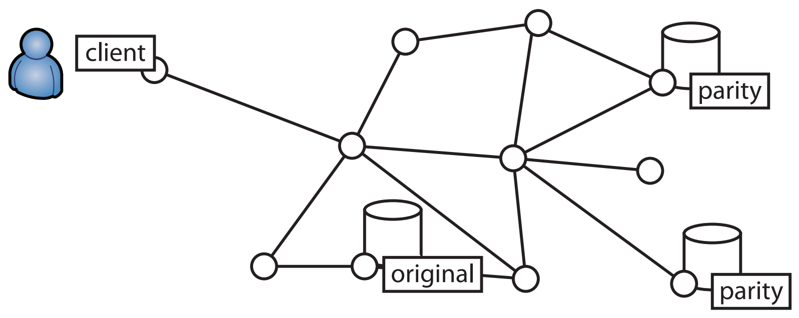

Figure 2.1: Distributed data storage with multiple download sources.

Figure 2.1 illustrates a network view on data access from a distributed storage. Several download sources exists for a file, or for the redundancy information (parity) related to the file. With the replication approach the client must be able to retrieve all the information related to the file from one of the multiple locations holding a replica. With the coding based approach, the client must retrieve fractions of data that constitute the file from multiple locations while parity information can be used to replace the missing parts of the data. Any of the storage elements can become unavailable between storing the file and retrieving it, that is, during the storage lifetime of the file.

In addition to saving storage space, a redundancy mechanism that can reconstruct a file from minimal amount of information can enable fault tolerant access to file so that only a small amount of download capacity is needed. This is beneficial especially close to the client where the downloads share the capacity of the same links. This is known as the *last-mile-problem* [143]. Similar challenges imposed by limited bandwidth are also discussed by two problem formulations in wide are file access. First, the Livny problem [5] discusses on how to optimally download a file that is divided into pieces and stored into several locations. Second, the Plank-Beck problem [29] describes how the access performance is measured as client's performance to retrieve a parts of a file from a wide-area filesystem. Thus, the approach presented in this section also discusses a possible solution to the above two problems.

Redundancy Mechanisms for Data Storage

Traditional mechanisms to secure against data loss create multiple copies of a datum and store each copy in a separate storage location. In case a datum is unavailable in one place, another instance of the stored datum can be accessed from another storage location. While simple, this approach uses a lot of storage space. To avoid excess storage space usage, erasure coding can be used. An erasure coding storage system can survive loss of data with theoretical minimum storage space requirements [114]. This section gives an introduction to using erasure coding for data storage and compares erasure coding to replication.

The widely used RAID disk array [106] technology is an example of system that uses coding to provide failure resistance in the disk based data storage systems. However, the RAID disk array uses a simple coding that enables surviving only from limited number of simultaneous failures. This property makes the used coding scheme less suitable for geographically distributed storages, which have higher fault detection times and maintenance cycles than in local installations [63]. In a geographically distributed storage, multiple failures can occur near in time and a mechanism is desirable that can cope with a configurable number of failures.

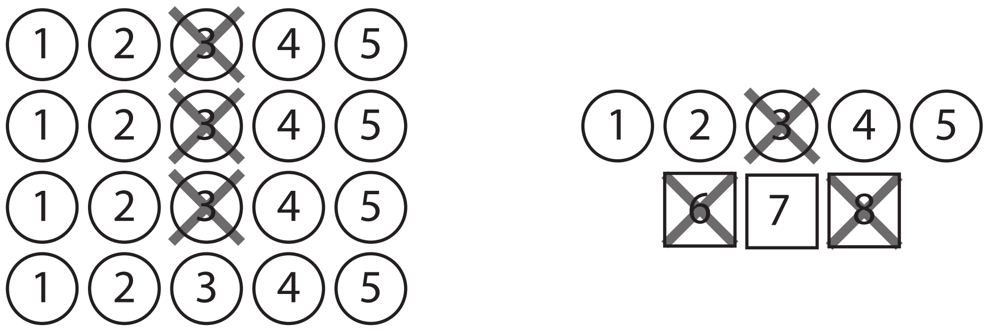

Figure 2.2: Replicated (left) vs. erasure coded (right) redundancy.

Figure 2.2 illustrates the space requirement of two well known redundancy schemes. Replication is shown on the left and erasure coding on the right. The figure shows a file divided into $m = 5$ parts, which are called *stripes* in the context of the distributed data storages (making file size divisible by m may require padding). Erasure coding can be used to create n data items out of m data items in a way that *any* m out of n are sufficient to recover the original information. This enables to secure data against loss of $k = n - m$ data items. A system with this property is called *k-available*. The left-hand side of the figure shows how replication requires $k + 1$ times the size of the original data to achieve k-availability. On the right, erasure coding requires only n times the size of a stripe (that is, $file/m$) of the original data. The decrease in the required storage space results from the property that the *parity stripes* created in erasure coding can be used to replace *any* lost stripe. This is different from the replication approach, where each stripe can only replace the loss of the byte-range it contains. The storage design presented in this thesis uses Reed Solomon (RS) erasure coding. The RS coding provides k-availability with the smallest possible storage space requirement.

The RS coding performs well when the values of coding parameters n and m are fairly small [112], as in the scenarios in this section. Moreover, the RS coding is a *systematic code*, which implicates that m stripes created from the original file form a verbatim copy of the input file. This means that with a systematic code decoding is unnecessary when there are no lost stripes. This enables partial file access and very fast reconstruction if no data is lost. In addition, the property that n and m can be freely chosen, allows to use coding to achieve desired level of redundancy in different storage environments. The level can be determined according to a simple failure model, for example, based on an assumption that the device lifetime is an exponential random variable [63]. Alternatively, statistics collected from large disk drive populations [110] can provide insight on how much redundancy is needed. The statistics also confirmed that developing an accurate failure models is very difficult even when modern storage monitoring tools are used. However, the system presented in this section can be configured to different operational environments, with varying properties such as failure models, failure detection, and repair times.

This thesis presents a storage system that uses erasure coding with striping of files to provide high data redundancy. The system allows recovering the data of a stored file even if some of the stripes of a file are unavailable. Stripes of data can be stored in larger number of locations than with the replication based approach, because coding enables creating an increased number of stripes (which can be used with any combination of other stripes) with similar storage space usage. When the stripes of data are later accessed, there is a larger number of sources to select from which to download. The cost of using erasure coding for storage redundancy arises from the computational costs related to coding and decoding. The evaluation part of this thesis shows how this cost is low in terms of time requirement when compared to the cost of downloads.

Earlier research on erasure codes [113, 94, 131] has produced novel coding methods, also referred to as forward error correction (FEC) codes, which can efficiently handle multiple simultaneous failures. These technologies have been shown to work for uses such as distributed storage in local networks [94] or securing against data loss in communications [18, 119]. However, the applicability of real implementations of software based erasure codes in geographically distributed storages had not been widely tested yet at the time of the publications I and II in 2006. During the course of this thesis work, the applicability of the presented approach was confirmed, for example, by the Cleversafe [70] company that sells storage solutions based on similar mechanisms. Another example of viability of using the coding based approach is Wuala [71] storage, which uses similar redundancy mechanism for storing data on the resources provided by the Internet users. In the latter case, a benefit of the approach compared to the other Internet storages, for example, to striping and replication based BitTorrent, is that with the coding it is more affordable to provide predetermined availability guarantees.

2.2 Related Research

Data Management in the Grid Using Replication

The Grid architecture is used as a platform to evaluate the feasibility of the mechanisms that are proposed in this chapter. The concept of the computational Grid was defined by Foster et *al.* [53] as a distributed computing infrastructure for advanced science and engineering. The design of the Grid data management infrastructure has been largely guided by the large data processing needs of the users from scientific community. Motivation on the Grid data management infrastructure have been discussed in several publications, for example, [15, 95, 22]. In a typical use case, several institutes form a *virtual organization* [69] to share resources amongst institutions that participate into cooperation, for example, in a physics experiment. These institutes are often geographically in distinct locations, thus a distributed solution is needed.

Storage resources in the Grid employ standardized access protocols and security mechanisms to enable secure access to the shared data. Storage elements (SEs) are services that expose a GridFTP interface to the Internet for data transfer and access. GridFTP extends the functionality of the Internet file transfer protocol (FTP) by providing extensions for high-performance operation and security [3]. These extensions include, for example, single–sign–on security, parallel data transfer using multiple data channels, re–startable transfers, and partial file transfer. Later additions to GridFTP [15] include pipelining to enable multiple simultaneous transfer requests, and operation over UDP-based data transfer (UDT). Striped data transfer [4] allows data to be spread across multiple servers in both ends of the transfer and it has been shown to achieve very fast transfers with performance in the scale of tens of Gbits/s. Reliable file transfer (RFT) can be used to control the transfers on behalf of the user [96].

In the Grid storage system, several copies of each data item are created and placed close to computing resources where they are likely to be analyzed. This is done to optimize the efficiency of accessing data and to increase fault tolerance in the distributed environment [117, 19]. The copies of a data item are called *replicas* in the context of the Grid. Each data item is identified by a logical file name (LFN). which is an unique identifier for a data content or a resource. A physical file name (PFN) presents a physical occurrence, that is, a replica, of the data content. The Grid data management infrastructure uses replica location service (RLS) to manage mappings from a LFN to the PFN(s) [24]. The Data Replication Service (DRS) maintains a replica of each specified data item available at predefined locations [23].

Coding Techniques and Storage Systems

Reed-Solomon coding is a well–known erasure coding technique. Rizzo [119] showed how RS coding can be efficiently implemented on software to secure against lost packets during the data transfers. Rabin [115] proposed Information Dispersal Algorithm (IDA) which can be used for similar purposes, that is, to secure against loss of data during storage or transmission. Some novel coding techniques have faster (de)coding performance than traditional codes, but not all combinations of stripes can be used for decoding. This is a drawback when used for storage applications [30], since download scheduling needs to take into account also the availability of the other stripes that can be used for decoding. A well known class of such coding techniques is called low-density parity-check (LDPC) codes. The LDPC codes have also a slightly higher storage space requirements than the optimal. For a good overview on erasure coding and storage applications, see, for example, work by Plank [111].

Several peer-to-peer storage systems have been proposed for efficient wide area storage on top on the Internet. Some of the approaches propose content addressable storage [137, 118]. In these systems the location of stored data is defined by its contents. Addressing the storage by its contents, results in efficient and automatic load-balancing and data is automatically re-located when storage nodes join and leave. A different type of storage and content sharing mechanism is presented by the widely deployed BitTorrent [28] application. BitTorrent users download data from multiple peers and allow the other peers to consecutively download the shared data. The wide adaptation of BitTorrent illustrates how people are willing to cooperate to achieve efficient storage and content sharing mechanism in the Internet. To overcome the reliability challenges in the distributed Internet storage, the authors of OceanStore [84] have proposed erasure coding to enable high-availability of the stored data with small space requirement. The OceanStore approach proposes the erasure coding to be used at the archival layer of the data storage, where data is not frequently accessed but can remain in storage for long durations.

Other work on erasure coding [17, 119] has suggested to use coding to remove need for interactive communications during data transmission. The work has especially suggested coding to avoid frequent acknowledgments in the unreliable communication channels. Modern coding theory has also proposed several novel LDPC codes [112] (for example, Raptor [131] and LT [131]) and development of new codes is an active area in research. Network coding [54] has been proposed to perform coding also in the intermediate nodes of the network. Kamra et *al.* [77] have presented Growth Codes which can be used to increase the amount of data that reaches a sink in zero-configuration sensor networks. Growth codes also allow recovering partially transferred data, the utility of which is highly application-dependent. Dimakis et *al.* [40, 39] have shown how to generate new coded fragments in a bandwidth-efficient manner to replace failed fragments. This is achieved by regenerating codes which require less bandwidth for storage maintenance than the previous approaches.

2.3 Fault Tolerant Storage System Based on Erasure Coding

This section presents a design for a data storage application. The design applies erasure coding and striping in the Grid storages in a novel way and it is capable of working with arbitrarily large files. The section continues with discussion on distribution of the files to network storages and on how to manage metadata of the distributed files. Then, the applicability of the presented approach is investigated in two different scenarios. First, a wide area network employment is used as a example of a scenario where communication latencies are large. Second, a local area network employment is used for evaluating the benefits of efficient local communications for improving performance of the storage application.

Figure 2.3 illustrates how file striping and erasure coding are applied to provide high redundancy of files. In the process of striping, the file is read block-by-block. Afterwards, each block is encoded from m packets to n packets and these packets are each written

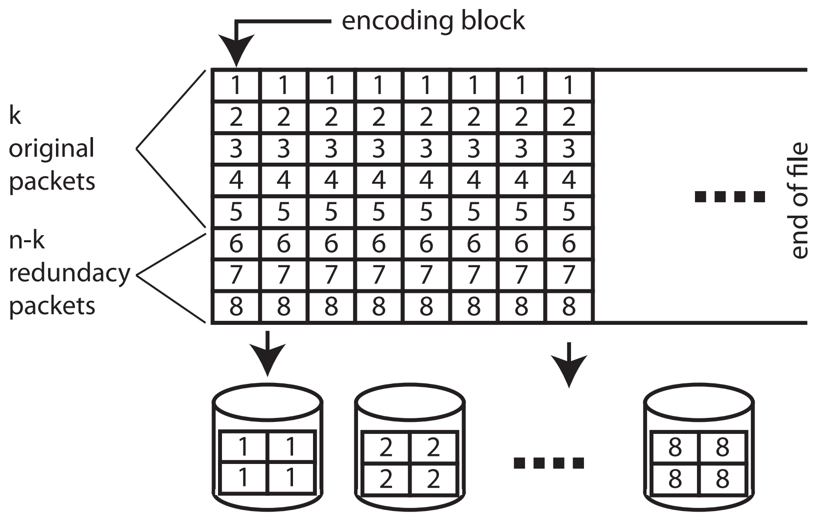

Figure 2.3: Striping and erasure coding scheme in a coding based storage application.

into a separate file, each file presenting a stripe of a file. In this striping approach one file is read as an input and n files are produced as output. After coding, the stripes can be handled as traditional files. This allows for easy operation with different transport protocols and storage resource providers. The implementation of the design discussed in this section uses Reed-Solomon coding from an open source Java project by OnionNetworks, which has derived the code from a work by by Luigi Rizzo [119]. The coding and the striping scheme is openly available as a part of the GridBlocks project in the SourceForge open source repository [37].

The presented approach requires only packets of a single block to be maintained in memory at any given time. The packet size defines how much data is transferred from files to memory (I/O operations) before computing (CPU operations) is started for the encoding, and vice-versa in the reconstruction phase. Thus, the choice of the packet size defines how much data is transferred for each computational step. Choosing a good packet size, helps to improve coding performance if both data and computing operations can be done in full speed once they take place. While the optimal packet size is dependent on the host system, packet sizes between $10 - 40$ KB were shown to have good performance on Linux systems, see Publication I for further details on the evaluation of different coding parameters.

```
#FEC Properties for : test.file
#Thu Aug 25 15:29:28 CEST 2005
m=5
n=9
packetSize=20000
fileLength=104857600
test.file.0.tmp=db-ibm.fri.utc.sk
test.file.1.tmp=pchip69.cern.ch
test.file.2.tmp=pc15.hip.fi
test.file.3.tmp=stout.pc.cis.udel.edu
test.file.4.tmp=n2grid.pri.univie.ac.at
test.file.5.tmp=moonshine.pc.cis.udel.edu
test.file.6.tmp=pchip11.hip.fi
test.file.7.tmp=db-ibm.fri.utc.sk
test.file.8.tmp=pchip69.cern.ch
OPTION: CHECKSUM INFORMATION
```

Figure 2.4: Description of a file distribution with erasure coding.

In addition to the contents of a stored file, some metadata needs to be maintained to reassemble a file from stripes stored in remote network storage elements. Figure 2.4 illustrates an example information set about a file stored by the storage application. The shown location information enables retrieval of stripes and the parameters are used in decoding the original file. The parameters m and n are striping factors as explained earlier. The attribute *packetSize* describes the size of the used encoding packet and the *fileLength* parameter is stored as bytes. The mappings from a file stripe to a storage element location contain the contact information to each server that hosts a stripe. In addition, checksum information can be added for ensuring integrity of the stored data. In the presented design, storage elements have a role of simple data storage without any embedded logic. The metadata is stored in an independent look-up service. Publication I contains some further security considerations on storing the file stripes and metadata.

Architectural Overview

The use of erasure coding is novel approach in the context of the Grid data storages. This section presents a design that uses this approach with the existing Grid storage system. The system is implemented in software and different pieces of functionality of the system are grouped into separate logical entities. The grouping allows to deploy the storage in various networking environments. For example, in local area networks low communication latencies make fast communication possible between the nodes of the storage system. This can be used to group multiple clients together to enable efficient access to data and to perform decoding of stored data. In a global setting, a storage client can operate independent of other clients and access data directly from multiple locations. This enables direct access to data and removes need to use a third party for retrieving and decoding the data. In the following, the implementation of the GridBlocks-DISK storage design is discussed.

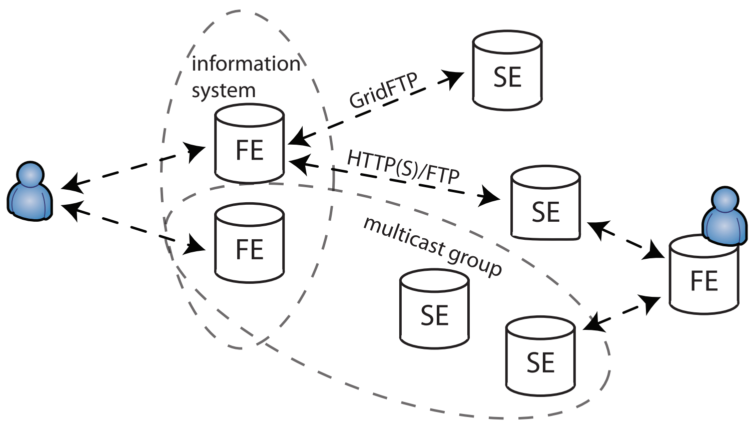

Figure 2.5: GridBlocks-DISK architecture in local area network.

Figure 2.5 illustrates architecture of GridBlocks-DISK [37] storage system. The functionality of the system is grouped into several components, so that each component can take independently care of a separate part of system's functionality. The figure illustrates the system when deployed for use in a local area network. The system has two main components, which are front-end (FE) and storage elements (SE). These components host the following functionality.

Each *Front-end* hosts a *Coordinator* that coordinates the overall operation of the storage. The Coordinator performs file striping and erasure coding. It has a responsibility to keep the information about stored files up-to-date in the metadata service. The Coordinator also manages leaving and joining of storage elements. In addition, the Coordinator works as an access point to data by providing interfaces to the Grid and the Internet and exposing a simple API for third party applications.

The *Information system* stores metadata about the stored files. The metadata includes erasure coding parameters, stripe integrity and the physical locations of the stripes. The information system is implemented using caching and replication of data objects over JGroups [38], which is an implementation of a reliable multicast protocol. The information system provides a scalable, highly-available and fault-tolerant metadata service for the storage system.

Storage elements contribute storage space for storing the stripes. The storage implements only a simple file storage and the processing requirements from the hosts is mainly limited to network and disk I/O operations. The elements need to support only a simple set of operations (get, put, exists, delete), which makes it flexible to adapt different elements to work together with the existing components. New storage elements can join the storage system by opening access to the most convenient of several supported transport protocol interfaces.

The minimal installation of the presented storage system requires a front-end instance and at least a single storage element instance; both can be run in the same physical host. In more fault tolerant deployments, the information system runs on multiple front–end servers to provide load balancing for coding and decoding and high availability of the stored metadata. In a typical use case a client uploads a file to the Coordinator. Then, the Coordinator performs striping and coding and stores the file fragments in n separate storage elements. It provides user interfaces for the client through command line, web browser or a WebDAV folder utility. It is also possible to run the Coordinator locally on a client node. In such deployment the client performs coding and striping and stores the metadata either locally or shares it between the other Coordinator instances. If the client wishes to use services for automatic resource discovery or metadata replication, it can join the group communication channel of the nodes hosting a Coordinator instance. To secure the channel, a cryptographic authentication token can be used as a requirement for joining. The Coordinator is run locally and metadata is stored locally in the client node when the storage system is deployed in a global network setting in the evaluation part of this Thesis.

The presented storage system makes it possible to investigate distributed data access in the Grid. The design of the system ensures interoperability with the widely used Grid storage elements by supporting the GridFTP interface. The Grid architecture is used through functionality that belongs to the following layers in the Grid protocol stack [69]. The storage system provides services of the Collective layer, since it coordinates use of a set of resources. GridFTP protocol [51] is used to provide access to data which belongs to the Resource layer. The GridFTP connection and security operations belong to the Connectivity layer. Finally, the storage systems in diverse sites and the networks connecting these sites belong to Fabric layer.

Previous research has focused on algorithms and architectures for optimizing replica placement in prior to data access in the Grid. There are other studies, for example, by Vazhkudai [143], on optimizing data access during the download by using several replicas in different sites as sources for the data transfer. The presented system is closer to the latter, since it increases the number of possible download sources. This is possible because erasure coding allows to store more large unique data items than it is possible with replication and fixed storage space requirements.

2.4 Evaluating Coded Storage in a Global Testbed

This section illustrates how erasure coding can be efficiently used with existing data storages that are connected to the Grid infrastructure. A design is proposed for a storage application and its software implementation is used with a distributed testbed to evaluate the feasibility of using erasure coding and striping to store data in the Grid. The evaluations show that the computational delays caused by erasure coding operations are small and that the proposed approach works well also with large files. Moreover, the evaluations show that the distributed data can be efficiently accessed and the approach can overcome poor access performance caused by low transfer capacity between distant locations.

Test Environment

To conduct the measurements in a Global testbed the storage system is implemented as a software application. The application performs software based erasure coding and manages transfers between a client and several storages in the Grid. An existing GridFTP client [85] is used to gain secure access to the industry standard Grid storage elements, which are provided by the Globus project [6].

For the measurements, a client in Switzerland at CERN distributes file stripes to five different storage elements, which are connected to the Internet. The storage elements are located as follows: the client and one of the storage elements reside at CERN, one storage element is in Slovakia, two storage elements are in Finland, and two in Delaware in the United States. All the components of the testbed are connected to high speed academic networks. This experiment setup produces a setting which is characteristic to data intensive Grid infrastructures.

Summary of Performance Evaluation

Before conducting the measurements in the storage testebeds, experimentation with a range of parameters is conducted to optimize the local coding and decoding performance. The results, shown in Publication I and achieved by using standard Linux desktop computer, can be summarized as follows. When the number of original stripes $m = 5$ stays fixed and n is increased, encoding performance slowly decreases. Performance stays in the order of tens of Mbits/s. Measurements suggest choosing 20 KB for the packet size in the striping and coding process. The coding measurements also show how the coding performance can be very efficient with small files, but maintains a steady performance even with very large files.

The first measurements use files of three different sizes to investigate how the file size affects the ratio between the time used for coding and the time used for network transfers. The measurements are conducted 40 times with file size of 10 MB and 100 MB and 10 times with larger 1000 MB file. For all results the average of the measurements is plotted. The client application creates a dedicated thread to control transfer from each separate storage element. Authentication and connection establishment is carried out separately for each connection.

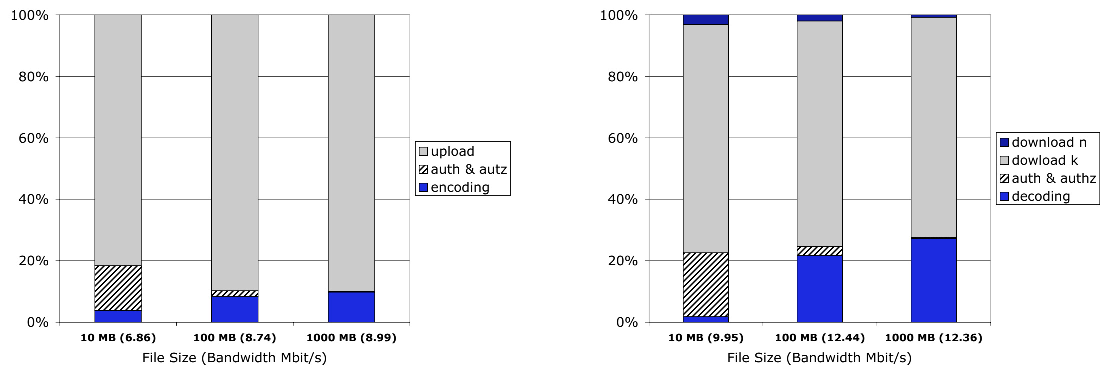

Figure 2.6: Coding delays during file upload (left) and dowload (right).

Figure 2.6 shows the time used for different parts of the file transfer process. Time used for encoding from $k = 5$ to $n = 8$ stripes is illustrated as the lowest part of each bar. In the download figure, this part represents time to decode and re–assemble. The time used for the Grid security operations, including authentication and authorization for all $n = 8$ stripes, is plotted as the second lowest part of each bar. The time used for completing all n uploads is plotted as the upmost part of the bars. In the download figure, the second upmost part illustrates the time for completing download of first $k = 5$ stripes, after which the decoding can begin. The time for completing the remaining transfers is shown as the upmost part of the bars.

The lowest part of each bar shows how the coding overhead is small compared to security and transfer latencies that are frequently present in the Grid environment. With small file sizes, the encoding takes even less time than used for mutual authentication between the client and the storage resources. The proportional time used for encoding increases with the larger files, but as mentioned above the coding bandwidth is close to constant with files larger than 1000 MB.

During file upload, the erasure coding causes processing overhead that is less than 10% of the total time. In addition, the amount of data that needs to be transferred is increased to $(n/m) filesize$. This increases the traffic that the client and its network connection experience during upload. During download, the proportional time for decoding increases with growing file sizes, but remains in the range of 25% of the total time even with 1000 MB files. In the storage element end, the network and each server need to handle only $(1/m) filesize$ and efficient load balancing is achieved. To further optimize the download, the client can stop the slow transfers already in the beginning and treat those stripes as unaccessible. In addition, compared to an equally fault tolerant replication, less storage resources need to be allocated.

The following compares the erasure coded upload and access to the plain replication approach. The transfer time for stripes is measured against parallel transfer of multiple replicas. For the comparison, in both cases a transfer approach is applied that leads to successful transfer even if $n - m$ connections cannot be formed or are disrupted, or if any $n - m$ storage elements are or become unavailable.

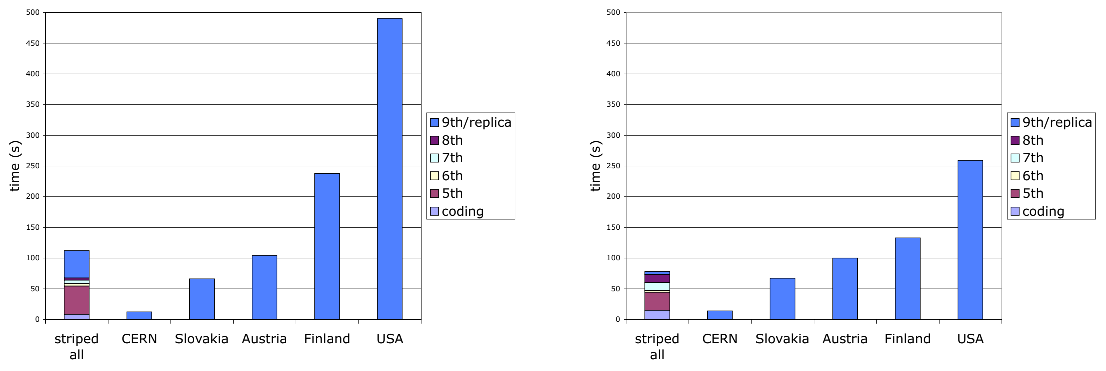

Figure 2.7: Delays in 4-available file upload (left) and download (right).

Figure 2.7 shows comparison of a striped 4-available upload (left) and download (right) to an equally fault tolerant replicated transfer. For striping, parameters $k = 5$ and $n = 9$ are used and, for replication, 5 replicas are concurrently uploaded. A file size of 100 MB is used and average of 10 measurements is plotted. Seven GridFTP storage elements are located in five geographically distributed sites, as explained above. The results show how the transfer time for the striped files is significantly lower than the replicated transfer to and from distant locations. Even when two stripes are transferred to and from a site in the USA and two stripes to and from Finland the overall transfer time stays clearly lower than transferring an entire file to or from a costly location. The amount of data that needs to be transferred close to client to achieve 4-availability, is $(9/5) \times filesize$ with striped transfer against $5 \times filesize$ with replication approach. Both, the upload and download measurements, show that the proposed design loses in performance to an access to a nearby replica. However, good performance can be sustained even when some of the data is transferred to or from distant locations, which have effect similar to slow connections or heavily loaded severs.

2.5 Evaluating Coded Storage in a Local Network

This section discusses performance of the erasure coding storage system in a local network. An example use case for a local setup is a class room or an office where storage elements are hosted on commodity desktops. In this type of settings, the erasure coding can protect from unavailability of storage elements or Coordinator instances. The experimental setting allows to observe how the overall operation consisting of computing intensive coding taks and network intensive access tasks can be performed in parallel when adjacent nodes must share the limited resources of the local network.

Test Environment

The local network allows to make use of the small communication latencies and group multiple physical resources for sharing the computing intensive striping and coding tasks. The grouping is established by deploying multiple Coordinator instances into the same network. The Coordinator instances join a group communication channel, which makes it possible to share metadata and computing tasks within the group.

The storage system is installed and run in multiple desktop workstations in a classroom. A single workstation hosts either a front-end or a storage element. The use of a single front–end is compared to a case where coding and decoding tasks are shared between two front-ends. A client accesses data through a front-end, as shown in Figure 2.5, which retrieves data from multiple storage elements and performs decoding and re–assembly on behalf of the client. Any front-end can start retrieval and decoding immediately upon client's request, because metadata is shared among all the Coordinator instances. Thus, the client does not have to use same front-end for accessing a file that was earlier used to upload the file.

Summary of Performance Evaluation

The following measurements use one or two front–ends to serve a set of clients. For each measurement, every client transfers 50 files through the FEs. The files have a random size from a set $[20, 40, 60, 80, 100]$ MB. The randomized file sizes are used to avoid synchronization and artificial load peaks. For each performance measure, average, max, min, median, lower quartile and upper quartile results are provided.

For each measurement, aggregate data transfer bandwidth is calculated from the file size divided by the time it takes for a file to travel through the components of the system. This includes transfer between the client and the FE, coding, calculating checksum, metadata operations, and transfer time between the FE and multiple SEs. The aggregate time describes the capacity of the FE to process stored data. All measurements use $m = 3$ and $n = 5$ to obtain a configuration where any two SEs can be simultaneously unavailable. The results show how duplicating the FE node helps to increase the throughput of the storage system. Simple round-robin scheduling is used to issue "put" and "get" requests between two identical FEs. Metadata is replicated to both FEs so that the metadata can be accessed locally in any FE independent of which one was used to store the file. Metadata replication also helps to preserve data when FEs are lost.

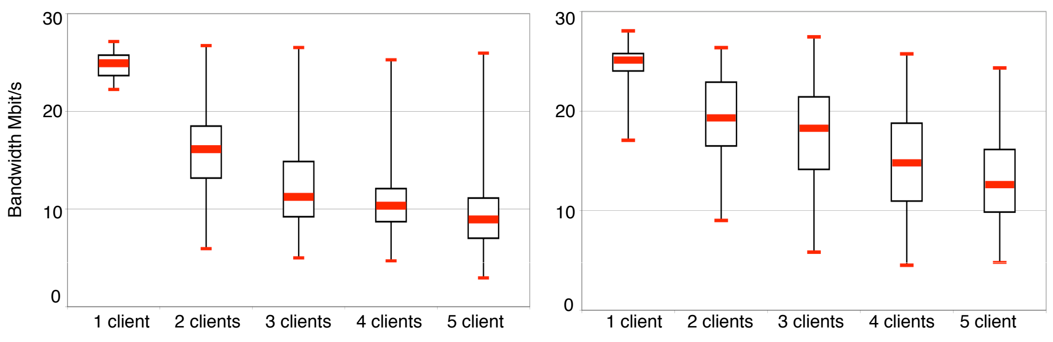

Figure 2.8: Access performance with one (left) and two (right) front–ends.

Figure 2.8 shows data access performance with a single FE and two FEs. The combined bandwidth of the transfer and decoding procedure is plotted as seen from the client's perspective. A single FE can serve one client with half of the cases falling between $24-26$ Mbit/s. With greater number of simultaneous clients, the median performance decreases moderately, achieving close to a third of the performance with 5 clients compared to the case with only single client. The performance becomes less predictable when the number of the clients increases. With the addition of another front–end, performance increases of about 50% are seen when three or more clients are accessing the storage. Although the performance decreases from the clients' perspective, when more clients are served simultaneously, the overall throughput increases

Different system requirements of the sub-tasks help to increase parallelism, because the file transfer, decoding and integrity tests do not continuously interfere with each other. Using two FEs decreases the metadata service look-up time significantly. Even that the look-ups are always processed locally, the duplication of FEs helps to share the load. The upload measurements show results very similar to download case, because the presented figures describe also the upload case well they are omitted. However, the clients can upload files to a remote FE like to a traditional server, after which the FE takes over the processing. Thus, the client sees fast upload not including the time used to wait for coding and transfer of stripes to SEs.

2.6 Summary

This section presented a novel design for a Grid storage system. The system uses erasure coding and striping to ensure high availability of data. The data availability is increased in two ways. First, the decreased storage space requirement makes it possible to survive from larger number of simultaneous failures than with any approach based on simple replication. Second, distributing larger number of data items increases the number of sources for download scheduling and can decrease access delay to the data. The choice to manage striped data as plain files enables simple operation by using standardized data transfer tools. However, a tradeoff is a temporary need for $(n/k) * filesize$ amount of storage to store the stripes between the coding and network operations. This temporary need for the extra storage space could be removed by sending data immediately after the coding. More sophisticated stream–oriented coding remains an item for future work.

The performance evaluation shows first results on using a real implementation of an erasure coding storage client and the Grid storage elements. The results show that the software based erasure coding is feasible to use when implemented as a Java application. The coding causes relatively small processing overhead when compared to network and security related time requirements. A small set of resources available for the testing purposes still limited the generalizability of the results, and more comprehensive tests are needed with more storage locations used in the measurements.

The erasure coding storage system operates efficiently also in local area networking environments. The low communication latency makes it possible to efficiently distribute metadata and group membership information by using a reliable multicast service. This enables employing the storage system in such way that computing intensive tasks can be distributed for load balancing purposes. Empirical measurements in a modern workstation environment show that multiple front-ends improve overall storage performance. The improvement stems from performing striping and encoding in parallel. Using two front–ends showed that significant performance improvements can be achieved when several clients are simultaneously transferring data to and from the storage.

So far, only simple round-robin scheduling was used for uploading the stripes. In the downloading process n transfers were started. After m stripes arrived and were verified to be intact, the remaining transfers were stopped. However, the metadata system could be used to store also operational information, such as performance statistics, about the storage elements. Based on this information, the storage system could then perform more sophisticated scheduling to decide the best locations for uploading the files. Similarly, the application could also be extended to deduce the fastest storage elements to download the stripes from.

Recent work on erasure codes has given motivation to study new classes of codes for network file storages. While Reed Solomon codes have been shown to be suitable for the presented system, some other codes can be more feasible when applied in new innovative ways. In particular, applicability of LDPC codes [112] (for example, Raptor [131] and LT [131]) is an interesting topic for future research. Moreover, support for streaming data together with partial file access offer several interesting directions for future work.

3 Coping with Failures of Network Communication

This chapter continues research on methods for increasing data availability in the presence of failures. The investigation moves to more challenging scenarios, where the network communications cannot be considered reliable. In these scenarios the connections can be frequently unavailable, which leads to partitioned networks and broken network paths. The network can also be built entirely on an *opportunistic* basis, that is, the communications are available only when a possibility to form a *contact* between a pair of nodes appears [32]. This type of network communicates over intermittent connections between pairs of adjacent nodes and provides a very challenging network scenario for communications research. This scenario is used to investigate mechanisms for increasing reliability of the content retrieval in opportunistic networks in this chapter. The numerous investigated mechanisms improve information access and communications performance in challenging conditions.

To support content retrieval in an opportunistic network a design for a mobile node is presented that is able to interpret application operations related to content retrieval and adapt routing accordingly. The design allows a node, for example, a mobile phone carried by a user, to contribute resources and act as a source for content retrieval requests that are initiated by the other nodes in the vicinity. This enables content retrieval operations even when distant and pre-determined network services cannot be reached. This chapter also evaluates mechanisms for fragmenting large messages, so that they can be delivered over intermittent contacts with limited transfer capacity. In addition, content search mechanisms enable content retrieval when the user does not know an exact identifier of the content at the time of content access. Moreover, partial infrastructure support for content access from the web is investigated to discuss how limited structure and geo-awareness can provide support for information retrieval. Together these mechanisms outline the basic support for accessing content in challenging networks where failures are a feature rather than an exception.

In the Internet, a large number of IP packets can be used for communication, because relatively stable paths are assumed between the end-points of the communication and the packets are likely to reach their destination. A common assumption is that the packets are not stored for a long period of time in intermediaries, but are transmitted after relatively short time is used for queuing and transmission. This model of communication is called *store-and-forward* paradigm. The delay-tolerant networking (DTN) architecture is designed to make communication possible where the IP-based communication fails, because end-to-end paths cannot be established or maintained in the network. Thus, instantaneous delivery between the end nodes may not be possible in the DTNs. To overcome this limitation the intermediate nodes in the network store and possibly *carry* the messages until an opportunity to forward appears. This leads to a communication model, which is called *store-carry-and-forward* paradigm. Examples of networks where end-to-end paths cannot be assumed to exist include sensor-based networks with intermittent connectivity, satellite networks, underwater acoustic networks, and unreliable terrestrial wireless networks [20]. DTN is also suggested for communications in rural areas [34] and as a protocol to enable direct inter–personal communications based on the proximity of the communicating nodes [124].

From the application perspective, the traditional IP networks use independent data items (that is, IP packets) that are meaningful only for the routing layer, but often by themselves not so significant for the communication applications. Thus, a large number of packets need to be exchanged to transfer an information entity that is meaningful for the communicating applications. In an unreliable network this prevents application interactions if some of the related packets do not arrive to destination. To make the individual messages more meaningful for the applications, DTN uses messages that contain set of resources that can be interpreted by the communicating applications. These set of resources are called Application Data Units (ADUs) [20] and they are carried inside *bundles*, for which Bundle Protocol [125] defines the format. This model offers a basis for the application-aware router that is discussed in this section. Moreover, it motivates to investigate the applications themselves that are used for content retrieval and communications.

The DTN communication is often suggested for communication scenarios where the resources are scarce. The resources include, for example, contacts between adjacent nodes, memory and storage in nodes, connectivity of nodes to infrastructure networks, processing power, and battery capacity of the nodes. When a large part of the network is formed from nodes with scarce resources, it becomes important to design applications and communication models in such a way that the resources are used sparingly. At the same time, the goal is also to deliver messages to the destinations with the highest reliability. This section discusses the DTN architecture and shows how it can be used for communication in a network that is formed by mobile users with scarce resources. Several mechanisms for and extensions to the current architecture are proposed to enable efficient content access in DTNs formed between mobile users. Some of the presented mechanisms are fairly general and not specifically tied to the DTN architecture. Thus, they could be applied also with other implementations that support opportunistic networking, for example, Haggle [124, 99] architecture.

This chapter has several parts, which cover both background work and actual contribution presented in this thesis. The first subsection reviews the delay-tolerant networking architecture. The second subsection covers related research from several related fields. The actual contribution by the author starts in the third subsection, which introduces the concept of application-aware DTN networking that allows the nodes in the vicinity to respond to content retrieval requests. The fourth subsection continues by discussing how the application-aware design can be used to enable content caching and storage in DTNs. The fifth subsection discusses various message fragmentation mechanisms that enable overcoming limited transfer capacity of contacts in opportunistic networks. The sixth subsection covers search mechanisms that allow accessing data in mobile DTNs when content location is not known. The seventh subsection discusses how placing DTN–enabled WLAN hotspots to opportunistic scenarios adds structure to network that can be used to improve Web access via DTN. The eight subsection shows performance evaluation results for the mechanisms developed between the third and seventh subsections. Finally, the ninth subsection summarizes the chapter.

3.1 Delay-tolerant Networking Architecture

This section introduces the delay-tolerant networking architecture, which is background to work conducted in this thesis. The architecture is currently under active development in DTN research group (DTNRG) [60], which works under the IRTF and aims to define concepts central to the DTN networking and communications. The group has produced an informational RFC 4838 [20] that explains the DTN architecture. This section gives an overview on the DTN architecture. The outline of the section mostly relies on RFC 4838, and further references from the literature are frequently provided.

Bundle Protocol and Internetworking

The DTN architecture uses messages which are called *bundles* [125]. The bundles are often larger than packets used in IP networks. Nodes in a DTN network store and possibly carry bundles until an opportunity to transfer bundles to the next node appears, these opportunities are called *contacts*. When transferring bundles over the contacts, the nodes can use heterogeneous mechanisms to ensure reliable bundle delivery to the next intermediate node. This is different from the traditional mechanism used in the Internet, where the TCP is used to control transmission between the end points of the communication.

Figure 3.1, illustrates the protocol structure in a DTN network. The functionality of the network is grouped into several layers. All entities in a DTN network implement the bundle protocol, which is specified in RFC 5050 [125]. The Bundle Protocol layer provides transfer services for the communicating applications, and can use services of heterogeneous lower layers to forward the bundles. The source and the destination of the communication are both presented as a *DTN host* in the figure. A *DTN router* communicates with the other nodes by using the TCP/IP protocol stack. A *DTN gateway* runs two different stacks below the Bundle Protocol layer to mediate bundles between two different types of networks.

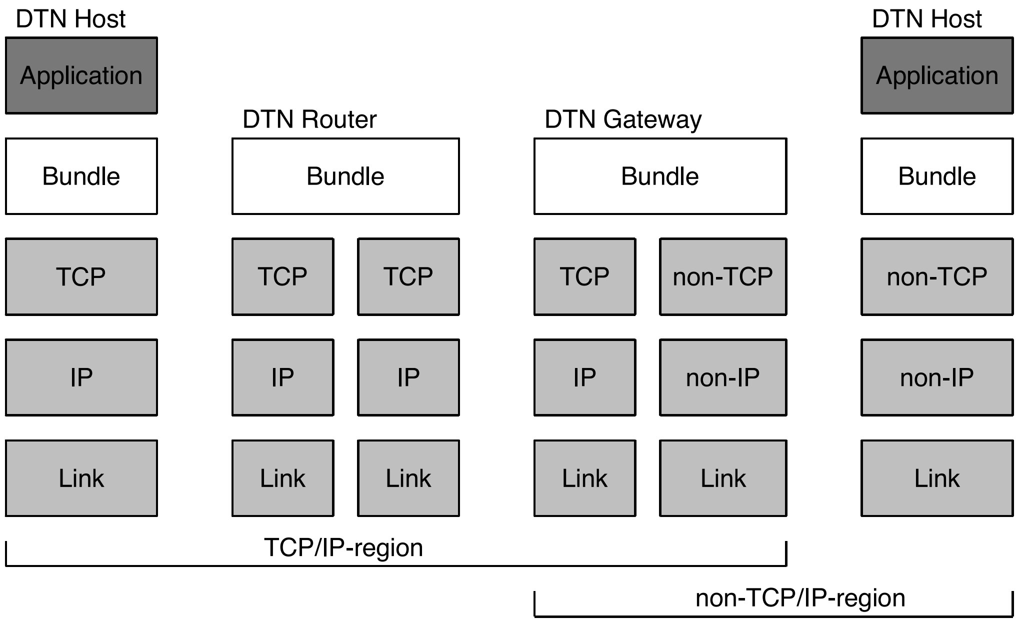

Figure 3.1: Architecture of the DTN protocol stack [145].

The DTN architecture allows internetworking between heterogeneous networks that use different protocols and addressing families below the bundle layer [46]. To separate between the different networks the architecture uses the concept of *regions*. When nodes that belong to separate regions are willing to communicate, they need support from a DTN gateway, which is described as a point in the network that data must pass to gain access to a different region. The gateways may perform potentially needed encoding translations between the regions and may also enforce networking and security policies.

Convergence Layers

The DTN nodes can use different techniques below the bundle layer to communicate. To enable communication between the bundle layer and the heterogeneous lower layer protocols, a concept of *convergence layers* has been introduced. The convergence layers manage detailed interactions with the lower layer protocols, for example, TCP/IP, and present a consistent interface to the bundle layer. This enables running the bundle agents without any modifications on top of several different networks.

The TCP convergence layer [36] which is a part of the DTN reference implementation is an example of a convergence layer. The convergence layer can be used to convey bundles between two nodes by establishing a TCP connection between the nodes. The convergence layer also specifies how to encapsulate data to bundles for a delivery over TCP, and provides acknowledgments for the delivered data segments, which can be used to support partial message delivery.

DTN Nodes and Endpoint Identifiers (EIDs)

A *DTN node* is an engine capable of sending and receiving bundles. In other words, each DTN node implements the bundle layer, as illustrated in Figure 3.1. A DTN *endpoint* is logically a group of nodes, and the bundles are sent between the endpoints. Each group may consist of one or many nodes and each node may be part of one or many DTN endpoints. A message send to an endpoint may require reaching a single node (unicast), one node of the group (anycast), or all nodes of the group (multicast and broadcast) in order to lead to a successful delivery.

Each DTN endpoint is identified by an endpoint identifier (EID). An EID is a name that follows the general format of uniform resource identifier (URI). Applications use an EID to define the destination of each bundle that is sent over DTN. In the receiving end, the application performs a *registration* when it is willing to receive messages destined to a specific EID. Nodes often persist registration information in order to manage restarts of the node or an application. The EIDs are not required to have functionality related to routing, but they may have if it is found beneficial in the deployment environment.

The process of interpreting an EID, for determining the way to carry the message towards destination, is called *binding*. In the binding process, the destination EID can be mapped, for example, to the current IP address of the EID. In the traditional Internet communications the binding is done prior to sending, when name of the destination is associated to its network address by the means of a DNS query. This is called *early binding*.

In DTNs *late binding* is often used. In the late binding the destination EID is not necessarily bound to the destination address in the source DTN host. Late binding is especially feasible when the messages may travel to the destination longer than the bindings are valid, that is, when name to network address mapping may become obsolete during message transfer. Use of late binding also removes the need to spread up-to-date binding information across the network.

Ongoing work [48] defines rules for constructing EIDs. The rules can define bundle content to be carried inside various protocols, for example, by using HTTP or email mechanisms. The rules provide increased control on delivery path and operations that are performed on the bundle, and their use is briefly discussed in Publication VIII.

Routing and Forwarding

The DTN network can be considered as a multigraph, where each node-pair may be connected by more than one edge. A period of time that an edge is available for delivering messages is referred as a *contact*. The contact's *volume* is the duration of the contact multiplied by its bandwidth and it is equal to the amount of data that the contact is able to carry. The routing differs from the routing in fixed networks in that the contact availabilities and volumes may not be known in advance. Instead of just finding a path in space, the routing needs to find a space-time path to the destination according to the temporal availability of the contacts. It is often difficult or even impossible to establish an instantaneous end-to-end path between the source and the destination. Thus, the path and its availability may not be known at the time of sending the messages and the message delivery may not be guaranteed. The DTN architecture leaves the choice of the routing algorithm open and provides support for different routing algorithms so that an appropriate algorithm can be chosen according to the characteristics of the network.

This thesis assumes routing algorithms that work over *intermittent* and *opportunistic* contacts and omits details of routing messages over other possible contact classes, such as scheduled contacts. An *opportunistic routing* protocol can operate based on zero-knowledge, that is, it does not require information about the network topology. Several such protocols have been proposed for routing in DTNs and new routing approaches emerge continuously. In the following, a brief overview of the DTN routing is given together with references to more detailed information.

Several DTN routing protocols create multiple copies of the message to find a good route to destination and to survive possible message losses. A well known example of such protocol is Epidemic routing [141]. In Epidemic routing, the nodes exchange a summary about the messages which they carry when meeting the other nodes. Then copies of the messages are replicated to the nodes that are not yet carrying them. In this process, a large and uncontrolled number of copies is created and it may lead to an undesired congestion. Spray-and-Wait [134, 135] routing works by initially creating and sending (spray phase) limited number of message copies into network. Then these copies are carried towards the destination (wait phase). In Spray-and-Wait routing the number of the message copies is explicitly limited and does not directly depend on the number of nodes in the network.

Routing algorithms that create only one copy of a message have been proposed for networks where the resources are scarce [136]. In such environments, single–copy routing helps to avoid issues arising from congestion and energy usage, which are often faced with flooding based message delivery. The simplest form of the single–copy routing is called *direct transmission*, in which a node forwards a message to the next node only if the next node is the destination of the message. In a randomized routing algorithm the node which carries the message passes the message to another node with some probability. First contact (FC) routing [73] is an example of such a protocol with property that the node always passes the message to another node.

Utility–based routing uses information about earlier encounters between a pair of nodes to find a node that is likely to have an encounter with the destination in the future. The use of this information can be based, for example, on observation that smaller time since last encounter usually indicates a shorter distance between a pair of nodes. PRoPHET [92] protocol maintains for each node a metric called *delivery predictability*, which indicates how likely it is that the node can deliver a message to a given destination. The metric is increased when encounters between the node and the destination happen, and it is decreased when encounters are not seen for a while. The metric has also a transitive component, which is increased if the node can reach the destination via another node. Seek-and-focus [136] routing combines the random and utility–based routing. It first uses random routing to find a good lead towards a destination and then uses utility–based approach to follow that lead.

Spray-and-Focus [135] protocol combines the Spray-and-Wait approach with the seek-and-focus protocol. In the focus phase of the Spray-and-Focus routing, an utility function is used. If the encountered node is more likely to meet the destination based on an utility function, the message is forwarded to the node. Thus, Spray-and-Focus takes active measures to propagate the messages towards the destination also after the spray phase. The MaxProp routing algorithm approaches the congestion problem by explicitly acknowledging successful deliveries and removing copies of the acknowledged bundles [16]. The RAPID routing protocol creates new copies of a message until a copy reaches the destination [8]. RAPID uses an utility function to evaluate which packet would lead to highest increase in utility, and then replicates this packet first. The utility–based evaluation is also used to determine which packets to drop when congestion takes place. Since the utility function can be customized, the protocol behavior can be adapted to different environments.

Several studies [76, 151] compare performance of different DTN routing protocols.The above introduction classified the DTN routing protocols based on how many copies they create per message. Other approaches [144] and Publication IV have discussed the use of coding to increase DTN forwarding robustness with only a small amount of additional traffic. Also routing protocols that find and exploit social models in community have been proposed, for example, Bubble RAP [66]. A solution for routing response messages backwards that uses the previously existed route as a hint has been proposed by Hui et *al.* [65].

Fragmentation and Reassembly

The DTN architecture proposes to send data in bundles, which aggregate information, instead of sending several small data packets. The model is proposed to be used in environments where the resources are scarce and the contacts may be only shortly available for the data transmission. This leads to a question on how to maximally utilize the short contact durations, and how to forward even when the contact durations are too short for transmitting entire bundles. To overcome this challenge, the DTN architecture proposes *fragmentation* of the bundles. In the process of fragmentation, the bundles are divided into multiple smaller bundles which are called fragments. Conceptually, each resulting fragment is treated as a plain bundle by the DTN network. When possible, fragments of a bundle are merged and the bundle becomes re–assembled.

Fragmentation is designed to increase the utilization of contact volumes, but the approach also raises many questions. The size and the content of the bundles can change in the intermediate nodes during the transfer, and the payload authenticity can become an issue. Uncontrolled fragmentation can lead to excess message splitting and large number of fragments, while loosing any of the fragments during the transfer leads to an unsuccessful message delivery. Also, it is not straightforward to route the bundles in an optimal manner, so that the entire resource may be later re-assembled. The DTN architecture defines two forms of fragmentation. In *proactive fragmentation* the sender prepares for small contact volumes by fragmenting the bundle into several smaller blocks already before the transfer. In *reactive fragmentation*, two adjacent DTN nodes fragment the bundle during the transfer when it is only partially transmitted.

Reliability and Custody Transfer

The bundle delivery model in DTNs is one-way and does not include interactivity. The model supports unacknowledged, and priority supported delivery of bundles and it is often compared to the model used in the traditional postal service. To provide some degree of message delivery guarantees for the communicating applications, the bundle protocol supports two mechanisms. First, applications that have an explicit need for delivery guarantees are allowed to use end-to-end acknowledgments. Second, a node can promise to store a bundle reliably until it can forward it towards the destination, that is, it can accept *custody* of the bundle. The node can move the responsibility of storing and forwarding the message for another node by means of *custody transfer*.

Accepting custody of a bundle requires a node to reserve resources for storing the bundle, and it is not allowed to delete the bundle. The custody transfer mechanism allows a node to release the resources when the responsibility is moved for another node. However, the architecture does not require all nodes to act as custodians, and some nodes with scarce resources may decline becoming custodians. This limits the use of custody transfer for providing hop-by-hop reliability.

3.2 Related Research

This thesis assumes the DTN architecture [20] as a basis for communication in challenging networks. The DTN bundle protocol [125] and its reference implementation [59] are used as a starting point. Then several mechanisms that extend their default functionality are proposed throughout this chapter. The following gives a review on related work in the field.

DTNs and Applications

Earlier studies on using the DTN for web access [102, 10] have illustrated how the web resources can be bundled into semantically self-contained units. Ott [100] has discussed application protocol design for mobile Internet and concluded that while asynchronous communication model seems suitable for the Internet access, the end user applications should proactively adapt to asynchronous communications and communicate their intentions at once. Lindgren and Hui [93] discuss several areas where DTN could provide a feasible solution for applications to communicate. They propose that the rural communications and bulk data transfer in urban areas are potential scenarios where the DTNs can be the most prominent basis for communications.

DTNs have frequently been referred to offer a postal class of service [46, 20]. This service model has been leveraged in enabling Internet email system on top of the DTN Bundle Protocol, for which a practical implementation has been demonstrated [68]. Other examples of applications implemented for DTN include a web blog [107] and a web server [108] capable of bundling resources for DTN delivery.

Vallina-Rodriguez et *al.* [142] have proposed a social networking service for developing regions that uses combination of DTN and infrastructure networks. Similarly, Clark et *al.* [27] have proposed social networking application for that uses the DTN for communication. Moreover, MobiClique [109] is a middleware for opportunistic networking that has been used to support applications for mobile social networking, asynchronous messaging, and epidemic newsgroups. While not built on the DTN architecture, the approach has many similarities to applications discussed in this thesis, as it uses store-carry-and-forward paradigm, epidemic messaging, and TTL to limit message spreading. Lenders et *al.* [89] have proposed broadcasting to share and distribute media content over opportunistic wireless links. While using smaller data units and channels to identify content, their use of opportunistic contacts share similarities to mecnahisms discussed in this thesis.

Content Redundancy and Cooperative Caching

Jain *et al.* [72] have discussed how redundancy can be used to cope with communication failures in DTN. They formalize the problem of sending coded blocks over different paths and propose strategies for packet scheduling so that the overall message delivery probability is maximized. Deploying additional DTN nodes to serve as bundle routers has been also suggested for improving capacity, and thus also reliability, in DTNs [152]. Erasure coding-based flooding has been shown to provide good performance in opportunistic networks [144], reducing the message delivery delay. Other proposals to use erasure coding [17, 119, 57] discuss the benefits of removing the need for interactive communications, especially sending acknowledgments, during reliable data transfer. Work with data storages, discussed in Publication I and in Publication II, has shown how how large files can be efficiently striped and erasure coded for storage. The striping scheme can be used for very large data units that are similar to ADUs in DTNs.

Other research has discussed optimizing the file availability in peer-to-peer content distribution communities, defined by intermittently connected nodes that contribute storage, contents, and bandwidth [79]. The primary goal is to satisfy file requests from within the community, which also increases the speed of downloads from the global network since less peers compete for the outgoing link capacity. Only when the community cannot serve the file the request should be sent further to the global network. Similarly, cooperative caching may be applied in ad-hoc networks to serve requests from mobile users and reduce the distance requests and responses have to travel [150]. This issue is further discussed in the context of cooperative caching for multimedia contents in [86].

Scalable peer-to-peer overlays have been studied in the content addressable network (CAN) [118] and Chord [137] projects. These projects have provided algorithms for addressing content in scalable manner. The PAST project has investigated caching in storage overlays and shown how locally controlled caching may be used to increase the fetch performance in the storage overlays [120]. Their research also illustrated how even during very high storage utilization the small files may be still stored for caching purposes to where some unused storage exists. This is similar to the motivation to perform cooperative caching, since the proposed model can opportunistically store even small bundle fragments without inducing overhead of registering the stored fragments to look-up service. The Internet Backplane Protocol [12] creates a global storage service by sharing an end system's disk resources via an infrastructure network and offering access and management functions. While the above approaches provide rather predictive resource access, the focus in DTNs is more on probabilistic operation in opportunistic networks where individual nodes are not assumed to be well connected. The maintenance cost of look-up structure in opportunistic networks easily becomes unacceptably high. OceanStore [84] has investigated storage access for weakly connected nodes and others (for example, [33]) have studied applying peer-to-peer overlays in mobile ad-hoc networks.

Cooperative caching has been suggested to save energy [122, 121, 130], increase data availability, and decrease access delay [43] when data is accessed from mobile ad-hoc networks. In the ad-hoc networks the routing maintains and uses state about the node's neighbors to compute paths. This may not be suitable in highly mobile scenarios where purely opportunistic operation is used. Another approach, which is used in ad-hoc networks [149], uses path information to make decisions on caching. However, in opportunistic DTNs paths may not exist at all. Replica allocation in ad-hoc networks has been investigated to increase data availability in partitioned ad-hoc networks [62]. The authors suggested that when replica allocation is based on data access frequency and group information of the node the data availability is higher than without the use of group information. This could be paired with a method designed for finding social groups in DTN communities [66]. Group-based caching with closest neighbors in ad-hoc networks has been also suggested for increasing data availability [140].

The bundle protocol [125] defines also the *custody transfer* mechanism which combines hop-to-hop reliability and persistent storage to enable reliable end-to-end communications. As node resource constraints may lead to congestion, Seligman et al. propose mechanisms to temporarily shift storage load by moving custody of bundles from one node to another and retrieving them when the congestion has cleared [127]. Moreover, Seligman [126] has shown how the storage usage of the custody transfer mechanism can be used to make resource usage efficient when targeting reliable transfer in networks with intermittent connectivity.

Bundle Fragmentation in DTNs

The bundle protocol specification [125] defines the bundle format and handling of bundles. The specification also discusses fragmentation and defines that the fragments of a bundle are represented as bundles which can be fragmented further. Although, the syntax is in place, little information is available on how and when to use fragmentation. Most routing protocols defined so far consider transferring bundles only in an all–or–nothing fashion, for example, MaxProp [16], Spray-and-Wait [134], Prophet [92], and RAPID [8]. Using only complete bundles makes routing and forwarding design simpler. Exceptions are protocols which fragment messages to add redundancy for increased robustness of message delivery, as seen with erasure or network coding [144, 72, 146].

In IP networks using fragmentation has been discouraged [81], for example, because of increased packet loss probability, and is generally avoided. In datagram-based communications, the application layer is usually responsible for segmenting application data units so that they do not exceed the maximum packet size on the path, following the concept of *Application Layer Framing* [26], which is used for media streams [123, 61]. A similar approach can be used in DTNs and applications can limit the size of the messages that are sent to network.

Application level message fragmentation has been proposed for overcoming limited communication and storage resources in store-and-forward networks already in the late 1980s by Brachman and Chanson [13]. Their paper discusses many issues also applicable to communication scenarios targeted in this thesis, however, it assumes high reliability of the message transfers and their intermediaries communicate over more stable links.

Search for Content in Opportunistic Networks and in the Web

Searching for information is an active area of research in the computer science. It has produced numerous publications with topics varying from algorithms and operations of large-scale databases to search in many different application scenarios. The practical applications of searching have been illustrated by popularity of the Internet search engines such as Google or Bing. This thesis focuses on distributing search requests and results in challenged networks rather than the process of actually finding the results. Thus, the discussion of related work limits to similar efforts that perform search in opportunistic networks and to studies about user behavior which give insights about requirements.

There are two basic approaches for searching in unstructured networks, namely flooding and random walk. Most proposed flooding based schemes use a TTL-based limit to control the spread of the queries, for example, as described by Chang and Liu [21]. In contrast, search schemes based on random walks [7, 128], avoid the massive spreading of messages that flooding creates and still achieve a degree of reliability by using probabilistic paths to reach responders.

BubbleStorm [139] is a probabilistic search strategy in peer-to-peer networks. It is based on a combination of replication and probabilistic distribution of queries. BubbleStorm depends on a network that is a random multigraph with a fixed degree distribution. Thus, the system is not applicable for DTNs, as nodes cannot control the number of neighbors to peer with as peer-to-peer systems working over the Internet can.

Adamic et *al.* [1] present search strategies that exploit power-law node degree distributions, which estimate the number of neighbors for a node. Their method passes a search request from one node to another, choosing the neighbor with the highest degree. After the node with the maximum degree has been reached, it will be avoided, thus the search descends in the degree sequence. Yang and Hurson [148] present probabilistic schemes for locating content in wireless ad-hoc networks. Based on knowledge about the query history, they use heuristics and Bayesian probability calculations to guide the dissemination of queries. Hui et *al.* have presented a search mechanism for pocket switched networks (PSN) called *osmosis* that derives from analogy to its biological counterpart [67]. In osmosis, the queries in the network are spread based on epidemic forwarding, and the results are routed back based on the traces that queries left while traveling between requesting node and the responding node. The work in this thesis, in contrast, limits the system resources used to spread and evaluate searches and relies on different existing routing mechanisms to deliver responses.

Balasubramanian [11] et al. have presented a system called *Thedu* to enable efficient web search from a city bus. Thedu acts as a web proxy and collects search queries from a mobile user at the time of disconnections. Thedu differs from this work by assuming that nodes can eventually establish a connection to the Internet where a centralized search engine can be accessed. The proposal in this thesis, on the other hand, allows search in completely disconnected networks, where all content is distributed among the nodes.

There are a number of studies analyzing the use of web search. Based on the logs of the Excite search engine for one day (in March 1997) Jansen et *al.* [74] found that the majority of web searches consist of only few terms, and that the frequencies of occurrences of search terms exhibit a highly skewed distribution with 44% of the search terms occurring only once. Search sessions, that is, a series of queries by a single user within a short period of time, were also found to be short.

A similar study by Silverstein et *al.* [132] based on AltaVista query logs from 1998 supports the results of [74]. A later study using Altavista data [75] from 2002 showed a slightly higher number of terms per search and the diversity of search terms was found to be larger than in previous studies with the most frequently used term accounting for only 0.6% of all queries.

Kamvar and Baluja [78] analyze wireless search behavior based on Google's mobile search logs gathered during one month. Their results are in line with the general properties observed in the studies cited above whose data is mostly about searches from desktop computers: short queries and sessions and a high variation of the search terms. However, there is an observable difference between the types of devices. Searches from cellphones are slightly shorter than those from PDAs. The distribution of the used search terms is more biased for cellphones where the top 1000 queries account for 22% of all queries than for PDAs where the top 1000 represent only 6% of all queries.

Church et *al.*[25] present an extensive study of mobile information access conducted in 2005 based on logs from mobile a provider which include both search and browsing patterns. Their results are in line with the findings of the analysis based on Google mobile logs, and they predict that the diversity of search interests will grow for mobile search as it did for web search in general.

The overall observation is that the searches, and especially in mobile space, are repetitive. However, the variety of topics is very large, so that the top searches constitute only a fraction of the total search traffic. As a consequence, optimizations based on the most popular searches will have only a limited effect on the overall user experience. Unfortunately, none of these studies provides insight on the similarity of the search results.

With the available numbers, it is possible to only state that different queries also matched different results, but not to verify to what extent.

Opportunistic Web Access and Partial Infrastructure Support

Mobile Internet access, directly via diverse wireless technologies or indirectly via a co-operating peer, has been a research topic for more than a decade. Various approaches for dealing with opportunistic connectivity between mobile nodes and to access networks can be found. Numerous proxy architectures were developed for Internet access in disconnected environments. Ott and Kutcher suggest a proxy-based solution for Drive-thru Internet [102] access in disconnected environments by using a gateway between interactive HTTP request-response sequences and asynchronous DTN communications. This thesis re-uses their earlier scheme for encapsulating HTTP in DTN messages. The Tetherless Computing Architecture [129] has pursued similar ideas.

Besides content retrieval, searching for content in disconnected mobile environments has received quite some attention. Balasubramanian et *al.* [9] presented *Thedu* to enable efficient web search from a city bus for mobile nodes with intermittent WLAN connectivity: search queries collected by clients are aggregated by a central search engine, and the most relevant results are prefetched. Thedu assumes intermittent, but direct connectivity to a WLAN through which the server can be accessed and places its emphasis on dealing with search results rather than on generic page requests as in this thesis. The Haggle project [124, 99] provides an architecture for sharing and retrieving content from within (mobile opportunistic) social networks, associating users with social context for routing/forwarding and attach metadata to content for searching and identification.

In addition, various systems for pro-active content dissemination were developed. TACO-DTN [133] assumes an infrastructure backbone made up of well-connected infostations, for example, part of travel information system attached to bus stops. The infostations are used as gateways to publish contents to which the mobile nodes subscribe when they come into range. Published content is routed to those infostations with active subscriptions to be stored and ultimately delivered to the mobile nodes. A system for urban content dissemination originated from fixed access points is analyzed in [87]. A publish/subscribe architecture or large-scale (rural) dissemination is presented in [58]. PodNet [80] describes an interest-based content sharing system between mobile nodes in which content is organized according to channels described by metadata.

Finally, efficient content access can be supported by cooperative caching inside the network and in mobile nodes, reducing the distance between requester and a copy of the content and enabling content access without the need for infrastructure access. Internet access via densely populated MANETs was studied, for example, by Lim et *al.* [91] and numerous approaches to caching were devised. Lin and Cao [150] applied caching in MANETs using redirection of request packets in intermediate systems towards areas with cached copies. Ditto [41] is a caching system for multi-hop wireless networks that divides content into *chunks* of data that can cached in the nodes along the data path and in those overhearing the wireless transmissions. Afanasyev et *al.* [2] propose caching for individual packets at the link layer of wireless networks so that nodes overhearing wireless transmissions can keep packets to satisfy later requests.

The work in this thesis differs from the above in at least one of three key respects. First, operation is not done at the level of individual packets but larger messages. Second, the work aims at supporting the traditional access methods to structured web contents. Third, the focus is on opportunistic networks. Furthermore, the work on hotspot based Internet access extends the work in other publications of this thesis by exploring caching in fixed and mobile nodes.

3.3 Application-aware DTN Networking

One of the motivations for the DTN architecture is to make networking decisions more useful for applications [20]. To achieve this goal, the DTN applications communicate by sending data in Application Data Units (ADUs). An ADU may be packaged into a single or many bundles. However, in this thesis the aim is to package each ADU into a single bundle and to use ADUs to transfer semantically self-contained application protocol messages, for example, a single or even all objects of a web page. A message header extension is proposed that exposes the application contents of a message to the DTN routing layer. This allows also the intermediate DTN nodes to be aware of the purpose of the message for the communicating applications. The model is referred as *application-aware* DTN networking. This section discusses how a cooperation between DTN nodes can benefit from application-aware message forwarding. The discussion begins by showing what is needed in the DTN nodes and messages to enable application-aware operation. The discussion continues by focusing on caching and storage applications and by showing how to increase data availability by equipping DTN routers with an application-aware community caching and storage module.

Enabling application-aware processing allows to study how it can be used to share storage resources for increasing content availability in a sparse community of mobile nodes. The motivation to share storage resources for caching in DTNs derives from an interest to decrease retrieval latency, increase delivery success, and reduce network traffic. These are goals often pursued also in traditional network systems design. Earlier work [79] mentions how fulfilling requests from within such a community network also increases the performance of fetching data from outside the community as the gateway nodes become less congested.

The application-aware functionality of the nodes can be used for multiple purposes, for example, media format conversions. However, this thesis presents the generic application support, but discusses its applicability only to content access applications. The application support is investigated with a group of mobile nodes which together form a community. The nodes in the community correspond to users willing to contribute communication resources, for example, storage and bandwidth, to the other nodes in the community. Some nodes are willing to support cooperative caching and storage by storing messages over longer period of time than required by the DTN bundle layer lifetime. At the same time, some nodes in the community contribute storage only for forwarding purposes like plain DTN routers do. The cooperation between the nodes makes it possible to create a distributed storage from the shared resources. The storage can be used for caching data content previously retrieved from the infrastructure network via gateway nodes. The storage support is discussed in more detail in Section 3.4.

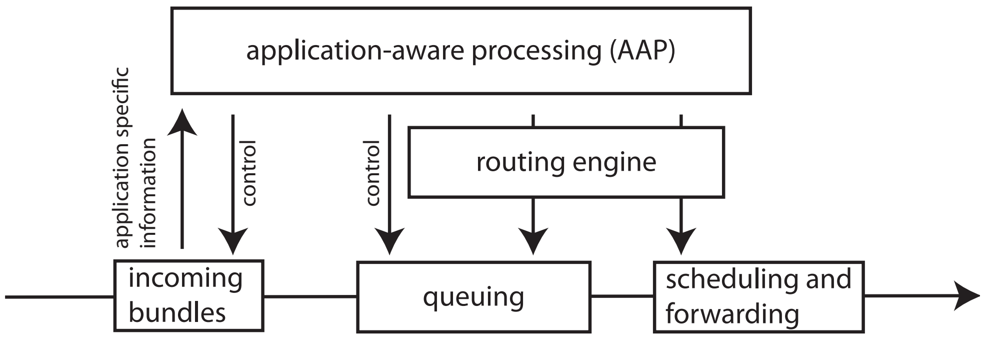
application-aware processing (AAP)
application specific information
control
control
routing engine
incoming bundles
queuing
scheduling and forwarding

Figure 3.2: Simplified view on an application-aware DTN router.

Figure 3.2 illustrates conceptual overview of an application-aware DTN router which is used to implement caching function. An application-aware processing (AAP) module is capable of interpreting application specific information carried in a DTN bundle header extension. The application information can be used, for example, to drop bundles that carry duplicate content already before the content is stored, or to manipulate queuing or forwarding decisions based on the purpose of the messages for the communicating applications. For example, with storage application the small request messages could be prioritized over large bundles in the queue.

The following discusses how the bundle protocol specification [125] created in DTNRG can be extended to support application specific processing in the DTN nodes. A novel header format is introduced that can be used to carry the necessary bundle protocol extensions. The design of the bundle protocol extension block together with the application-aware router design forms the technical basis for the evaluation carried out later in Section 3.8.

Extending Bundle Protocol for Application Specific Processing

In order to provide support for application specific-processing of bundles a new bundle protocol extension block is proposed. The proposed extension is named *Application-Hints*, and it carries information about application semantics of the bundle payload. DTN nodes in the communication path can then interpret the extension and apply advanced treatment on the bundle. The extension block is often small in size compared to the bundle payload and can be encoded in a few bytes plus the URI and optionally the extensions. Thus, the extension block will not cause significant overhead in terms of data transmission. If bundle fragmentation is used, this header needs to be copied into every fragment. The extension header carries application information, which for the most important part contains the following:

Application protocol: The application protocol has to be known to perform resource matching and to enable selective support or other differentiation: a node can support only subset of protocols or give preferential treatment to some applications. The application protocol also defines how responses to retrieval requests are generated.

Resource: The resource, for example, a media content item carried in a bundle or asked for, needs to be identifiable for any matching operation. A URI is used for resource identification. Since resources may change over time, also application-independent version information and a cryptographic checksum of the resource contained in the ADU are included to allow simple equality matching. This also helps avoiding caching two identical copies of the same resource.

Operation type: Independent of a particular application protocol, the general class of operation is indicated to allow for minimal support even if the application protocol details are not understood. The operation can indicate for example, a *request*, a *response*, an *unconfirmed event*, or an *unknown operation type* and whether the bundle contains a resource. The distinction between operation classes makes it possible to apply different storage and routing policies based on the class of the bundle.

Lifetime: If a content resource is contained in the message, its application layer lifetime can be made explicit, so that keeping and discarding bundle can be adapted according to application needs. The application layer lifetime can be extracted, for example, from HTTP *Expires* header [50], which is used for cache control and defines the period of the content validity. This allows to use time-to-live information carried in the DTN bundle headers for forwarding purposes only, and it does not need to be overloaded according to application needs.

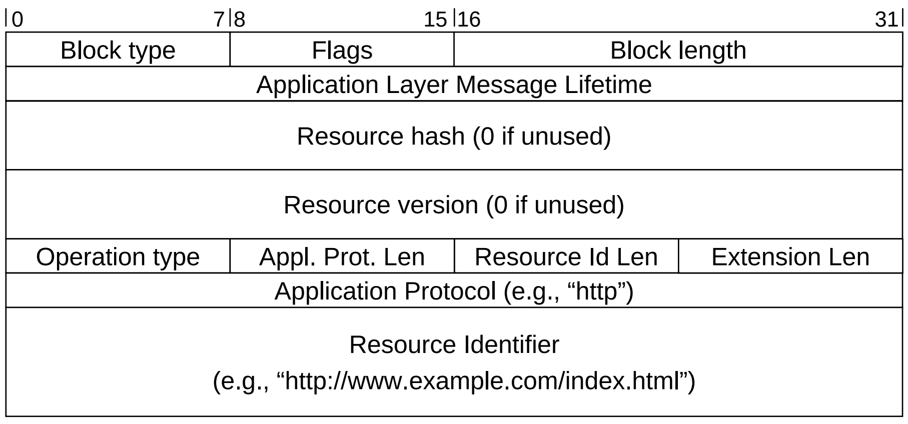
0
7
8
15
16
31
Block type
Flags
Block length
Application Layer Message Lifetime
Resource hash (0 if unused)
Resource version (0 if unused)
Operation type
Appl. Prot. Len
Resource Id Len
Extension Len
Application Protocol (e.g., “http”)
Resource Identifier
(e.g., “http://www.example.com/index.html”)

Figure 3.3: Application-Hints extension block for the bundle protocol.

Figure 3.3 illustrates construction of the Application-Hints extension block for the bundle protocol. The structure of the block is explained in more detail in [101].

In addition, one further *Routing* extension block is defined to the bundle protocol which is independent of the specific application but supports various routing protocols to limit the spreading of bundles. A *hop-count* metric is used to restrict the spreading diameter and a *copy-count* indicates the number of copies that may be created per bundle in order to limit the total load on the network created by a bundle. The former is initialized to the number of hops a bundle may travel and decremented upon reception at each intermediate node. If it reaches zero, the bundle is no longer forwarded. Hop-count limit has been proposed also in an ongoing work [47], which presents a DTN bundle protocol extension block for the taks. The latter is set to the number of copies that can be generated from an instance of a bundle. For example, with binary Spray-and-Wait the number of copies is halved whenever the bundle is forwarded. An initial value of zero indicates that the field is to be ignored.

3.4 Content Caching and Storage in DTNs

This section discusses a design for a DTN node that provides caching and storage functionality based on the application-aware DTN networking approach. The design allows the node to identify content retrieval requests and to serve responses to the requests if the node has resources that match the request.

Figure 3.4 shows a simplified view of a storage-enabling DTN node. The storage/retrieval logic module provides interfaces to perform cache lookups for the resources stored as bundles in the queue or the cache. If a passing bundle contains a request for a content that the node stores, the content is retrieved locally based on an unique identifier. The matching bundle is then packaged into a response message which is originated by the DTN node. If the passing bundle contains a response that carries a content item, the

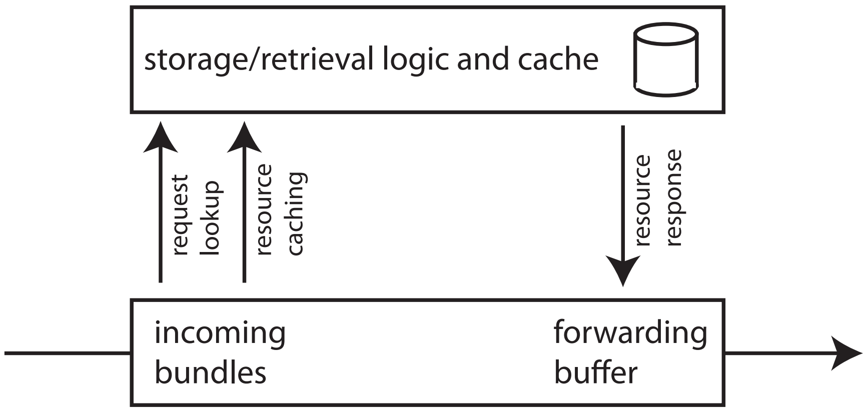

Figure 3.4: Architecture for a distributed DTN storage module.

bundle can be stored locally on the node's storage. The Application-Hints extension presented above exposes the identifiers and the application layer protocol operation to the DTN layer.

The storage-enabling DTN node allows mobile nodes to share and access resources in peer mobile nodes. For a use case of the design, a sparse network of mobile nodes N_i which participate to cooperation by sending, forwarding, and serving responses to requests for identifiable content is considered. The contents, identified by unique identifiers U_M, are packaged as self-contained and identifiable ADUs. The intermediate DTN nodes perform application-aware processing in addition to the usual store-carry-and-forward operation. The nodes are able to modify the local queue management policies to match the demands of the communicating applications.

Making use of the DTN store-carry-and-forward paradigm and enabling the intermediate DTN nodes to keep a copy of a message for a longer period of time than required by the forwarding algorithm allows creating a cache for the ADUs. Separating the cache from the DTN forwarding queue is beneficial from at least two perspectives. First, it allows to store messages according to the application layer validity of the carried content that is often more long-lived that the message TTL used for limiting the bundle forwarding. Second, it allows to manage cached content according to eviction policies, which prefer storing messages that contain likely response to later content requests.

The application-aware nodes can match resource requests $req(U_M)$ to response messages that carry the sought resource $rsp(U_M)$ based on the unique content identifier. When a match is found, they generate a response destined to the originator of the request. Some of the nodes act as Internet gateways and thus are the ultimate target for requests as they represent access to all resources in the Internet. Responses can be then cached by the intermediate nodes forwarding them. In this model, the requests can later be served also from the cooperative cache formed by the community nodes. Generally, a request $req(U_M)$ for a complete resource U_M is sent by a node N_s. The request is forwarded until it hits a node N_d that either has a cached copy of U_M or is a gateway and can thus retrieve U_M from the Internet. Furthermore, Publication III introduces store requests $srq(U_M)$ that can be used to store a resource for later retrieval .

Content Redundancy in DTN Caches

DTN is proposed for communication in environments where delays may grow large and where disconnections and network partitions occur. In these kinds of environments adding *redundancy* can help to increase availability and transfer success of data. The following investigates how redundancy can be applied to bundle forwarding and content caching in DTNs. Different redundancy mechanisms are discussed that increase the response success and the cache hit rate, and decrease the response latency. At the same time, the proposed mechanisms aim to keep network and storage resource requirements suitable for communication environment where resources are limited.

Redundancy of content results in naturally when multiple copies of a content are created in the network. In DTNs, this can result from using a multi-copy routing algorithm, or from multiple entities originating the same data item nearly simultaneously. If a single-copy routing algorithm is used, redundancy can be added through creating multiple copies at source or adding erasure coding. In order to apply erasure coding, it is possible to perform the coding in the application layer and allow parts of the coded message to be transferred as independent bundles. The decoding can then take place in the application layer in the receiving node. This does not require modifications to the DTN forwarding scheme. Then, the existing protocols can be used without changes and functionality can be added only to applications at the end hosts. The use of Reed Solomon codes allows to divide single bundle carrying a resource U_M of size $S(U_M)$ into m source blocks and to encode $n - m$ additional blocks of the same size. Any combination of m blocks out of n of equal size $S(U)/m$ can be used to construct the bundle. As discussed previously in Section 2.3, redundancy can be significantly increased with only a small amount of additional data when erasure coding is used. This helps to avoid wasting both the scarce buffers and the limited contact volumes in a DTN network. Publication IV contains more details on working with blocks, resource identifiers, and coding parameters.

When erasure coding for cached resources is allowed, requests for resource fragments are identified and treated similarly to requests for the entire resource and so are the corresponding response bundles. This common treatment allows intermediate nodes who do not understand erasure coding or fragmentation to reply to requests if they hold a copy (or fragment) of the resource in question. In addition, the nodes which do not have functionality to perform the look-ups in the forwarding buffer or cache may still participate with plain forwarding functionality. If a match for one fragment is found, the reply is generated and the request for this fragment is dropped while the requests for other fragments are forwarded further.

3.5 Message Fragmentation in DTNs

In DTNs, messages can be large compared to transfer capacity of available contacts between adjacent nodes. Large messages lead to longer transfer times making it more likely that contacts break in the middle of message transfers. This motivates investigating how to support partial message transfers by using fragmentation. This section formulates fragmentation independent of routing algorithms and introduces several strategies for using fragmentation in DTNs.

The DTN architecture allows arbitrary message sizes, but does not define a maximum transfer unit (MTU). However, since messages can be large, the *contact duration* can easily become the limiting factor for message forwarding. For example, assuming fast contact establishment, it takes some ten seconds to reliably transmit a message of 1 MB over a 1 Mbit/s wireless link. Thus, contacts can be too short–lived to even transmit just a single message and support for partial message transfer can be essential to make communication possible over short-lived links. Partial message transfer also avoids wasting link capacity with incomplete transmissions. This motivates fragmentation for DTNs.

The DTN architecture [20] defines two types of fragmentation, which are discussed in this section.

Proactive fragmentation works so that a node splits a bundle into fragments *before* transmitting it. This approach is similar to IP fragmentation. The fragmentation decision can be made based on knowledge of the contact availability or according to buffer limitations on the next hop.

Reactive fragmentation works so that the forwarder starts transmitting an entire bundle and, when interrupted by a sudden contact failure, the forwarding node and the next hop arrange the remaining portion which has not been transmitted yet and the already received portions into valid bundles. The agreement on how much data was successfully sent can be achieved by a suitable convergence layer, for example, the TCP convergence layer [36].

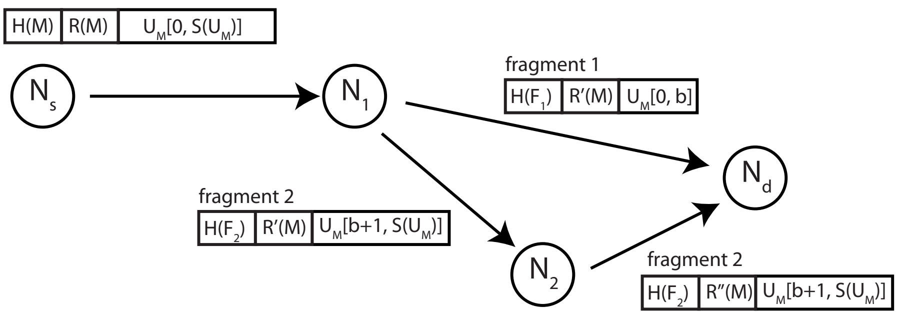

Figure 3.5: Message fragmentation in a DTN host.

Figure 3.5 illustrates a simple example of fragmentation. A message M carries message header *H(M)*, which contains control (source, destination, ...) and payload-related (size, offset, granularity, checksum, ...) information. A routing-specific header *R(M)* carries information that is used by the routing protocol for forwarding and deletion decisions (time-to-live, custody, ...) The payload U_M contains the application data and its size is noted as $S(U_M)$. During the process of fragmentation, a node N_i divides the bundle payload, which can be the complete resource or a resource fragment, into two or more non-overlapping fragments $F_1, F_2, ..., F_M$.

A fragment of the original payload is denoted as $U_M[a, b]$ which indicates the inclusive offsets for the start at a and the end at b, and is of size $b - a + 1$. The node performing the fragmentation copies H(M) and R(M) into each of the resulting fragments. The fragmentation process updates the fragmentation offset and size in H(M) for each fragment, but does not modify R(M). The forwarding process will update R(M) according to the routing protocol but will not change H(M). Moreover, the fragmenting node will not include any information about itself in the resulting fragments.

The proposed approach treats fragments as independent bundles in the network intermediaries, and reassembles them only at their final destination. Reassembly can be carried out also in intermediaries, but because of complexities related to managing routing information this is limited out of scope in this thesis. When the destination node performs the re-assembly, fragments are collected and maintained until their time-to-live (TTL) expires. The TTL is defined in absolute time in the sending node and copied at the time of fragmentation, and thus is identical for all fragments. With multi-copy routing algorithms, several fragments can overlap.

The message is considered delivered only when the destination can recover the entire resource from the received fragments.

Proactive Fragmentation

Proactive fragmentation can occur at any node. The decision to fragment is made *before* message forwarding to the next hop and can be based, for example, on estimations about expected uptime of the contacts to along the path to the destination. A special case is *source fragmentation*, which is performed by the source node N_s. The node passes the resource from the application to the bundle layer, where the message H(M) and routing headers R(M) are created. Then, node splits the payload into m non-overlapping fragments of equal size, to which the headers are attached.

After the fragments are created, the node sends them out sequentially, starting with the beginning of the payload. Intermediary nodes forward the fragments without modifying them, and no further fragmentation takes place in the network. At the destination node N_d, all m fragments are needed to re-assemble the message.

This thesis considers proactive fragmentation only at the source. It has also a beneficial property that the fragmentation boundaries can be matched with data authentication boundaries in the sending node [105]. Reactive fragmentation, not relying on uptime estimates for the contacts, seems more suitable for fragmentation at the intermediaries

Reactive Fragmentation

For reactive fragmentation, any two nodes in the network are able to fragment a message with a payload larger than a single byte. Fragmentation is triggered when a connection breaks during a message transfer from N_i to N_j. From the message headers, which are assumed to be transmitted first, and the payload part of the message received by N_j before the connection breakdown, one fragment F_1 is created. Another fragment F_2 is created from the part of the message not yet delivered to N_j. Both fragments inherit the headers from the message, with the control information adjusted. Routing information $R(M)$ of the forwarded part F_1 is updated to reflect the successful forwarding and it remains unchanged for the other part F_2. The unfragmented bundle at the forwarding node is deleted to free up resources, and the two fragments continue to exist independently. This allows maintaining different routing headers and keeps fragmentation independent of the routing protocols. Fragments may be fragmented again in the following intermediate nodes.

This thesis also investigates a version of reactive fragmentation, which limits the size of the resulting fragments to pre-determined boundaries defined by the originator. This is often referred to as *toilet paper* approach [49], and allows, for example, for fragment authentication [105]. Fragmentation boundaries also increase the probability that fragmenting copies of a single resource in different paths result in identical fragments. As only identical fragments are detected as duplicates, the pre-determined boundaries help duplicate detection and thus enable to reduce redundant traffic.

In practical implementation of reactive fragmentation, the forwarder and the next hop first establish a suitable convergence layer connection, for example, TCPCL [36]. Then, the forwarder starts transmitting an entire bundle so that the bundle header is sent first. When interrupted by a sudden link failure, the convergence layer gives information for the forwarding node and the next hop to infer how many bytes of the bundle were transmitted successfully. The sender and receiver then agree the transmitted and remaining byte range portions into valid bundles and append them with an adapted protocol header.

Fragmentation and Routing

The introduced message fragmentation mechanisms work with the existing DTN routing scheme. The basic fragmentation information is part of message header *H(M)* and common across all routing protocols, whereas routing headers *R(M)* are protocol-specific. Fragmentation takes place in the queues (of the forwarding node and the next hop), which minimizes the interaction with routing.

For proactive fragmentation, the source creates M fragments instead of a single message. The fragmentation does not require routing-specific actions, because complete messages and fragments do not differ from a routing point of view. Both types of reactive fragmentation can operate without changing routing in the single-copy algorithms, part of the message F_1 is simply passed to the next hop and the other F_2 is kept at the forwarding node. In multi-copy routing schemes the fragmentation process removes the message M in the forwarding node and replaces it by two pieces (F_1 and F_2), which inherit the forwarding properties of the original. After the fragmentation, both fragments (F_1 and F_2) will be forwarded in their own and their routing headers will be updated independently. This is not relevant for *Epidemic* and *Prophet*, but routing protocols such as *Spray-and-Wait* and *MaxProp* require message-copy specific routing information. As node pairs operate independently, the proposed approach can result in a mixture of overlapping fragments of different sizes, which are only detected as duplicates if the byte ranges of the fragments match exactly.

Expected Effect of Fragmentation

Fragmentation is expected to have both positive and negative effects on the message delivery ratio. As motivated earlier, fragmentation is beneficial when the contact durations become the limiting factor for message delivery. Without fragmentation the shortest contact duration along a path would determine the upper size limit for messages along the path. With proactive fragmentation at the source, the upper limit no longer applies to the message but to individual fragments. However, it is still difficult to estimate the maximum acceptable size for fragments in opportunistic networks. In contrast, reactive fragmentation allows to match the per contact limit along the path as the contacts break, and thus enables to maximally benefit from available communication capacity. In both cases, this aspect of fragmentation is expected to increase the delivery ratio, because fragmentation allows exploiting shorter contact opportunities and decreases the fraction of unsuccessful message transfers due to interruptions. Thus, fragmentation is expected to increase the connectivity in the DTN networks.

However, messages in DTNs often exhibit poor delivery ratios, particularly in opportunistic networks. If a message is split into two fragments, both need to arrive for successful message delivery. If either of them is lost, so is the entire message. Since for a fragmented message two or more paths are required to deliver a fragment, the delivery ratio of the message is expected to be lower than without fragmentation, see Publication VI for more details. The bundles are also subject to congestion-induced deletion at each node. Thus, the delivery ratio is expected to decrease the more hops have to be traversed and the more fragments are to be created. Moreover, a large number of fragments requires also a large number of message headers to be forwarded, which increases protocol overhead. Further insight on the effects of fragmentation are provided based on findings of evaluation carried out for this thesis in the seventh subsection of section 3.8.

3.6 Searching for Content in Mobile DTNs

Users often have an idea what content they want to retrieve, but do not know an exact identifier or locator to feed into an application for the resource retrieval. A search engine helps by mapping search queries to resource locators that point to content, which likely includes the sought information. However, accessing a dedicated search engine is not a desirable approach in challenged networks, since it introduces additional network and service dependencies. This section discusses content search in mobile opportunistic networks, and proposes mechanisms that can directly search from other mobile nodes in the vicinity of the user.

A traditional approach to map search queries to resources is implemented by maintaining an index about the resources. The index contains keywords and maps them to sets of related resources. This can be used to map a query consisting of several keywords to a list of suggested search results. Examples of centralized search engines that use such indices are Google and Bing. The problem with such centralized services is that it is expensive to keep the index up-to-date and the assumption is made that the clients can reach both the index and the locations where the contents reside. There are approaches for managing decentralized indexes in peer-to-peer environments, but these models often assume good connectivity among the nodes. Otherwise, unreliable connections may lead to a high maintenance cost of the index.

Furthermore, all content is not just on servers connected to the infrastructure network, but also mobile users produce content that they are willing to share with others. Particularly in disconnected environments, this content cannot easily be indexed by centralized search engines unless uploaded to some server. Moreover, as discussed in Section 3.4, content that is carried in the caches by the other mobile nodes in the proximity can be a potential source for responses. This content may be especially interesting to other mobile users in proximity, which could access it by means of opportunistic networking.

This section investigates support for searching and retrieval of contents in mobile DTNs. A simple notion of content queries is introduced and it is used to discuss how nodes should process and forward queries and respond to them returning matching locally available contents. Nodes carry content items, which in the context of the web are also referred to as *resources*, that can be matched by a local search function against incoming queries generated by other nodes. They may respond to and/or forward the query to one or more other nodes. When responding, a node can choose to return all or a subset of the matching local contents. The discussion limits to searching the mobile ad-hoc environment and does not discuss interaction with infrastructure networks, which are discussed more in the next section and in Publication VIII.

Searching in Opportunistic DTNs

The following discusses search in mobile opportunistic networks and describes a model, which is used to analyze the problem of limiting the search coverage and the number of returned responses. Alternative solutions are introduced based on the DTN bundle protocol [20, 125]. In the following scenarios, some devices in the network store content that they are willing to share with others. Like in earlier sections, nodes are willing to cooperate and supply a limited amount of their local resources to assist other nodes. The goal is to allow users to spread queries for content that is stored on other nodes anywhere in the network and to provide methods for controlling the chances to obtain the sought information. To facilitate searching, an assumption is made that nodes are able to perform searches on their local storage and find the relevant results for a given query.

This section uses the following notion to introduce the search problem (further notation on the problem can be found in Publication VII). N is used for the set of nodes, C stands for the set of all content items that are available in the network, and Q marks the set of queries issued over the lifetime of the network. The mapping from a node to its stored content items is defined by $s(n) \subseteq C$, for $n \in N$. The function w maps queries to the content items they match $w(q) \subseteq C$, for $q \in Q$. Function $rsp_{local}(q, n) = w(q) \bigcap s(n)$ defines the response generated by a node n when replying to query q. Finally, the function rsp marks the responses for a query seen by a given node: $rsp(q, n) \subseteq w(q)$, for $q \in Q$ and $n \in N$. This includes the locally generated responses.

The processing of a query can be divided into three phases. During a *local matching* (i) phase nodes execute the query searching for content in their local storage and generate a set of all matching resources, from which the *selection* function picks a subset to return. Depending on whether or not results were found, the node runs a *query processing* (ii) function that decides about forwarding the query or about search termination, as well as modifying the query. Finally, *response forwarding* (iii) is responsible for routing and forwarding locally generated responses and those received from other nodes.

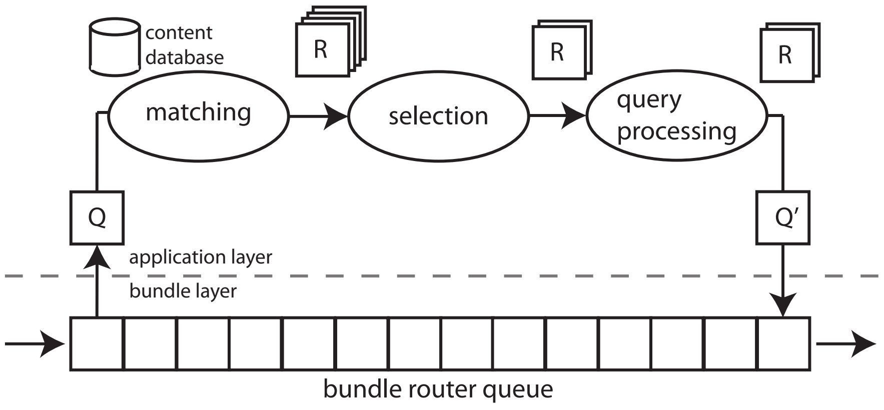

Figure 3.6: Query processing in a DTN node.

Figure 3.6 illustrates the query processing flow in a search enabled DTN node. When a query arrives at a node, the matching logic retrieves all content items by executing the search on the local content database. The matching resources and the original query are passed to the selection logic, which selects a subset of the content items to return. This decision can take into account information carried by the query (for example, the number of resources returned by other nodes) or be based on local policy (for example, random selection). Finally, the resources and the query are passed to the query termination logic which chooses to terminate or forward the query. The original query may be modified by adapting information, such as the number of returned resources, before passing it to the bundle layer for forwarding.

The simplest strategy to distribute both queries and responses is flooding. However, flooding consumes often too much resources, which motivates fo limit the use of scarce system resources to perform a search. Ideally, a query should only be forwarded to neighbors that hold matching contents, or are on the path to other nodes that have the sought resource, and different nodes should return non-overlapping responses to the searcher. As global knowledge or active coordination is not an option in discussed scenarios, each node can only select next hops for queries in such a way that the system resources used on a search are balanced against the probability that a useful results will be received. Similar limitations apply to local content selection and routing responses.

The following subsections discuss effective controlling of response matching and selection, query forwarding and termination, and response forwarding. The emphasis is on the *query forwarding and termination* part and the other two are addressed only briefly. Focus is on strategies for limiting the number of copies of a query that are distributed and the number of responses that are returned. However, the limits can be defined in greater detail. Instead of just limiting the number of copies to be distributed for a query, the sender can specify, for example, how many bytes of results are to be returned. This can delimit both the consumed storage space and the required bandwidth. However, these aspects are out of scope of the discussion in this section.

Local Matching and Selection

A wide variety of systems to formulate search queries exist, most of which depend on assumptions about the format of the content to be searched. While it is desirable for a general protocol to support many different types of queries and contents, this work abstracts from the actual matching process and focuses on what is transmitted over the network. The related evaluations use a simple keyword matching to substitute for the actual execution of queries.

When a node has found the set of matching content in its local storage, it may wish to respond with only a subset of the results. This can be done to limit the amount of resources used both locally and globally for transmitting and storing the responses and to limit the number of out potential duplicates. The query originator may assign this limit based on the number of responses or the volume of response-data that should be generated. This limit is encoded in the query and each node then subtracts the number of results responded from its local storage and never returns more than the allowed number of responses. Before forwarding, the respective field in the query message is adapted accordingly.

If the local matches exceed the reply limit, the node needs to select which ones to forward. When the matching process yields a score for the relevance of the results, this can be used to choose the elements of the selection set. If this is insufficient, for example, if only a single attribute was used in the query, the resources to be returned can be chosen at random. The probability of including a resource can be then, for example, inversely proportional to its popularity, so that the chance for the searcher receiving only duplicates of the most popular content is minimized.

Query Processing

The evaluation part of this thesis investigates four alternatives to limit the spreading of queries. As a *network level* mechanism *hop-count limit* and a limited *time-to-live* (TTL) are used. Two *application level* mechanisms are a simple *first matching response* drop and a termination based on *global response count estimate*, which records information along the path of the query message. The first two approaches can be implemented completely at the bundle layer without any input from the application layer. First response drop and global response count estimates require application layer logic but the former is based on explicitly known information while the latter requires estimation of a number of parameters.

When using a hop-count limit, the query is discarded once it has been forwarded L times. The limit L can be chosen by the originator of the query or it can be selected by the network, even by a node-specific policy in which case the hop-count limit on different paths can vary. The count of hops $hc(q)$ a query q has traveled when arriving to node n defines the forwarding condition as:

$$forward(q,n) = \begin{cases} true & \text{if } hc(q) < L \\ false & \text{otherwise} \end{cases}$$

The time-to-live limit $ttl(q)$ of a query q is a similar approach in which the query is forwarded for a period of time specified by the originator, independent of the number of nodes visited. Under this scheme the spread of queries is likely to differ significantly depending on the node density of the scenario. With t_{now} as the current time and $t_{creation}(q)$ being the time when the query q was created, the following defines the forwarding condition:

$$forward(q,n) = \begin{cases} true & \text{if } (t_{now} - t_{creation}(q)) < ttl(q) \\ false & \text{otherwise} \end{cases}$$

When using a single-copy routing scheme such as First Contact, the spread of queries with both hop-count and TTL limits is essentially a random-walk, where the next-steps are determined by the motion of the nodes and the rules defined by the routing scheme. When used with replication-based routing, the queries are distributed in the form of controlled flooding.

The first response drop scheme forwards the query if no local match was found and terminates the search otherwise. This means that a dependency on the local contents is introduced which impacts DTN routing.

$$forward(q,n) = \begin{cases} true & \text{if } (rsp_{local}(q,n) = \emptyset) \\ false & \text{otherwise} \end{cases}$$

Finally, a search termination function is formulated that takes into account further information recorded along the path of the query message. While two-way coordination between nodes is not feasible in opportunistic scenarios, it is possible to pass information one-way along with the query message. The bundle layer is assumed to pass hop-count and transit time information to the application layer, while the response count can only be recorded at the application layer.

The goal of this search termination mechanism is to limit the spread of the query as the number of responses grows. In other words, the marginal value of returning additional responses to the originator of the query is assumed to decreases when the total number of responses increases. This can be justified by at least two arguments. First, the likelihood of additional responses being unique decreases (due to the birthday paradox). Second, the value of additional unique responses decreases (the user might have already received an acceptable response).

As each node has only a limited view of the network, it needs to estimate the number of responses that the query originator has already received. In order to make this estimation the node can guess the number of nodes that have seen the query globally and the number of responses each of those nodes has generated. For this, an estimator function $u_{est}(q,n)$ is defined that is directly proportional the number of responses generated globally to the query q based on knowledge at the node n. After constructing $u_{est}(q,n)$, it can be used to terminate the query when its value exceeds some threshold value T. This is stated as:

$$forward(q,n) = \begin{cases} true & \text{if } u_{est}(q,n) > T \\ false & \text{otherwise} \end{cases}$$

In order to construct $u_{est}(q,n)$, the number of nodes that have received the query is estimated. To achieve this, a hop count metric hc is recorded in the query which is initialized to 0 at the querying node and incremented at each hop to determine the number of nodes a query has already traversed. From this, the number of nodes $nodes_{est}(q,n)$ potentially having received a copy of the query after up to hc hops can be estimated as a function of the routing protocol. For single-copy routing protocols, $nodes_{est}(q,n) = hc(n)$. For multi-copy routing protocols with a fixed number of copies K, $nodes_{est}(q,n) = K - K(q)$, with $K(q)$ indicating the number of copies the received message represents. For epidemic routing protocols, every node on the path records its own *node degree* estimate ($degree(n)$), which it calculates as a moving average of the number of contacts it had during the last $ttl(q)$ seconds. This represents the possible replication factor at this node for the query, yielding a local approximation of the number of nodes reached by the query can be defined as:

$$nodes_{est}(q,n) = degree(n)^{hc(q)}$$

The number of responses generated per node can be estimated by recording in the query message the total number of responses, $rsp_{tot}(q)$, generated by nodes along the path. If no responses have been generated along the path, a static estimate C can be used instead. The estimate can be either pre-configured or calculated as an average from past queries. Based on the above, the global number of responses is approximated as:

$$u_{est}(q,n) = \begin{cases} nodes_{est}(q,n) \cdot \frac{rsp_{tot}}{hc(q)} & \text{if } rsp_{tot}(q) > 0 \\ nodes_{est}(q,n) \cdot C & \text{otherwise} \end{cases}$$

There are a number of assumptions in the above approach that should be considered. First, it is assumed that the global node degree distribution has an existing mean and that the node's measurement of its own degree is a good estimate for the average degree in the network. Should the node degrees be, for example, power-law-distributed, the nodes that have very large node degrees will drastically overestimate the number of nodes that the query has been distributed to and are therefore less likely to forward the query. Second, since $node_{est}(q, n)$ is dependent on the routing protocol the application layer must be aware of the underlying routing and the routing must be uniform across the entire network. Nevertheless, the above can be considered to be an estimate based on the limited knowledge available to the nodes.

Response Forwarding

After a node has locally found matches for a search request, it needs to transfer the results to the searching node. No search-specific functionality is needed, but it suffices to simply rely on the underlying routing protocol to deliver the reply messages to the originator of the query. This may lead to two types of duplicate responses.

First, when using multi-copy routing protocols, even replies from a single responder may get duplicated and reach the querying node via different paths. The duplication can occur even if the routing protocols suppress forwarding of duplicates in every intermediate node. This is inherent to the nature of the routing protocol so that further optimizations would need to further improve the routing protocols themselves

Second, identical or partly overlapping replies from multiple responders may reach the searching node. The above response selection mechanism attempts to reduce such duplicates heuristically and the search termination mechanisms helps by limiting spreading the query. However, overlapping responses cannot be prevented from being generated by different responding nodes. This duplication occurs at the application layer and only there the message semantics (that is, being identical/overlapping) are known. At the DTN routing level, messages are typically identified by the pair of originator and destination address and a locally unique identifier assigned by the originator. This makes messages from different responders to appear as different ones to the DTN routing and are thus not suppressed as duplicates.

Extensions are possible to explicitly identify the contents of messages and the query they respond to. The extensions can make the response semantics visible to an *application-aware* routing layer, as partly suggested in [138] and in Publication III. This could aid application-aware nodes to suppress semantically duplicate messages. Furthermore, the limited number of duplicates are not necessarily negative in DTN routing as they may increase the delivery probability.

3.7 Opportunistic Web Access via WLAN Hotspots

This subsection investigates content retrieval in a realistic urban scenario, where WLAN access points act as gateways between an opportunistic DTN network and the fixed Internet infrastructure. The scenario allows investigating how infrastructure nodes add partial structure to the network. This partial structure and implicit location of hotspots enable advanced response delivery towards the area where the content request was originated. Moreover, opportunistic DTN connectivity through co-located mobile nodes enables web access for the nodes that are not able to connect directly to hotspots.

Using WLAN hotspots to provide web access to mobile users can be appealing when using the cellular network is not feasible or even possible. This kind of conditions can occur, for example, in rural conditions or when the high roaming costs prevent using the cellular data connections while traveling abroad. As a solution to limited infrastructure support, using the WLAN hotspots spread across urban areas is proposed for communication between the mobile nodes and the infrastructure network. This is possible, since an increasing number of WLAN community networks, for example FON or Wippies, offer an opportunity for Internet access. While they often are not as thoroughly administered as cellular networks, they offer high access rates, exhibit short RTTs, and are usually underutilized. This makes them a suitable platform for mobile web access, albeit with the shortcomings that their coverage is very limited and access is usually constrained to subscribers or community members. This section shows how to use opportunistic delay-tolerant networking to extend the reach of WLAN hotspots to users that are not authorized to directly access them or are out of their radio range.

The presented approach largely solves the limited coverage of WLAN-based access for users and applications that can tolerate some delay in receiving a response. The delay may be result of user having to wait until an access point comes into reach as a result of the user's mobility. Alternatively, the user may have to wait until messages can be opportunistically relayed through other nodes to the infrastructure network. Relaying through the other nodes further helps when all the users are not authorized to access the network, but can indirectly forward messages via other users. The indirect access is feasible because of common flat rate tariffs and the high access rates, which are usually not exhausted by the subscriber's device alone.

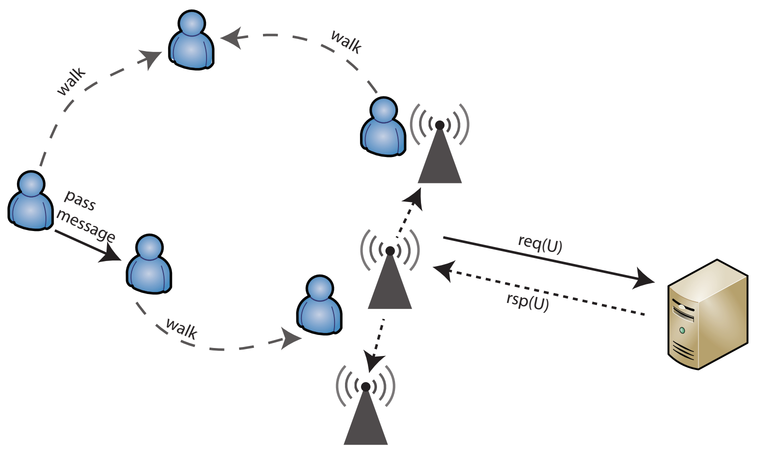

Figure 3.7: An example communication scenario for web access.

Figure 3.7 illustrates an example communication scenario consisting of several mobile users in an urban area. The mobile users' devices are capable of communicating with each other opportunistically when they come into radio range. When in the coverage of a WLAN hotspot, the mobile nodes can connect to the Internet via the access point that provides DTN routing and an optional caching functionality. Geographically neighboring WLAN access points may cooperate to replicate responses from the Internet to reach mobile nodes who may have moved in the meantime. Mobile nodes and access points holding a copy of the sought content may reply immediately to requests. This is enabled by extending application-aware routing functionality in the nodes to support the mechanisms discussed in Section 3.7. The design enables a DTN messaging overlay among the mobile nodes, the hotspots, and the Internet.

An additional benefit is that the design offloads data belonging to delay-tolerant interactions from the 3G network to the closest WLAN access points. Thus, it can help to preserve cellular connectivity for real-time and more interactive applications. Several applications, for example, fetching RSS feeds, P2P file retrieval, or posting to blogs do not require real-time interaction, and thus are good candidates to benefit from the presented opportunistic web access model. Moreover, the cellular network can also serve as a backup if DTN-based communication does not yield a result in the expected time frame. This enables the complementary DTN overlays to support and enhance pervasive information access for mobile users.

This section expands the work discussed in the previous subsections, and the earlier work by Ott and Kutcher [102]. The caching friendliness of the message-oriented communications in DTNs is leveraged to allow investigating the impact of opportunistic caching in both access points and mobile nodes. This expands the earlier work on cooperative caching in mobile opportunistic networks, discussed in the publications III through VII, and introduces partial structure to the network by using static hotspots that have a fixed geographic location.

Opportunistic Web Access and DTN-based Resource Retrieval

In the Internet, web content is accessed by sending one HTTP request per web resource and receiving the related response with a message body that contains the resource. Multiple request-response pairs are usually required per page, because typical HTML web pages contain multiple embedded objects, for example, images. This iterative models leads to a retrieval delay, which depends, for example, on number and size of the objects and the round-trip time between the client and the server. This type of retrieval is only practical if the endpoint is continuously connected to the Internet, otherwise retrieval of some objects fails and the requested page cannot be fully displayed. Furthermore, low round-trip times are required for acceptable retrieval performance or the iterative process will take too long. Finally, while HTTP is stateless and can support asynchronous communications, the current web applications rely on TCP and thus require end-to-end connectivity to the origin server.

The naming scheme in the DTN architecture allows for a straightforward mapping of HTTP URIs to DTN addresses and define web-traffic-specific routing rules [48]. The arbitrary size of bundles allows them to be self-contained, that is, carry either a complete request or response. In particular, this allows to avoid iterative retrieval as a self-contained web page including all embedded objects can be contained into a single message. The encapsulation of HTTP requests and responses follows the approach originally described in [102]. Multipurpose Internet Mail Extensions (MIME) [55] encapsulation is used for HTTP messages. It provides content identification (*message/http*) and allows for aggregating multiple pieces of content via Multipart MIME. The latter is needed for combining all objects embedded in a web page into a single response. The conventions of MHTML [104] are followed and the *Content-Location* MIME header is used to identify the individual objects inside the message body.

To access a web content item, for example, all items of a single web page, a DTN-enabled web client issues a single HTTP request. The request is contained in a DTN bundle by using MIME encapsulation in the client application. The bundle destination is set to an address of a DTN-enabled web server and passed to the DTN router in the client node. The message is then forwarded in the network towards the server, and any intermediate mobile node or access point holding the requested resources may reply and stop forwarding the request further. If the request reaches the DTN-enabled web server, the server obtains all objects of the requested web page and encapsulates them in a single MIME message to create a response. The objects to be included are assumed to be known from web authoring tools and dynamically generated content on the server side [102]. The server may use end-to-end authentication for bundles or Secure MIME (S/MIME) [116] to prove authenticity of the contents. The response is then contained in a DTN bundle and its destination address is set to the source address of the request bundle. The response message's time-to-live (TTL) is set to the value seen in the request bundle to limit the spread of the response to vicinity of the user. When the client receives the response, it extracts the individual objects and renders them according to the *Content-Location* headers.

Routing, Urban Mobility, and Geographic Awareness

As described above, the urban web access scenario consists of three types of nodes; web clients, web servers, and access points. First, web clients installed in the mobile nodes issue requests for web resources and use opportunistic contacts to forward them towards the web servers. Second, web servers in the fixed network are distinguished by their EIDs so that the node type of the bundle destination can be identified. Third, access points in hotspots have the property that they are all interconnected on the fixed network side and thus any access point can serve as a default router for bundles destined to a web server. Moreover, often the access point forwarding such a request is also a reasonable candidate for delivering the response to the mobile node. Thus the DTN routing is split conceptually into two parts, which are routing in the fixed network between the access points and the DTN web servers, and routing in the mobile network between the mobile nodes and the access points.

For the fixed network side deterministic routing is used between the access points and the web servers. In the simple case, upon receiving a request bundle the access point resolves the services and the location of the web server based on the destination address of the bundle. The target web server can be determined, for example, by using the destination address for performing a Name Authority Pointer (NAPTR) record [98] lookup, which returns the server's IP address, the protocols, and the services available services available via those protocols. For example, the lookup result can indicate that the target web server is available in a given IP address by using the DTN Bundle Protocol. This information can be used to establish a TCPCL [36] connection between the access point and the web server. The connection is then kept open and implicitly a reverse route for the response to the access point is established. In the more complex case of a routing overlay in the fixed network consisting of many bundle routers (including the web server), link-state routing [34] can be used within the overlay to forward the bundles based on the destination EID.

In both cases, an access point receiving a request from a mobile node needs to ensure that the response is routed via itself. No such mechanisms readily exist in the bundle protocol or its extensions. One option would be to create a bundle *return routing* extension block and define a loose source routing mechanism that is then to be employed for routing bundles back to the source EID of the request. Such mechanism would allow the hotspot to insert itself along the path of all web bundles from the server to the client. Alternatively, the access point might change the source EID to include itself as a path element [48] so that a response to the source EID would automatically yield the desired result. However, this could invalidate signatures applied to the bundle and eliminate duplicate detection at the server if bundles are forwarded by different access points. A similar duplicate detection problem occurs if the request from the web client is terminated in the access point, which then generates a new request with its own source EID and creates a local mapping between the original and its own request for later forwarding of the response.

For routing between mobile nodes to and from the access points the probabilistic routing approach is used, because it is not possible to keep consistent reachability information in a disconnected environment. Numerous probabilistic *single-copy* and *multi-copy* routing protocols were developed in the past. Out of these, three are chosen that are stateless, that is, they do not make forwarding decisions dependent on local per node state, for example, contact history between nodes.

A variant of *Direct Delivery* [73] only forwards a message directly to the first access point encountered, not relying on other mobile nodes for relaying. With *Spray-and-Wait* [134], the requester, or the access point issuing a response, creates no more than a fixed maximum number of copies and hands them to other mobile nodes for indirect delivery. *Epidemic* routing [141] results in bundles being replicated to every encountered node, thus flooding the network. In addition, a variant with a supplementary hop-count limit is used to restrict the flooding process to the vicinity of the sending node, since the proposed approach seeks quick replies and aims to avoid overloading other areas of the network. All three protocols discard bundles when their TTL expires. Access points are configured to accept messages on behalf of the web servers, so that all three probabilistic routing protocols will deliver requests through them. All DTN nodes will detect duplicates based upon the bundle identifiers and discard them, avoiding flooding a server with dozens of identical requests.

Routing bundles between a mobile client and a fixed server is asymmetric. Requests from a mobile node may utilize any of the available access points to reach the server. In the opposite direction, the mobile nodes needs to be located, which is especially difficult if the request was delivered indirectly via other mobile nodes to the access point since all the nodes including the requester may have moved.

To increase success of response delivery, *geo-aware* optimizations are proposed. As a first approximation, the responses are routed back to the *forwarding access point.* Given sufficiently short processing and transmission time (a few seconds), this will generally cover the case that the mobile node communicates directly with an access point when sending the request. But even for requests that reached the access point indirectly, this seems to be a reasonable approach. If users will tolerate some response delay and that they do not move very fast, they may still be found somewhere in the access point's geographic neighborhood.

To extend the reach of responses further and increase the likelihood of indirect response delivery, a notion of response routing via the *k-closest access points* is introduced. The web server returns the response to the forwarding access point, which then replicates the response to its $k-1$ geographically closest neighbors. This can be achieved, for example, by using an online database in which hotspots are registered, since connected hotspots can rely on centralized coordination. Each of the k access points then forwards the response to mobile nodes in reach according to the routing protocol in use. This approach spreads the messages around the area from where the request originated, attempting to cover the entire area where the client may have moved in the meantime.

Both approaches use geographic information to return the response to where the request originated from, but do not require explicit location awareness in mobile nodes, nor do they rely on dynamic exchange of location information, but only use relatively static data.

Caching

Using caching is feasible for HTTP because many HTTP responses are identical irrespective of the requesting node. To make sure that only suitable replies are generated from caches, the replying node needs to validate that the requester's capabilities, for example, as indicated in the various *Accept* headers of HTTP, match the content types and encodings of the cached responses. Also, requests should not be handled from caches if cookies are used. For simplicity, simple web clients and web pages without cookies are assumed to limit the scope of discussion in this thesis.

The caching optimizations are applied to mobile nodes as well as to access points. The mobile nodes may yield retrieving web resources faster and without any nearby access point. The access points make perfect caches since they are mains powered and can easily be supplied with extra storage capacity. Moreover, they naturally serve as aggregation points for requests, and can thus help further reduce the workload on the servers. Mobile nodes only use the routing-protocol-specific queue management, and the access points keep copies until their TTL expires.

3.8 Simulations for Performance Evaluation

This section contains performance evaluation of the DTN mechanisms presented earlier in this chapter. The section starts by outlining the used simulator tools, the node and community models, and the mobility models of the nodes. Then, evaluation related to each mechanism presented in sections 3.3 through 3.7 is provided under its own subsection.

The evaluation in this chapter uses network simulators to investigate feasibility of the several proposed mechanisms that were presented above. Three distinct simulators were used during the course of the work that are listed in the chronological order of their usage; *dtnsim* [73], *dtnsim2* [42], and *ONE* [82]. The first two tools were available when the evaluations for this thesis were conducted; and with support for basic DTN concepts and several routing protocols they provided a starting point for the evaluations. During the course of the work, the third was chosen as it gained momentum in the research community and new features were constantly added to it. This allowed to experiment with wider range of simulation parameters, for example, larger number of routing protocols were available. All three are publicly available Java-based simulators that offer possibility to extend the router and protocol design to implement the needed mechanisms. Moreover, all of them follow the semantics of the bundle based communication as specified in the DTN architecture [20].

Node and Community Model

In the following simulations, each node presents a user participating in cooperative communication by contributing communication and storage resources. We assume that the nodes form contacts with other nodes, when they move to radio range with each others. The nodes communicate through bidirectional wireless links with 100 ms delay and a capacity of 2.1 Mbit/s. The nodes transmit bundles in an all-or-nothing fashion when opportunity to communicate, that is, contact, appears. If a contact breaks before the transfer is complete, the bundle is dropped on the receiving side and kept for retrying on the sending side. Partial message transfer through fragmentation is used only in the simulations related to Publication III and Publication VI.

In the following simulations several different community models are used to investigate the effect of caching. Community members are assumed have shared interest in content that they access from the network. Moreover, the model about mobile community has evolved to more realistic while the knowledge has accumulated during the course of this work. Thus, the scenarios are likely more realistic towards the end of this thesis. An overview on each simulation scenario is given in this section before highlighting the most important findings. More details on simulation configurations as well as additional findings can be found in the publications III through VIII.

Real Mobility - MIT Reality

Publicly available mobility traces are used in many of the following simulations, the traces were obtained from the CRAWDAD public on-line directory. One of the traces, called *MIT-Reality* [44], provides behavior of 100 users at MIT university recorded during the academic year 2004-2005. The following simulations, use the “devicespan” information which provides contact times between Bluetooth devices carried by the observed users.

The connectivity periods are recorded by using mobile phones, which fits well the scenario envisioned for deployment of the design.

The MIT-Reality trace describes the behavior of 100 real mobile users during their daily, weekly, and monthly routines. The trace provides an example of a mobility model with real community behavior and sparse connectivity. The nodes meet based on social patterns and end-to-end paths between the communicating nodes are not often available. The connections may last relatively long once they take place, giving a node possibility to transfer the entire contents of its queue. More details can be found in a paper by the authors of the MIT experiment [44].

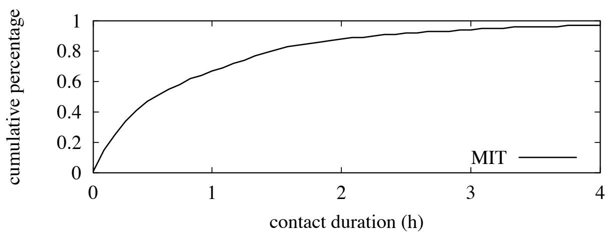

Figure 3.8: Contact durations in the MIT mobility trace.

Figure 3.8 shows the cumulative distribution function about the contact durations between the nodes in the MIT-Reality trace. A granularity of 5 minutes was used in plotting the distribution and it can be observe how the majority of the connections are relatively short-lived, but still the contacts last often in the order or hours. With these contact durations together with the used buffer sizes and contact capacities, the nodes have often possibility to transmit multiple messages or even entire contents of their buffer during a single contact. A trace with over 62,000 bidirectional connections among 100 nodes is used over the duration of 200+ days (see details in each publication).

Map-based Mobility - Helsinki City Scenario

The Opportunistic Networking Environment (ONE) simulator [82] is used in several simulations. The ONE simulator is used to provide the mobility trace for the event-based dtnsim2 simulator [42] . ONE provides a map-based movement trace, called the Helsinki City Scenario (HCS), which has nodes moving in a part of the downtown Helsinki area. With HCS, node mobility traces are generated by simulating 80 mobile users moving by foot, 40 by car, and 6 by trams according to real routes in the downtown Helsinki. Each node represents a user moving with realistic speed along the shortest paths between different points of interest (POIs) and random locations. The nodes are divided into four different groups having different POIs and different pre-determined probabilities to choose a next group-specific POI or a random place to visit.

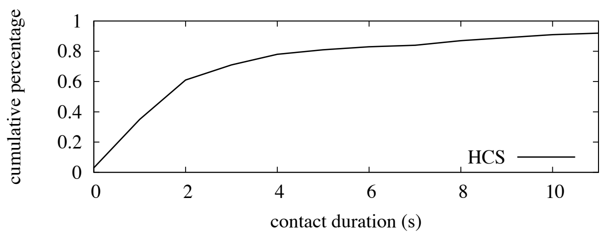

Figure 3.9: Contact durations in the HCS mobility trace.

Figure 3.9 shows the cumulative distribution function of contact durations between the nodes in the HCS trace. Large part of the contacts are very short-lived in range of few seconds. Thus, in the simulations the HCS trace presents scenario where communication is limited more by the contacts rather than the buffers in the forwarding nodes.

Caching - Challenging Artificial Scenario and Haggle Scenario

For the first experimentations with DTN caching, an artificial scenario is created that consists of 100 nodes organized in a grid topology. Each node in the topology has between two and four contacts to its neighbors. The topology is used to provide a very challenging scenario, where the contacts break during the message transfers. Although the used contact durations are shorter than likely to be seen in reality, the choice allows to observe protocol behavior with intermittent connectivity and frequent bundle transmission failures. In addition, the messages are given fairly small lifetime so that they get dropped if a relatively straight route is not available. A simple topology was chosen to avoid biases.

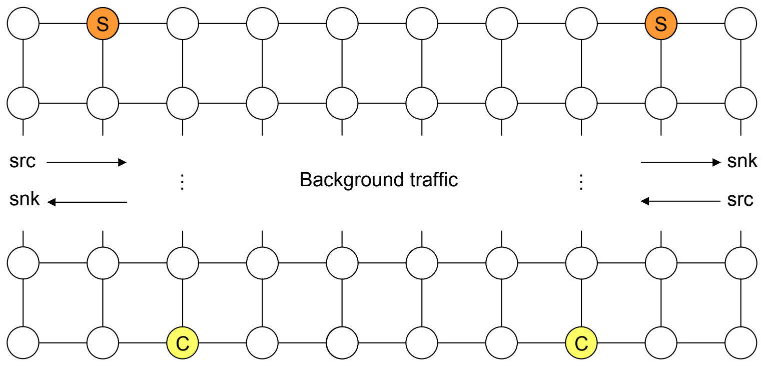

Figure 3.10: A simple grid topology used in the simulations.

Figure 3.10 illustrates the network topology used in the first simulations. The clients (C) reside on one edge of a 10x10 network and the servers (S) on the opposite side. Nodes connect to their neighbors by 802.11 WLAN links at 11 Mbit/s, and static 100 ms link delays. Each node may temporarily use 500 MB for queuing and 100 MB for caching bundles. Clients sent independently requests every 3 minutes and protocol messages time out after 20 s, unless cached. The clients choose 70% of resources from 4 URIs that are of shared interest and the rest 30% from a random pool of 400 URIs. The intermediate nodes may cache bundles for 20 minutes if there is cache space available for the bundles. The link behavior follows a pattern with the uptime and downtime are exponentially distributed with a 16 s and 4 s mean, respectively. Background traffic is added to cause frequent buffer overflows. For more details on the simulation, see Publication III.

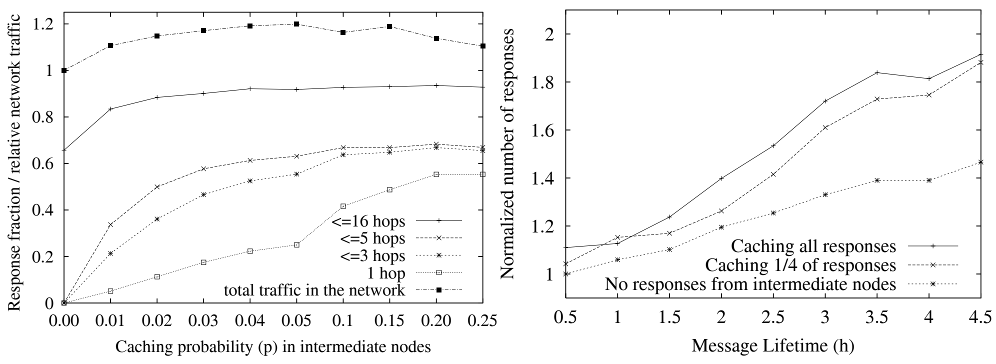

Figure 3.11: Caching in challenging (left) and in Haggle (right) scenarios.

Figure 3.11a illustrates the effect of caching bundles in nodes even after successful forwarding. The caching decision is taken independently for each message with the probability p shown on the X axis. On the Y axis, the *response fraction* shows the likelihood that client receives the response from a cache that is at most n hops away. The figure shows how increasing p strongly increases the probability that the response was sent at most n hops away, for small n. This underlines the positive impact of caching. As the responses come from the proximity of the client and the requests do not travel further, the increasing p also reduces the overall traffic—which is plotted normalized to the amount of traffic with ($p = 0.01$). Already a small value for p leads to significant performance improvements.

The effect of caching on content retrieval is also investigated by using realistic contact traces as obtained from an experiment in the Haggle project [88] using the same node implementation as above. From the Haggle trace, only those nodes are included for which at least 30 contacts are reported (a total of 122 nodes) and use only iMotes or fixed nodes for caching. Wireless bi-directional connections are assumed with a capacity of 2.1 Mbit/s and used again 500 MB forwarding and 100 MB caching buffer. Six fixed and four mobile nodes served as gateways to the Internet to access fixed servers. Random requests are sent for files of sizes 600, 1200, and 1800 KB, which seems realistic for a weblog containing photos, from two mobile and three stationary nodes.

Figure 3.11b, obtained with the Haggle trace, illustrates the impact of response caching as well as message and caching lifetime on the number of responses received. The x-axis indicates the message lifetime, and the caching lifetime is set to 15 times the message lifetime. The results are normalized with respect to the number of responses received without caching at a lifetime of 30 min (44 responses) and the responses are plotted with and without caching. The caching increases the response probability even in sparsely connected scenarios. It can be also seen that the increasing message and caching lifetimes yield better performance increase with caching. Due to the limited connectivity, no forwarding buffer overflows were observed during the simulations. Further simulations showed that most responses come from three hops or less in distance, which further emphasizes the achievable performance gain from caching. Even the contacts are longer than in the artificial scenario, the response distances are limited since large inter-contact times prevent traversing large number of contacts within the message TTL. In the caching simulations, the response time to bundle requests ranges from 1.5 to 3 hours, which appears reasonable for a travel scenario where shared contents can be obtained without infrastructure use.

The evaluations confirm that taking advantage of self-contained nature of DTN bundles and allowing intermediate nodes to use earlier responses as replies to retrieval requests can increase content access performance in the DTNs. The results further illustrate that even a small contribution from the intermediate nodes is enough to improve retrieval performance seen by the client applications. Moreover, the benefits of caching are further confirmed by observation that many responses are originated by nodes in the proximity, which helps to reduce network load and allows to receive replies even when distant storage locations cannot be reached.

Caching - More Realistic Scenarios and Content Redundancy

The first experiments confirmed that reliable content retrieval is not always possible in opportunistic communication, thus the investigation continues on how to use redundancy to increase success of content retrieval. For the following simulations, the MIT-Reality mobility trace is used. The trace contains significantly longer contacts than the previous challenging scenarios. This allows even multiple large content items to be successfully transferred during an individual contact. However, only a limited number of community members can be reached at a time because of node mobility. Moreover, the real-world social relations between the nodes affect the contact patterns leading some nodes to meet more frequently than with random mobility.

In the simulations, client and gateway functionalities are added to a subset of nodes. From those nodes having large numbers of Bluetooth contacts, four nodes (101, 102, 14, 30) are chosen to act as clients and four nodes (79, 86, 91, 88) to act as storage gateways. The clients send requests for files that have sizes picked evenly distributed from {600, 1200, 1800} KB. To emphasize the community aspect, all clients share a common core interest of six resources out of which 70% of the requests are chosen; the other 30% of are selected from 400 other possible URIs. The requests are sent out from each client, at application level, at 100 minutes intervals with small variations added to avoid synchronization. Each node contributes 512 MB of buffer space for (short-term) DTN forwarding using a FIFO queue. If cooperative caching is enabled, 256 MB out of these are reserved for longer term caching. The simulation results have shown that the queue capacity of 256 MB is never reached, that is, the observed communication performance is influenced by the user contacts rather than by buffer overflows.

The clients are capable of sending bundle requests in plain, fragmented, or coded form. When a client requests a resource in plain form it sends a single request per bundle. When requesting a fragmented bundle, a separate request is sent for each fragment. When a client requests a resource in coded form it sends separate request for each of the n fragments and starts decoding when the first m fragments have arrived. The extra fragments are discarded by the client. Requests for resources or resource fragments may be answered from the regular forwarding queue and its part "protected" for caching. Regularly queued bundles have a TTL of one hour, whereas those in the cache part are kept for 12 hours. Messages are discarded after eight hops (as simulations showed that responses generally come from close by) or when their TTL expires. Bundles are also removed from the queue after successful forwarding. If the queue, regular or cache part, is full when a new bundle arrives, the arriving bundle is dropped.

Figures 3.12 and 3.13 show several metrics that compare relative performance of the bundle transfer and caching strategies. The figures are scaled according to the following absolute metrics; number of responses (2500), number of cache hits (2500), number of forwarding buffer responses (200), traffic during simulation (450 GB), and average response time (2500 s).

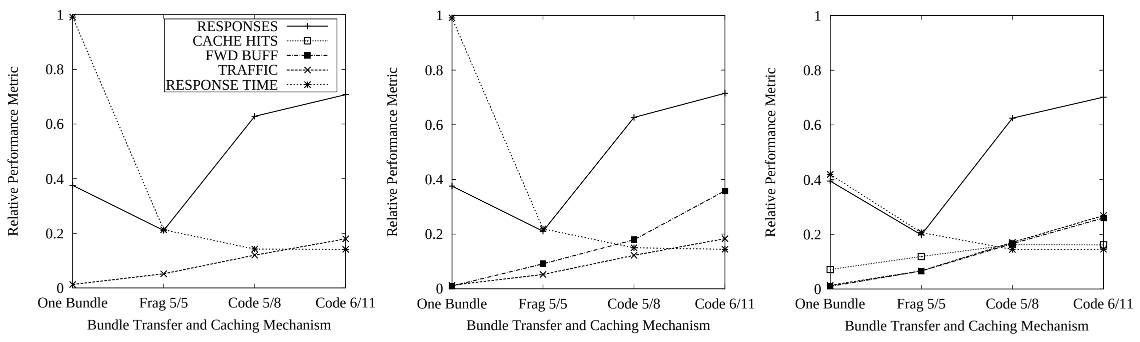

Figure 3.12: FC, responses from a) only gateways (left), b) also forwarding buffers (center), c) enhanced caching (right).

Figures 3.12 and 3.13 show the impact of redundancy and caching for two extreme routing strategies, namely *First Contact* and *flooding*, together with several different transfer strategies: non-fragmented *one bundle*, fragmented *Frag 5/5*, and coded (*Code 5/8* and *Code 6/11*). The values for m and n can be chosen freely but the observations are limited to two alternatives. *Code 5/8* has low communication and storage space requirement and can reconstruct the bundle when at maximum three fragments are missing. *Code 6/11* is significantly more redundant and is capable of reconstructing the bundle even when up to five fragments are missing. The three subfigures show performance metrics for a) end-to-end communication without caching support, b) caching support in the regular forwarding buffer (TTL=1 h, messages dropped after forwarding), and c) enhanced caching for half of the forwarding buffer (TTL=12 h, messages kept after forwarding). With c), bundles or fragments are cached at a probability of 50%. Publication IV contains further details.

The evaluation starts by using a protocol that does not create multiple copies of a message in the intermediate nodes. The *First Contact* (FC) forwarding [73] is used with a minor extension to prevent message oscillation. This forwarding scheme simply chooses the first available link and forwards the first messages in the FIFO queue. If multiple links are available simultaneously one of them is chosen randomly.

Simulation results with FC forwarding are shown in Figure 3.12. Without caching (a), the one bundle strategy manages to deliver a larger amount of responses than the simple fragmentation. This does not come as a surprise as multiple fragments need to be delivered to lead to successful delivery of the entire bundle. But fragmentation significantly reduces the response delay which suggests that contacts are often too short to allow for transmitting entire large messages. Both coding approaches increase the success in the message delivery beyond the one bundle strategy and reduce message delivery time further, which is in line with other findings (for example, [144]). As expected, the total traffic grows when smaller fragments travel further with fragmentation and increases further with erasure coding since overhead is added.

For FC forwarding, allowing responses from bundles or fragments stored in the forwarding buffer (b) has only a marginal effect on the number of responses or the response delay: both figures are largely identical. The line *FWD BUFF* indicates that less than 10% responses come from the forwarding buffer, which is not surprising for a protocol that only maintains a single copy of a message in the network. Enhancing the bundle lifetime continues by maintaining a copy in the dedicated cache part of the buffer even after forwarding (c), the number of responses increases marginally and the response time drops drastically for the one bundle case, while remaining roughly the same for fragmented and coded bundle transmission. Again, FC forwarding does not spread the messages widely, thereby limiting the likelihood for cache hits.

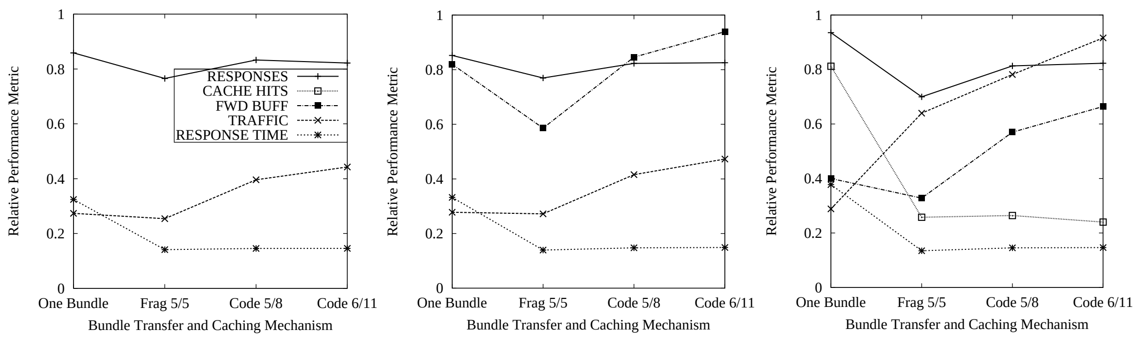

Figure 3.13: Flooding, responses from only gateways (left), also forwarding buffers (center), enhanced caching (right).

Figure 3.13 illustrates the simulation results for the same three scenarios with flooding which adds redundancy through creating message copies during forwarding process. As expected, the performance without any caching (a) is far better than for FC forwarding and the response time is much lower. Fragmentation reduces the response time further but coding does not yield additional reductions. The message delivery rate is lower with fragmentation, not reaching the one bundle case even with erasure coding—because the complete message already gets widely replicated so that retrieving it is easier than many fragments.

Allowing responses from the forwarding buffer (b) shows that some requests may be satisfied by this kind of caching but there is only a marginal overall improvement: the inter-contact times are too long for the 1 h caching period in the regular queue and bundle deletion after forwarding reduces the availability further. Using the enhanced caching (c), that is, keeping messages for 12 h, improves the response rate for the one bundle case, reduces it for fragmentation, and leaves it unchanged for erasure coding. The response time increases slightly for the one bundle case and remains unchanged otherwise; total traffic increases for (b) and (c) as expected. With (c), fewer answers are generated from the forwarding buffer as the cache maintains the bundles longer.

Simulations with realistic scenarios further confirm that the application-aware DTN routers can support content retrieval in DTNs. Adding application layer erasure coding for the messages or using flooding can improve performance. With single–copy routing, applying erasure coding can significantly improve message delivery, the delivery success was seen to increase with factor of two when compared to fragmented case and more than 50% improvement was seen when compared to non–fragmented case. In the multi–copy routing case the improvement is less pronounced, and coding increases response success only when compared to simple fragmentation.

The evaluation results show that appropriately adding the redundancy of bundles is beneficial for communication in opportunistic DTNs, but also confirm that excess redundancy does not lead to further performance gains. The benefit of using erasure coding to provide redundancy comes mainly from two factors. First, adding number of sources for content retrieval occupies only small amount of buffer and cache space compared to replication approach. Second, smaller fragments used in coding can be transferred over shorter contacts and require less overall transfer capacity. Caching, which also adds redundancy, in the intermediaries is seen beneficial when a sufficiently long content lifetime is allowed.

Caching - Retrieval and Resource Liveliness

In the following simulations, the MIT-Reality [44] mobility trace is used to investigate the effect of caching in a mobile cooperation.

More realistic behavior is added to communication model, and the clients request shared content according to Zipf distribution, which is known for web resource access. The simulations apply a wide range of buffer sizes and offered traffic in the network. In addition to simulations with the MIT-Reality trace, which are shown, the simulations are also run with HCS scenario. More accurate simulation description with additional results, also with the HCS mobility, are given in Publication V.

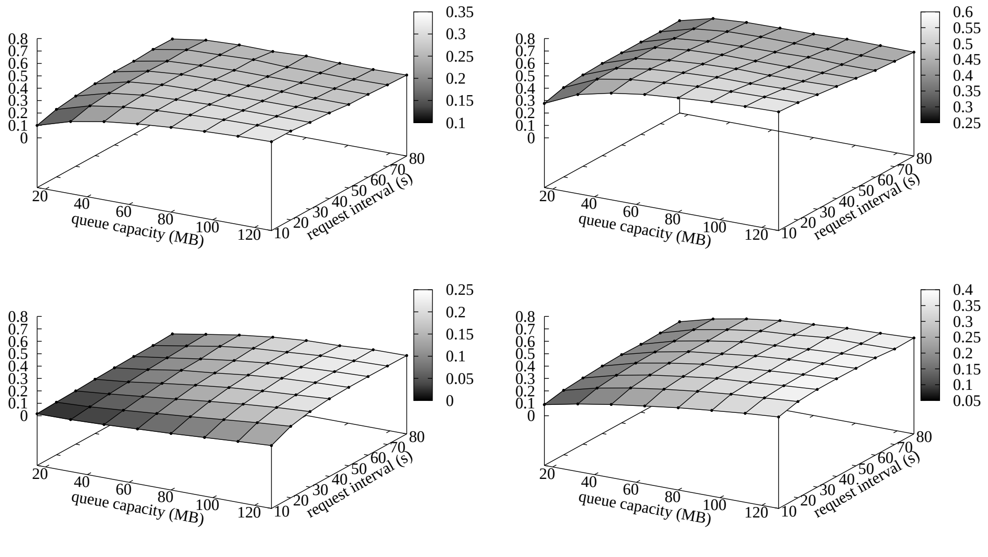

Figure 3.14: Content request performance with plain forwarding (left), with cooperative storage (right); with Spray-and-Wait (top), and Epidemic (bottom) routing.

Figure 3.14 illustrates the performance of opportunistic content storage and retrieval for the MIT scenario. The figure shows the mean probability that a client successfully receives a response to a request for two different routing protocols. For each routing protocol, a differentiation is made between plain forwarding, in which only gateway nodes can respond, and cooperative caching. Results show how the success rate changes as function of the offered load (interval between requests) and forwarding queue capacity in the mobile nodes. Messages are given 2 hour lifetime.

For both routing protocols, it can be observed that introducing opportunistic storage and retrieval improves the delivery rate across the entire parameter space roughly by at least a factor of two. Thus, with a limited message lifetime many enough caching nodes in the proximity of the requesting node can be reached to increase retrieval performance significantly. Except for Epidemic routing, performance increases slightly with the offered load.

The results also show that Spray-and-Wait performs better than the other routing strategy. Especially with small queue capacities, Spray-and-Wait maintains good performance and is still able to benefit from growing queue capacity when more resources are available. Its performance seems virtually independent of the offered load. Limiting the number of message copies in the system appears to be a good compromise that achieves resource availability while not creating too much congestion. In contrast, Epidemic routing suffers more from congestion due to its flooding nature, yet it spreads resources widely.

Interestingly, First Contact (FC) routing shows its best performance under very heavy load and large queue capacity, probably because single copies of requests and responses then easily fit into the queue/cache and congestion-induced message deletion is less an issue, so that it is more likely that a popular resource can be found in the cooperative storage. Except for this special case, First Contact routing shows poor performance with the MIT scenario. This is not surprising as only a single copy of a response is maintained and both requests and responses only take one path, making cache hits unlikely.

In the following, the focus is on Spray-and-Wait and Epidemic routing, but note that the cooperative storage approach may also improve a the performance of a single-copy routing protocol.

Figure 3.15 illustrates the retrieval delay from the client perspective. Caching reduces the delay by tens of minutes leading to roughly 30–50% decrease with Spray-and-Wait and around 20–30% decrease with Epidemic routing. An observation is that the delay is less affected by the buffer size and more by the traffic in the network.

Figure 3.16 (top) illustrates the resource liveliness (degree of replication) in the nodes. The forwarding queue size is fixed to 128 MB and a record is taken on how many nodes out of 100 carry a copy of n-th popular resource in the *cache* at the time of sending a request. The caching parameters together with Zipf-distributed content popularity cause the most popular resources to be carried by roughly one third of the nodes in the MIT

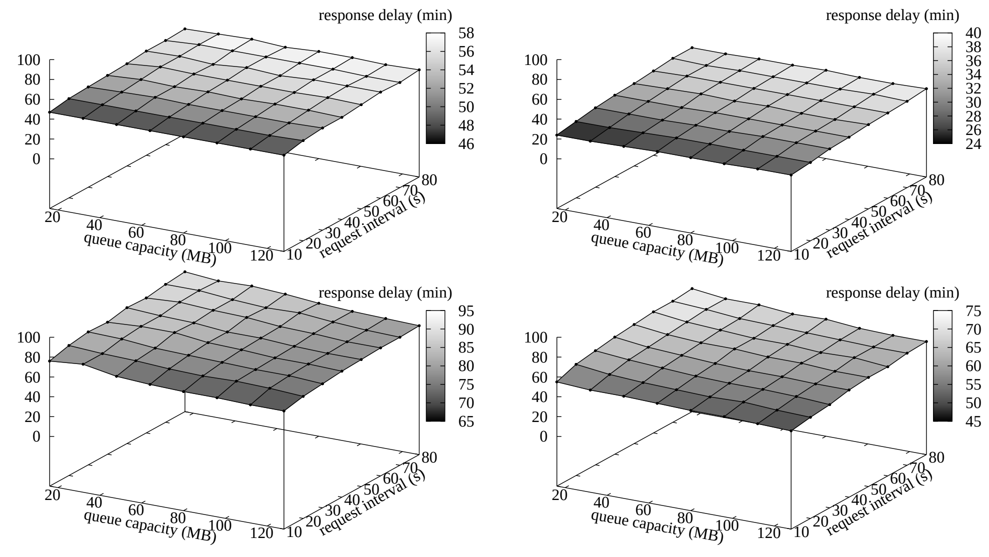

Figure 3.15: Response delay with plain forwarding (left) and cooperative storage (right); with Spray-and-Wait (top), and Epidemic (bottom) routing.

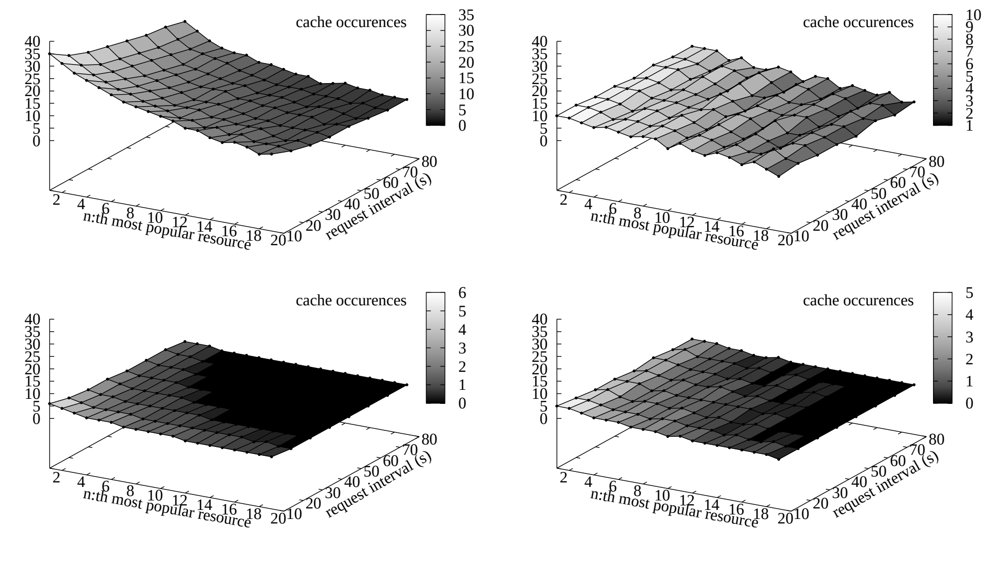

Figure 3.16: Resource liveliness in cache (top) with Spray-and-Wait in MIT (left) and, in HCS (right); and in forwarding queue (bottom).

trace. Higher request frequencies increases the liveliness of the popular resources. These observations seem reasonable for the popular resources within a community.

In contrast, in the HCS simulation, the most popular content is carried by only every tenth node and the degree of replication is more evenly distributed. This is attributed to the more frequent contacts, which support wider distribution in a short period of time. At the same time the the shorter contacts limit message exchanges. With the better mixing and spreading, the request frequency also becomes less relevant.

Similar trends can be observed for the forwarding *queue*, shown in bottom of Figure 3.16, but only a mean of five copies of the most popular resource is found. This lower number is also intuitive because the messages are deleted after their short-lived forwarding TTL (1h) expires. Moreover, in contrast to caching, occurrence of resources based on their popularity is less pronounced as queuing is not governed by the least recently used eviction policy, which allows popular resources to be maintained over long period of time. Finally, basically the same qualitative observations hold for First Contact and Epidemic routing in both scenarios for the resource liveliness in the cache as well as for Epidemic routing in the queue. Even Epidemic routing does not have more replicas in the node queues than Spray-and-Wait does. For first contact, the degree of replication in the queues is less than one in all cases.

This section evaluated content retrieval in DTNs with increasingly realistic mobility scenarios, community behavior, content popularity, and different DTN protocols. The evaluations confirm that allowing the intermediate DTN nodes to serve content retrieval requests from the forwarding buffers or caches is beneficial for users that form a community. The simulations also show that when the application-aware storage support is applied to all nodes in the network, the forwarding properties of the network are maintained and unexpected instabilities (for example, excess congestion) do not occur.

Fragmentation - Scenarios with Realistic and Random Mobility

The following provides an evaluation on the effect of fragmentation on message message delivery in DTNs. For the evaluation, the ONE simulator is used to run large set of simulations with a realistic mobility model. In addition, further results with a well known random movement based mobility model are reported in Publication VI. Two metrics are investigated: the *delivery ratio* and the *delivery latency* of the messages sent during the simulation. The messages are sent one way from the source to the destination. The delivery ratio shows the number of unique delivered messages related to the number of messages sent. In addition, fragment delivery statistics show possible biasses successful delivery of fragments, especially whether some parts of a message are delivered with larger probability than the others. Latency is measured as the time between message creation and delivery of the first copy and the latency distribution of individual fragments is shown.

Nodes connect using bi-directional links at 2 Mbit/s and have 100 MB FIFO message buffer so that the oldest messages are dropped first (except for MaxProp which has its own buffer handling scheme). Messages are sent with a time-to-live of 120 minutes. The binary version of Spray-and-Wait is used with 10 message copies. Messages delivered to the destination are handed over to the application and immediately stop occupying buffer space. Messages of the same size $[500\text{KB}–5\text{MB}]$ are sent from a random source to a random destination at intervals uniformly distributed in the range $[25, 35]$ seconds. In one series of simulations, this interval is maintained across all message sizes, thus increasing the offered load. In another series it is used only for the smallest message and increased for larger ones so that the total offered load is constant. The delivery ratio and average delay are recorded for all delivered messages. These metrics are recorded for unfragmented messages and for three different fragmentation mechanisms: proactive (using 3, 5, and 10 equally sized fragments, but the details are shown only for the 3 fragment case), reactive, and toilet paper (with 100 KB fragments).

The following explores the impact of fragmentation for six routing protocols. Two use single copy approaches, *Direct Delivery* and *First Contact* [73]; two perform variants of flooding, *Epidemic* [141] and *Prophet* [92]; *MaxProp* [16] floods but explicitly clears messages once delivered; and *Spray-and-Wait* [134] limits the number of copies created per message.

Figure 3.17 illustrates the message delivery ratios, the results are shown for only one scenario and four different routing protocols, for more details see Publication VI. The performance of routing protocols in decreasing order is: MaxProp, Spray-and-Wait, Epidemic, Prophet, Direct Delivery, and First Contact. With all the routing protocols (except for First Contact), both types of reactive fragmentation consistently show improved delivery ratio compared to the non-fragmented case, whereas proactive fragmentation constantly performs worse. The reactive toilet paper approach performs slightly better than reactive fragmentation without predefined boundaries. This can be explained by the better duplicate detection leading to better buffer utilization and reduced congestion losses. While reactive fragmentation improves the delivery, it also becomes less effective with increased message sizes. In the HCS scenario, 80% of the contact times are less than 6 s and one third less than 2 s — allowing the transfer of 1.5 MB and 500 KB, respectively. These short contacts result in that more fragments are created, but not all of them are delivered. For multi-copy routing protocols performing buffer management (Spray-and-Wait, MaxProp), the gain achieved by fragmentation increases with the message size due to less congestion, which leads to fewer fragment losses.

The fragment size distribution, see Publication VI for graphs, for both reactive fragmentation schemes reflects the contact time distribution and amplifies it since messages are not reassembled on the path. This is observable across all message sizes, and 80–90% of the fragments are smaller than 400 KB in size (equivalent to 2 s transmission time). This shows that it is advisable to perform fragmentation with short contact durations when it is allowed. The different fragmentation mechanisms impact the message delivery ratios but the latencies are fairly similar across all approaches. The latencies increase slightly

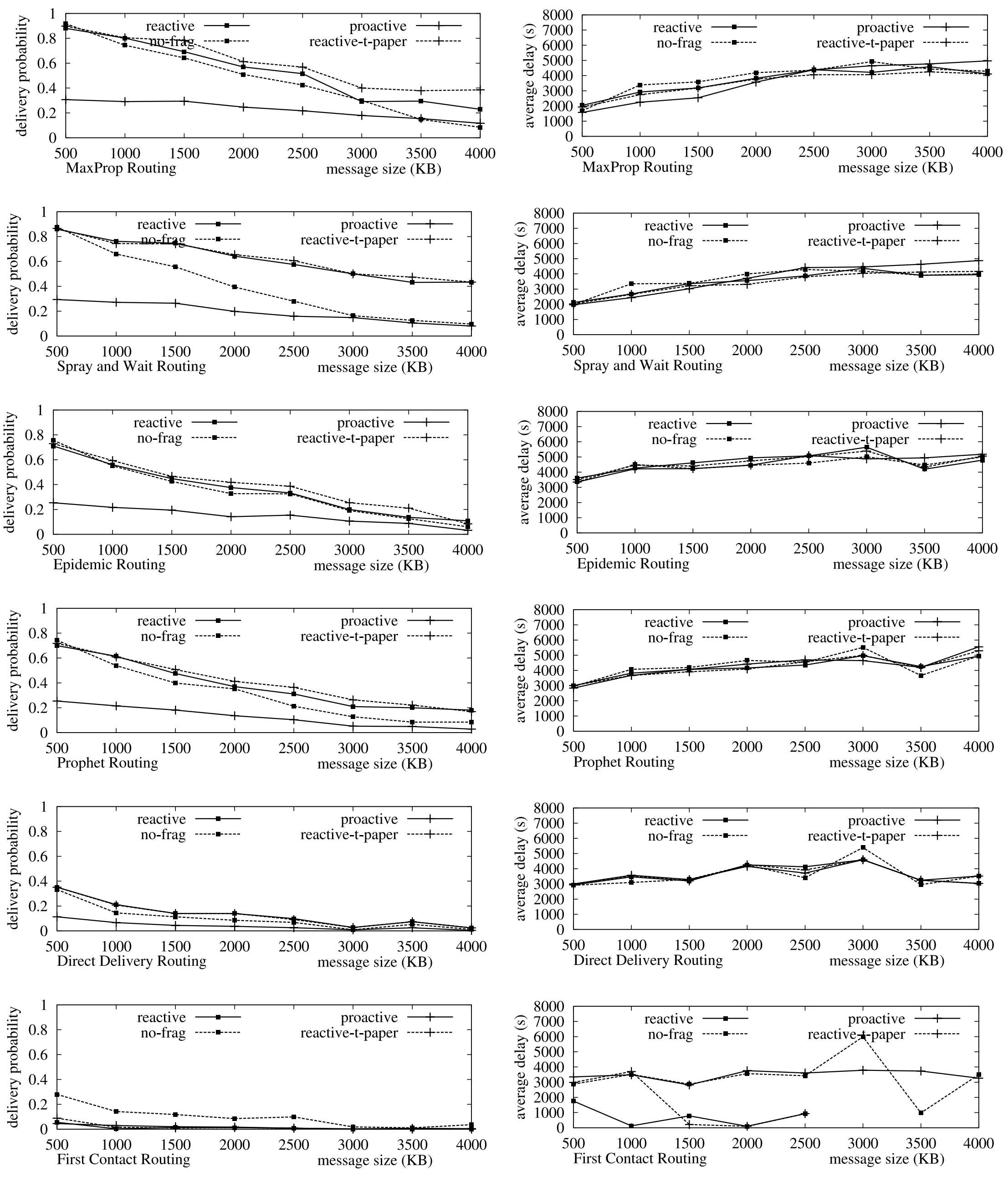

Figure 3.17: HCS mobility. Delivery probabilities (left) and latencies (right).

with the offered load for all multi-copy routing protocols. This suggests that they are determined by the overall queue lengths in the system.

Simulation results with non-congested network indicate that early fragmentation may have a negative impact. The proactive source fragmentation appears to suffer from spreading fragments across multiple paths. The reactive toilet paper approach, which prevents the creation of small fragments and hence limits path diversity, yields the best results. Generally, fragmentation expectedly improves the delivery rate of large messages.

The simulations also provide insight on which fragments of a message are received by the destination. The FIFO queuing results in the parts of a message sent first by the source to be delivered more frequently with both types of reactive fragmentation. This bias is less strong with proactive fragmentation. The overcome the bias observed with all routing protocols, random ordering of fragments is introduced in each intermediate node buffer. However, the effect was observed as long as the transmission order at the source remained the same.

After working with large self-contained messages was motivated in publications from III to V, this section discusses how to use fragmentation to allow message transfer when size of the bundle exceeds the capacity of opportunistic contacts. To summarize the observations, the evaluations show that the reactive fragmentation can increase message delivery probability and keep the delivery delay low at the same time. Limiting the reactive fragmentation to pre-defined fragmentation boundaries is further seen to increase the delivery success. For fragmentation in DTNs, it seems favorable to allow messages to proceed un-fragmented as far as possible, and fragment according to pre-defined boundaries only when needed. Moreover, maintaining bundles un-fragmented allows keeping the carried application content self-contained, which is beneficial for the cooperative content retrieval discussed in this thesis.

Web Search - Scenarios with Realistic and Random Mobility

The following evaluation uses the ONE simulator to investigate search in opportunistic mobile networks. Four different network scenarios are used, including an easy to understand static scenario and different mobility models, as explained later in this section. The nodes model devices of mobile users that are able to connect to each other by using bi-directional links at 2 Mbit/s and have unlimited FIFO message queues for DTN routing. Messages are transmitted in all or nothing fashion, that is, fragmentation is not allowed.

The four strategies, which are explained in section 3.6, are used to limit the spread of messages in the network with varying time-to-live and hop count constraints and value thresholds. While the query forwarding and search termination algorithms are designed to operate independently of the routing protocol, the routing protocols may create different numbers of messages in the network. The following evaluation compares three different routing protocols representative of different types of message replication. *First Contact* routing [73] represents the class of single-copy routing strategies, the other two use multi-copy behavior: *Spray-and-Wait* [134] limits the number of copies to a fixed maximum (10 copies) and *Epidemic* routing [141] performs flooding without any limitation on message replication.

The content query messages are sent to a DTN multicast address (*all-dtn-search-routers*) which yields anycast-style semantics in the search environment. Every search-capable DTN node joins this multicast address and is thus notified about an arriving query message. While unicast DTN messages will be deleted after delivery, in the search environment all nodes are intended recipients and messages may thus be locally evaluated by the search engine of multiple nodes, even if only a single copy is created and passed around. Besides adding local message delivery, the forwarding rules of the routing protocols are not changed; for example, First Contact only uses a single message and Spray-and-Wait will deliver the message to at most 10 nodes.

For searches limited by hop count and time-to-live, the corresponding search termination uses regular DTN routing so that the messages are forwarded until deleted by the corresponding limiting mechanism. For first response filtering and value-based termination, a special extension to DTN routing is introduced so that message forwarding may be deferred until the search application on the node signals either *forwarding* (possibly updating the query contents) or *termination* (deletes the message). Response messages are addressed to the querying node and sent using the underlying routing scheme without any changes.

The following simulations investigate searching of a single content item in a mobile DTN and then extend to searches that can result in multiple distinct items as a response. The network is observed from the point of view of a single actor. The actor frequently issues queries to the surrounding nodes, which carry a fixed and predetermined set of resources and respond by sending out contents matching the query in a reply message. The resources are distributed prior to starting simulations so that a fixed number of nodes are able to provide a response. The number of nodes holding a copy so that they could respond is varied between the simulations, and the observed metrics are plotted as a function of number of potential responses. While typical scenarios could see many nodes initiating searches simultaneously, the observations are deliberately constrained to a single requester to isolate the performance of the search distribution from effects due to congestion.

For each query, the network contains a number of distinct content items that match the query. Copies of the content items are distributed in the nodes' data stores in such a way that their popularity follows the Zipf distribution, which properly models the popularity of web content items [14] and is thus a reasonable assumption. The total number of copies of content items is fixed for each simulation run. For the first three scenarios, the number of copies of the most popular content item is half the total number of copies. For the fourth, which features more nodes, it is 20% of the total. The size of the content items is fixed at one megabyte.

Four different scenarios are used for the evaluation. The *Basic Scenario (BS)* is a static network topology with a central node around which other nodes are spread in concentric circles. The distance between the circles is 100 units, each circle having six more nodes than the next inner one. With a node transmission range of 150 units this results in node degrees between three and six for all the nodes in the network. Similar simple networks are used with each node having the same degree between four and eight. The search query is always issued at the middle node of the network. *Random Way Point (RWP)* presents a well known case for comparison. The model contains 125 actors walking in an area of 100×100 meters, comparable to an outdoor exhibition. The walking speed varies in range $0,5 - 1m/s$. The *Helsinki City Scenario (HCS)*, as described earlier, simulates 125 actors moving according to the streets in Helsinki. Finally the *Working Day Movement (WDM)* simulates a more realistic scenario whose characteristics such as inter-contact time distributions come close to those encountered in real-world traces [45]. The default scenario is chosen from section 5 in paper by Ekman et *al.* [45] in which nodes move in the Helsinki city area, but the number of nodes is reduced from 1029 to 544 by shrinking all the group sizes so that the basic contact characteristics remain.

Figure 3.18 shows the probability for retrieving a single specific content item in response to a search request issued by a node. Value based termination functions use thresholds of 0.5 and 1, referred to as $term(0.5)$ and $term(1)$. First response drop uses hopcount limits of 5 and 10, and a TTL limit is chosen to match the scenarios, BS uses 10min and all others 120min limit. As expected, the success of a search increases when the number of resources in the network increases. Both, First Contact and Spray-and-Wait routing show poor performance when there is only small number of potential responders in the network. Moreover, the small number of copies or short route of a single copy for a query does not lead to a high likelihood of reaching a node carrying a matching resource. Epidemic routing performs well with all mobility models and is less dependent on the resource density. This is an expected result when there is no contention and the query and its responses can reach large number of nodes.

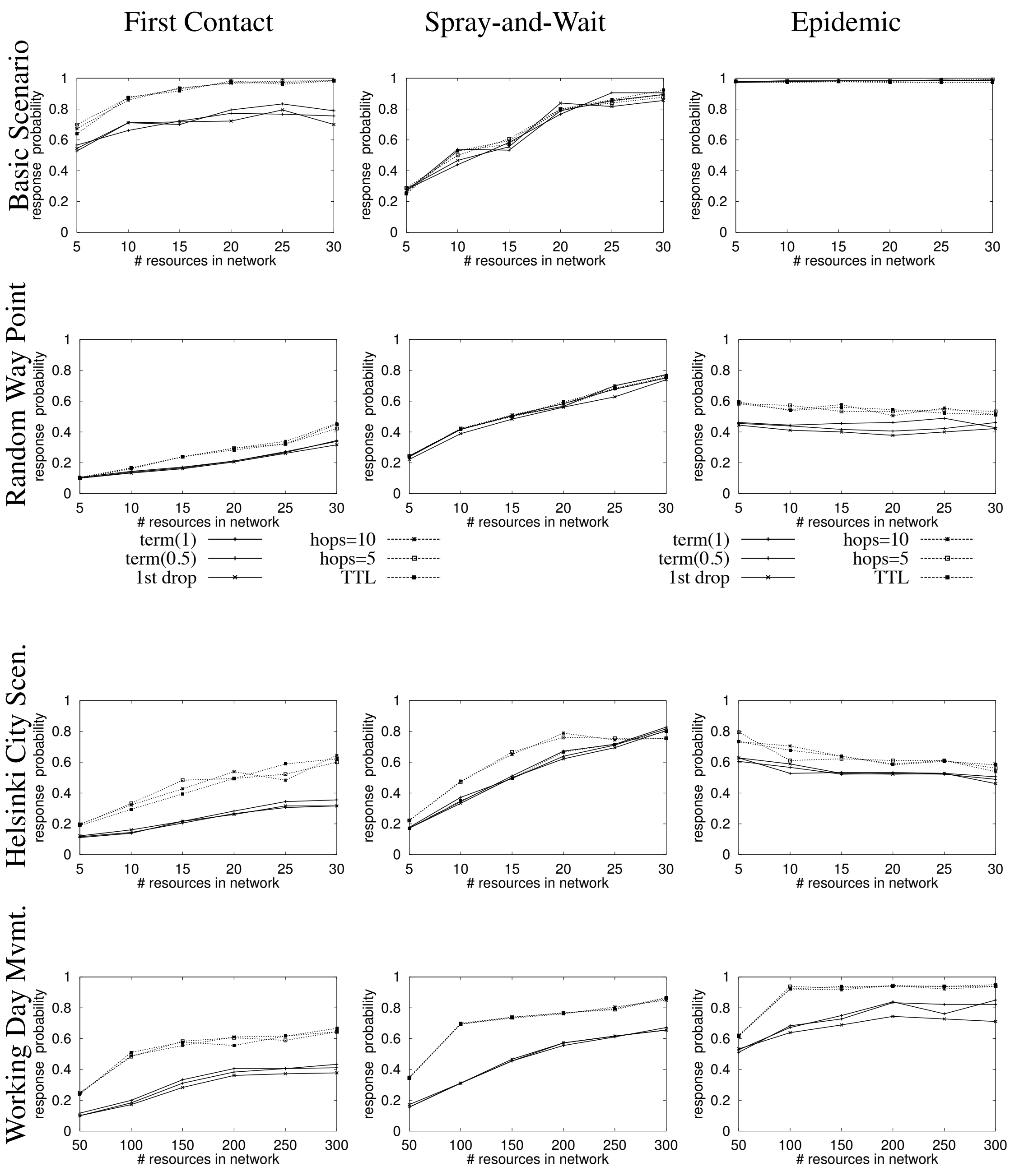

Figure 3.18: Response probability for search queries.

As explained in Section 3.6, the query forwarding and termination mechanisms can be split into two groups, termination mechanisms that are based on network level control (TTL, hopcount) and those which utilize application level information (first response drop and value-based). The former consistently yield higher retrieval success rates. With WDM mobility, the network level mechanisms lead to significant improvements whereas, with the other approaches, the improvement over local control (first response drop, marginal valued function u_{est}) is less pronounced to non-existent. This holds irrespective of the number of resources spread across the network.

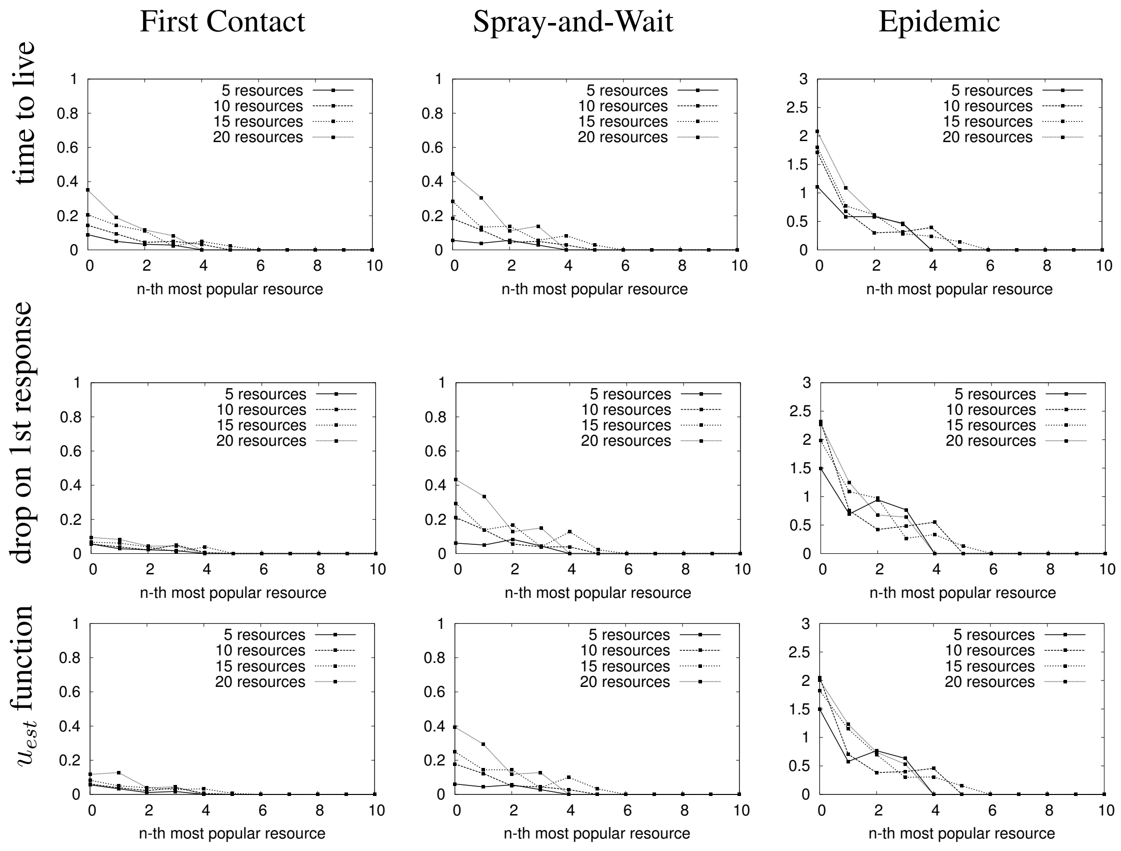

Figure 3.19: Simulation results for Helsinki City Scenario.

Figure 3.19 shows the mean number of unique replies per resource received in response to a query by a single querying node in the HCS scenario (note the threefold difference in range of the y–axis for Epidemic compared to First Contact and Spray-and-Wait routing). In this case multiple resources match a query. The resources are distributed randomly so that the number of copies per resource follows a Zipf distribution. This figure shows only one value-based termination (threshold 0.5) and only the TTL as representative for network level control: TTL and both hopcount limits perform almost identically and so do both value thresholds, which seems to indicate that expanding the search coverage by itself does not increase performance for these scenarios. The results obtained across all routing protocols reflect the occurrence frequency of the content items. However, using First Contact or Spray-and-Wait may not lead to obtaining even the most widespread content item in all cases. In contrast, Epidemic routing expectedly provides broader coverage with repeated responses for the most frequent resources. Counting also replicated responses created by the routing protocol further shows that Epidemic routing adds robustness by returning a tenfold number of responses. With Spray-and-Wait, the replicated responses lead to twofold increase in number of delivered responses. Limiting the query forwarding (and thus the resource consumption) with application layer knowledge is difficult; particularly, first response drop is not effective. TTL limiting is a workable limiting mechanisms and still allows even less popular resources to reach the querying node. An overall finding is that queries will likely be satisfied only for the more popular resources and content items from the long tail will only be captured occasionally. The rare content items are most likely responded from the vicinity of the querying node as the results for the hop count of the returned responses seem to indicate. This suggests that exhaustive searches for content items in mobile DTNs are unlikely to succeed (within a limited time frame) and should be avoided as they may be wasteful.

Figure 3.20 shows the spread of query messages in the network with Epidemic routing (other two routing protocols have a hard spreading limit stated by the protocol) to further assess the effectiveness of the limitation functions. The spread is also a metric for the incurred load on the network. It can be see how the network level mechanisms (hopcount, TTL) cause widespread forwarding of the messages, which is reflected in the better retrieval performance as observed above. In the simple scenario, the first response drop mechanism manages to limit the spread of queries especially when large number of content items are in the network, and thus close to the querying node. With the less realistic RWP mobility, the network level mechanisms limit queries as effectively as application level mechanism but still achieve good response performance. All application level ter-

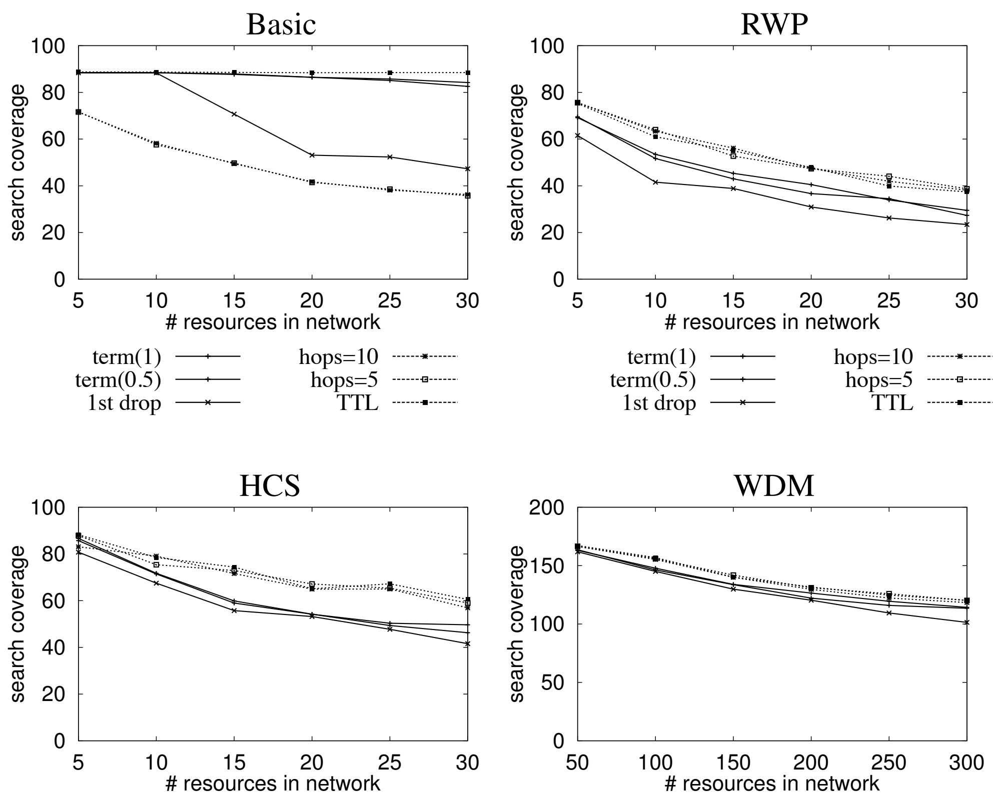

Figure 3.20: Search coverage with Epidemic routing.

mination mechanism limit the spread of searches in HCS effectively and still manage to deliver responses fairly well as can be seen in Figure 3.18 for the Helsinki City Scenario with Epidemic routing. In WDM model, all limiting mechanisms stop queries equally well but the network level mechanisms lead to a better response rate as observed above.

Figure 3.21 shows the latency from a query to arrival of the n-th unique response in the WDM scenario. In cases where the first response drop already satisfies the search, the content items are retrieved from the vicinity of the querying node. This results in low latencies for the search when First Contact routing is used. Spray-and-Wait leads to relatively similar latency profile. The latency is always highest with Epidemic routing, but the approach yields a good success ratio as discussed above.

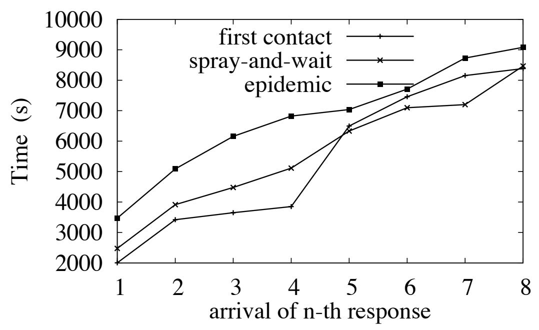

Figure 3.21: Latency of n-th arriving unique response - WDM, 300 content items.

Simulations on performing content search in opportunistic DTNs evaluated mechanisms for limiting the spread of queries in the DTN network. The results show that the content is often retrieved from the vicinity of the node, which seems favorable for the cooperative storage and caching mechanisms that were proposed earlier in this thesis. The simple network level query termination mechanisms seem effective as they limit the spread of queries and lead to good response performance. Moreover, the evaluations show that node mobility and large delays make the exhaustive searches difficult in DTNs, so that searching very rare content items does not seem feasible.

Web Access Via Hotspots - Scenarios with Realistic and Random Mobility

To evaluate web access with partial infrastructure support the ONE simulator is used with two different mobility models. First, Helsinki City Scenario (HCS) models pedestrians moving in city streets between interesting locations and more realistically matches tourist behavior on the streets. Second, random way point (RWP) with same number of nodes (140 pedestrians) and area dimensions (4.5×3.4km) is used to provide an intuitive comparison case. In both scenarios WLAN hotspots are located according to real geographic locations in Helsinki.

In the HCS scenario, the WLAN access points are located at the side of the streets, along which nodes move. For comparison, the RWP simulations report on the same area and access point locations, but with the pedestrians' movement not limited to streets. More realistic models, such as the Working Day Movement model [45], are not used since the investigation focuses on performance of users in the streets. Each mobile user in the scenario retrieves a web page at a mean interval of 5 min between requests, uniformly distributed in [280:320]s. Each request chooses a web page randomly from the list of 50 most popular websites requested from within Finland. The delay and the sender (origin server, hotspot cache, or another mobile node) of the response is measured for each request together with the number of duplicate requests and responses delivered.

Figure 3.22 illustrates a city map section showing the real access point locations in the Helsinki downtown area. The simulations use varying number of WLAN hotspots in the downtown area to observe the effect of infrastructure density. The locations of hotspots are chosen according to existing community access points (the locations are derived from `http://www.wippies.com`, for a similar service see, for example, `http://fon.com`)

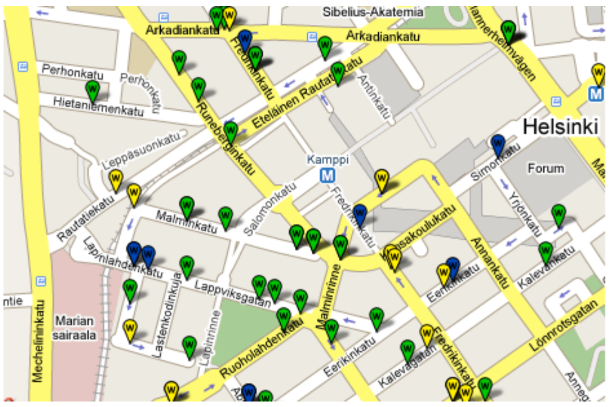

Figure 3.22: An example communication setting in downtown area.

The simulations use different random seeds and varying configurations. Each run lasts for 12 h simulation time and results in over 200k contacts between nodes. For each datapoint the mean of 1500 web access events is plotted (deviations between runs with different random seeds are negligible). The mobile nodes use bi-directional wireless links at 10 Mbit/s data rate and 50 meters communication radius to communicate with each other and with access points (APs). Each node has 512 MB message buffer capacity. The access points run with the same WLAN parameters and provide connectivity to the Internet via non-congested access link.

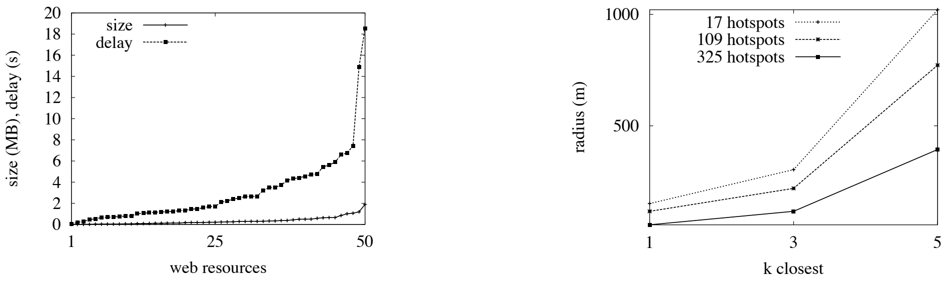

Figure 3.23: Web resources and average radius covering k APs around an AP.

Figure 3.23 (left) shows the sizes and delays for all web resources used in the simulations. The delay for accessing the web pages and their size distribution is based on measurements carried out in the lab. Scripts were used for retrieving the index pages of the top 50 web sites. For each resource, the retrieval times were recorded, and the sizes of the web pages were measured. The delay figure shows the average time to retrieve a web page based on 863 page requests. Figure 3.23 (right) illustrates the hotspot density by plotting the mean radius around a hotspot including the k-closest hotspots averaged over all hotspot locations.

While the resource retrieval and caching algorithms operate independently of the routing protocol, the routing protocols often create different numbers of messages in the network and thus affect the number of hotspots and caches reached by a request. The following evaluations use three different routing protocols as noted above: *Direct Delivery* [73], *binary Spray-and-Wait* [134] with 10 message copies, and *Epidemic* routing [141] for flooding without limit on message replication.

In the beginning of simulations all the resources reside in web servers in the fixed network. When a web server is accessed via an access point, the aforementioned measured retrieval delay is introduced in the fixed network. Responses are sent back and they also start to populate the caches of the respective WLAN access points, so that these can serve subsequent requests directly without the additional retrieval delay. The measurements are plotted as a function of message TTL, which limits how long both requests and responses can be forwarded before being discarded. Thus, the maximum response delay is twice the TTL. In addition, the simulations limit the maximum hopcount to four. Figure 3.26 below shows that this limitation is practical, since the first copy of a response message does not frequently come through a long path, which is expected result with the short TTL.

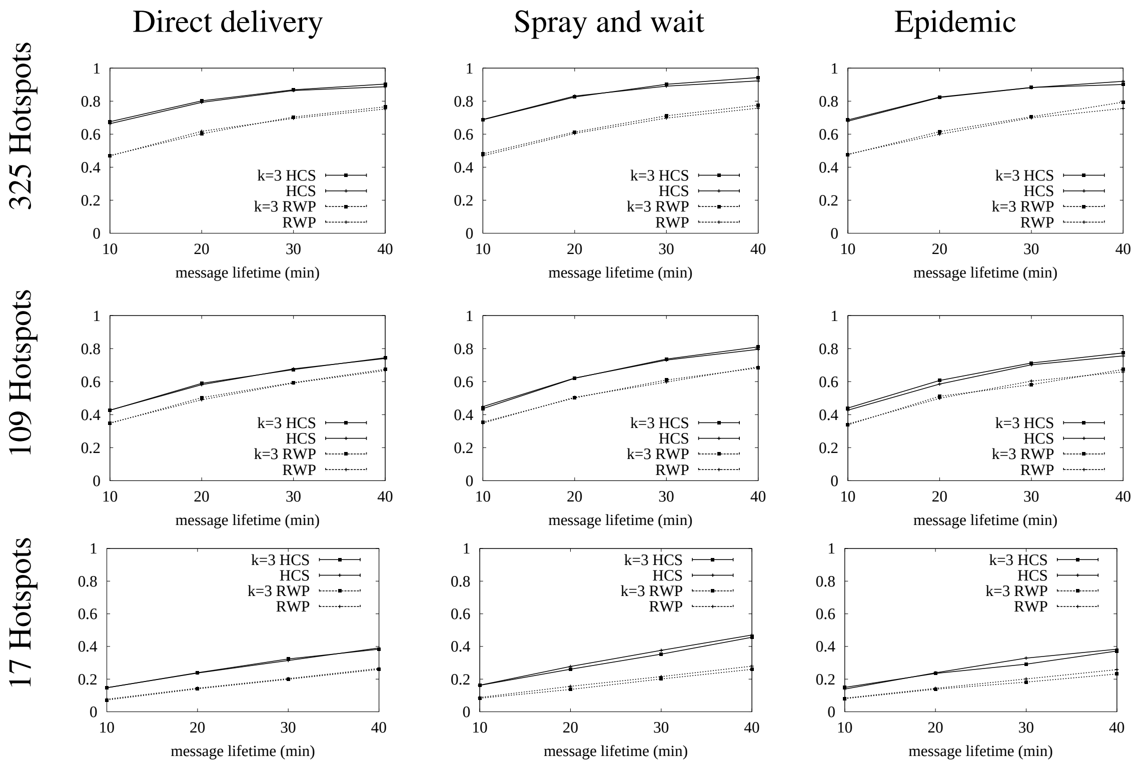

Figure 3.24: Probability that a pedestrian receives response with different message lifetimes; directly from the closest AP or via $k = 3$ closest APs.

Figure 3.24 illustrates the effect of the message TTL on the probability that a client receives a response. Results obtained by using RWP mobility are plotted in dashed lines and for HCS mobility solid lines are used. For both scenarios, $k = 3$ indicates that three close access points are used for response delivery, as explained in Section 3.7. The duplicate responses are not counted. As expected, allowing more time to deliver requests and responses significantly increases retrieval success. However, even with small TTLs half of the requests are successful. A comparison between the scenarios shows that, when pedestrians follow the city streets, instead of just randomly walking in the area, the retrieval probability increases while other variables stay the same. This is expected, since limiting node freedom increases the chances of passing an access point, leads to longer contacts, and increases contact frequency and thus forwarding opportunities. The difference between the various routing protocols is not very pronounced. Direct delivery performs almost as good as flooding, which indicates that direct interaction of mobile nodes with access points dominates performance, at least for getting *one* response.

The density of the available hotspots has a noticeable impact on the retrieval performance. Assuming the HCS model, using all 325 hotspots and leaving enough time leads to a response ratio > 0.9 for the largest and > 0.75 for the lowest TTL. Reducing the number of hotspots to about one third (109) still yields ratios of 0.5–0.8 and going down to about 5% (17) hotspots only yields a success ratio of 0.15–0.4. Even this small ratio shows some potential to offload selected traffic from using cellular network. Extending the reach of responses by applying the k-closest replication for responses does not notably increase delivery probability for nodes that are able to connect to hotspots directly, but the spreading of more message copies is highly beneficial for nodes that rely on other mobile nodes for web access (discussed later in this section).

Figure 3.25 shows the cumulative distribution function for response times with 40 min message TTL. With a large number of hotspots, nearly 30% of responses are received almost instantaneously. With a smaller number of hotspots, more time passes after request creation before the mobile node reaches a hotspot. A common observation in all scenarios is that even though responses can take as long as twice the message lifetime, most of successful retrievals lead to a first response soon after the request was issued and. Waiting

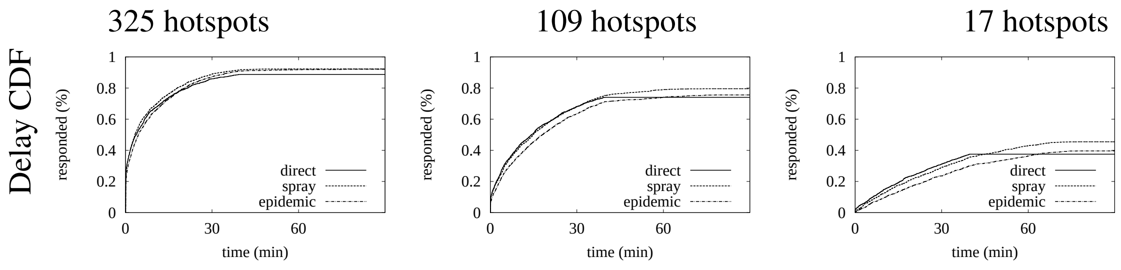

Figure 3.25: Cumulative distribution function of response delay in HCS scenario with 40 minutes message lifetime.

longer than roughly TTL does not increase much the responses. This is expected, given that most messages appear to be delivered directly to/from a hotspot. The hotspot must be reached before the request times out, but then the response will be virtually immediate, which also explains the straight line for Direct Delivery routing after TTL. Thus, the hotspot density has expectedly direct performance impact since it affects the mean time to reach an access point.

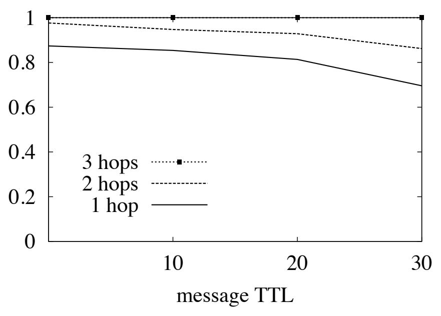

Figure 3.26: Latency of n-th arriving unique response - WDM, 300 content items.

Figure 3.26 shows the response hopcount distribution with Epidemic routing as function of message lifetime with only 17 hotspots deployed and only the closest hotspot responding. Observation is that when the message lifetime is large (30 min), only about 70% of responses come from one hop away, whereas with small message lifetimes successful responses arrive always from the closest neighbor. This shows how cooperation between the mobile nodes is important in increasing retrieval success in scenarios with sparse hotspot infrastructure.

Finally, to understand the relevance of indirect hotspot access, 20 pedestrians are configured not to be authorized to use hotspots directly For these nodes, requests and responses are relayed by other mobile nodes. These nodes can still achieve a response ratio of some 30% with Spray-and-Wait routing, TTL=40 min, and 109 hotspots. Applying the k-closest response routing increases this ratio significantly to some 50%, indicating that this controlled redundancy achieves the desired effect. A further observation is that caching in mobile clients, which is of little relevance if direct hotspot access dominates, can double the response probability compared just obtaining (cached or forwarded) responses from hotspots. Yet, caching in hotspots can lead to significant 30–70% savings in access link utilization.

Publication VIII reports further experience on implementing the proposed design for DTN based web access. Moreover, the publication describes a simple validation, which shows how the proposed design was able to retrieve web pages over an emulated test network, which uses the DTN2 reference implementation [59].

Evaluating DTN based web access via WLAN hotspots showed that even partial infrastructure support can allow mobile users to retrieve content form the Internet by using opportunistic contacts. Density of access points in a real-world urban setting is able to support retrieval in a relatively short period of time. This allows reducing the load on cellular networks, as applications that can tolerate some delay can be offload their traffic to DTN based communications. The number of available hotspots is seen as key to retrieval performance. The DTN based web access can benefit from cooperative caching, as suggested in Publications III through V, which is especially important to nodes that are not authorized to access the hotspots directly. Furthermore, caching in hotspots helps to reduce traffic between the hotspots and the web servers in the fixed network.

3.9 Summary

This chapter investigated several aspects of increasing data availability in opportunistic networks formed by the mobile users. The delay-tolerant networking architecture was used as basis to enable communication between the users. Several mechanisms, including locality, redundancy, and ability to adapt to scarce communication resources were investigated in the challenging network scenarios. The following summarizes the findings.

A design for application-aware DTN router was presented. The router provides application specific processing for messages that used for content access in DTNs. The presented router model enables using bundles that are in the forwarding buffer or cached for generating responses to content retrieval requests. The applicability of the design was investigated by a large set of simulations, which show how content retrieval in opportunistic DTNs can be made more efficient by introducing application specific storage functionality in the intermediate nodes. The simulations confirm that with the increased amount of caching the responses arrive from the closer distance. This helps both to reduce traffic in the network, and to increase the reliability of the retrieval. The simulations also confirmed that caching is beneficial when node mobility follows real world behavior recorded in the Haggle experiment.

The investigation continued on increasing data availability in opportunistic DTNs by showing how controlled content redundancy can be used to increase retrieval and communication reliability. A model for fragmented and erasure coding based bundle transfer was presented. In addition, a format for carrying application data in a message header extension was proposed. The applicability of the design was validated by a simulation that used a real world mobility trace describing behavior of a community of users. The evaluation results show how communication reliability can be increased when redundancy is used with consideration. The results also illustrate how excess amount of redundancy may lead to decreased retrieval performance if too much traffic is created. Excess amount of redundancy was seen when highly redundant erasure code was applied on top of flooding based routing, which creates multiple message copies.

After having shown how the caching and redundancy techniques make content retrieval more efficient in DTNs, the evaluation continued by looking into more realistic scenarios. Realistic characteristic to behavior of the community members was added. With both the content access pattern and node mobility being realistic, investigation focused on how different routing approaches perform with caching. Wide variety of network load and buffer capacities was used. The results further confirmed that the caching helps to reduce the retrieval latency and to increase the retrieval reliability. Good retrieval performance achieved with Spray-and-Wait routing suggests that limiting number of message copies seems a feasible approach. The simulations also showed how the resources live in caches and in forwarding queues of the community in a predictable manner, and serve as source to content requests.

Challenges of communicating by using large bundles in opportunistic DTNs with very scarce contact resources were discussed, and message fragmentation was proposed as a solution. Several fragmentation approaches were discussed and a model for fragmentations in DTNs was formulated. In addition, managing routing information with fragmented message transfers was detailed. The evaluation focused on communication scenarios where small contact volumes limit the message propagation. The results were shown for applying different fragmentation strategies together with many of the well known DTN routing protocols. The findings illustrate how fragmentation can help to increase bundle delivery ratio when applied in a correct manner. Using reactive fragmentation together with pre-defined fragmentation boundaries showed the best results in the simulations. Applying fragmentation as late as possible was seen as a good approach, and proactive fragmentation at source was not often the optimal approach. In the simulations, the first bytes or fragments of a message were seen to reach the destination with a higher probability than the bytes sent later.

Research continued to investigate search in opportunistic networks formed by mobile nodes that use DTNs for communications. The results showed that content items are often retrieved from the vicinity of the searching node, and particularly the popular content resources are likely to be found from the proximity of the searching node. Evaluations on search investigated several scenarios and showed that the node mobility and the message delivery delays make attempts for exhaustive searches unattractive in an unstructured network. This is pronounced since popular contents will be found often and rare items from the long tail only rarely.

Finally, using WLAN hotspots to enable web access in partial infrastructure support was investigated. The results showed that the density of commercial and community hotspots in a real-world urban setting is sufficient to satisfy the majority of the requests issued by pedestrians in a short period of time. The success of retrieval was seen to improve further if several adjacent hot-spots were involved in spreading the responses. While the number of hotspots available was the key to performance, cooperation between mobile nodes only yielded minor improvements for the users that were able to directly access the hotspots. Positive impact of cooperative messaging among mobile nodes was only seen clearly in scenarios with lower access point density and higher acceptable delay. Moreover, caching in mobile nodes and hotspots improved the retrieval ratio and reduced the delay further in all setups, and was especially important for the mobile nodes that were not allowed to directly access the hotspot infrastructure. These nodes also gained the greatest benefit from controlled replication of the response messages. Applying caches in hotspots significantly helped to reduce messaging between hotspots and the fixed network.

The several mechanisms presented in this chapter can be applied in parallel in DTN networks consisting of mobile nodes and some infrastructure support. The fragmentation can be, for example, used to overcome limited contact capacity during content access though with a caveat that the fragments usually do not carry complete self-contained messages. This further motivates to forward messages in the network without fragmentation as far as possible.

Different mechanisms for limiting spread of search queries were evaluated with static content distributions; however, when opportunistic caching and content storage are applied in the network the resources distribution becomes more dynamic. This motivates the search applications to proactively keep track on resource distribution in the network to allow adapting search termination according to varying popularity of the contents.

Using limited infrastructure support offered by WLAN hotspots allows web resource retrieval from the Internet. The evaluations showed that even in settings where communication with web servers is possible, the opportunistic caching can be beneficial. The benefits were particularly pronounced for delivering responses to mobile clients that are not directly allowed to access hotspots and for reducing traffic between the access points and the fixed network. Moreover, observation that applying caching in the intermediaries maintains overall forwarding properties of the network suggest that it can be applied for DTNs in parallel with other mechanisms studied in this section. Thus, it seems that the several suggested mechanisms can co-exist in the network and help to improve the overall system operation.

4 Conclusions and Future Directions

This thesis presented several methods for increasing data access and transfer performance in the presence of failures. Extensive set of evaluations support that the proposed methods can be applied to increase reliability of content retrieval in the modern network architectures and protect against unexpected failures when the communication resources are scarce

This thesis proposed to use a software implementation of the well known Reed Solomon erasure coding to secure against storage element failures in a distributed storage storage infrastructure. The presented coding scheme can be used to cope with multiple simultaneous failures and it provides a space efficient alternative for replication. The space efficiency of coding allows distributing a file to a larger number of data items, which are unique and can be used for reconstructing the file, than is possible with replication and the same amount of storage space. This can provide an appealing solution when distributed data storages are implemented with high availability requirements and limited budget. Such limitations are currently faced, for example, in scientific data preservation, where the amount of data increases but the supporting budget does not [64]. Moreover, the erasure coding approach was shown to be efficient for use in the modern Grid storage systems and with modern workstations. Additional benefit is that the presented approach allows using a large number of storage locations to download a distributed file in an efficient and fault tolerant manner from distributed data storage.

The applicability of coding for the distributed Grid storages was demonstrated by using an implementation of a novel storage application with existing Grid storage systems. Multiple clients were clustered in a local area network to offer parallel access to data resulting in performance increases. The achieved results were promising, but additional work could still be done to improve the generality of the research findings in large distributed storages. With wide area storage, additional tests with clients in large number of locations would provide more solid results. Moreover, experimentation with new LDPC-based coding mechanisms together with various algorithms for data transfer scheduling are an interesting direction for further studies. These kinds of tests are often difficult to implement, however, network testbeds such as PlanetLab [31] could be used for future work on the topic.

This thesis continued to cover even more challenging scenarios, where failures can render the communication network partitioned and it is not anymore sufficient to survive from an expected number of simultaneous failures. An opportunistic network formed by mobile nodes was used to investigate data access and transfer in these scenarios. Several mechanisms for increasing data availability were proposed, including, for example, application-aware processing, caching, content redundancy, coding, and fragmentation of messages. Moreover, searching content without content locator was investigated and methods for enabling web access with only partial infrastructure support were discussed. Performance of these mechanisms was investigated with a simulation model that uses the DTN architecture as a basis for communication. The simulation-based evaluations support that the DTN architecture can be efficiently used to enable content access and data transfer in challenging networks that follow realistic human behavior.

Fragmentation was formulated for DTNs, and its effect on transferring large bundles over mobile DTNs was studied. The results confirmed that fragmentation can increase message delivery probability, because it allows to transfer even large contents over short-lived contact durations between the nodes. The evaluations suggested that message should be allowed to proceed non-fragmented towards destination as far as possible. Moreover, the fragmentation should be limited to pre-defined boundaries to limit the number of distinct fragments and to help detecting and eliminating duplicate fragments. Combining fragmentation with content dissemination models provides a possible future research direction. The DTN architecture could, for example, send complete bundles with pre-defined fragmentation boundaries to DTN-enabled access points. They could then deliver the content to proximity by using chunks that are created according to the pre-determined fragmentation boundaries to enable content distribution as discussed in PodNet approach [89].

Searching for content in an unstructured network formed by mobile users was investigated with focus on controlling the spreading of search requests and limiting the number of responses to queries. Several routing protocols and mobility scenarios were used to investigate network and application level mechanisms for query termination. In the investigated scenarios, node mobility and the message delivery delays made the attempts for exhaustive searches unattractive. This suggests using content dissemination [133, 89] as a potential future direction or exploiting geographic proximity information in conjunction with searching in mobile DTNs. Moreover, nodes could continuously overhear messages carrying content as they forward them to obtain an estimate of the resource distribution in the community, and use the obtained information as additional input the search mechanisms.

This thesis also investigated web access via partial infrastructure support by using WLAN access points as gateways to the Internet. The proposed approach allows reducing the load on cellular networks by offloading selected web interactions of mobile users to a DTN messaging overlay and WLAN hotspots, and the feature can also be leveraged to enable access for nodes without cellular connectivity. The evaluations show that the density of commercial and community hotspots in a real-world urban setting is sufficient to satisfy the majority of the requests issued by pedestrians in a short period of time. The stationary nature of the access points adds structure to the opportunistic network and allows to use the location of hotspots as implicit geographic information to direct the response delivery to the mobile nodes. However, if limited cellular service is available to nodes, they could use it to provide control information for routing the bundles, as suggested by the ParaNets [147] approach, and use the high intermittent capacity of the opportunistic WLAN contacts for transferring the actual data content. Using a limited control channel seems possible future direction for enhancing web access for pedestrians, while GPS empowered automobile users could benefit from route prediction to deliver responses to the user's future location [90].

Many of the performance evaluations in this thesis were carried out by using simulation tools. Large number of scenarios, protocols, and system parameters enabled to capture wide range of parameters, which supports that the simulated models can work also in real world systems with varying characteristics. The observations also motivate to investigate more diverse scenarios as part of the future research to understand if the short-lived messages and the bounded spreading can ensure operation of the system in a broader scale, for example, across a metropolitan area. To add realistic behavior to the simulations, some of the models were derived from real world traces [88, 44] describing a community of mobile users, or from research on modeling the human movement in urban settings [45, 82]. To model the user behavior for the content retrieval applications, the access models were derived from earlier work on the Internet content access studies. While the simulation scenarios still contain many unknowns, the general findings seem fairly realistic, but there exists still many open directions for the future work.

This thesis focused on enabling content access when storage failures and mobility of the nodes cause challenges. The proposed mechanism can be seen to apply also to different types of opportunistic content sharing platforms beyond the DTN architecture. The mechanisms share a common goal to operate on environments where budget is scarce. This goal is in many ways similar to aiming to scarcity of electricity consumption, which can make the suggested solutions applicable, for example, to green computing platforms. Other challenges like participatory nature of content creation require future work to enable mechanisms for merging simultaneous input from the users and maintaining sufficiently consistent view on contents. These challenges are faced, for example, with blogging applications that enable sharing comments and allow content modifications.

This far, the investigations made assumption that the node connectivity and application behavior are dominating factors in defining the networking model. However, in a real deployment of mobile DTNs, several physical limitations, such as, device discovery or battery life, may have a significant effect on the operation of the network. These limitations are not yet well covered by the results of this thesis.

The simulation-based evaluations and partial implementations of the system show that the proposed concepts can be put into practice, but also further work on services that the DTNs should provide for applications is needed. New user interface designs are also needed for making the DTN based mobile applications intuitive to users. Only applications that fluently work on top of delay tolerant communications model can fully realize the potential of opportunistic communications.

Future looks bright for rich mobile applications as the mobile Internet usage is growing rapidly in the industrial countries and the pace of innovation for the handsets is remarkable. Yet, there are many places in the developing regions where ubiquitous wireless access is still a distant vision. In the former case, the mechanisms proposed in this thesis, both for the fixed network that supports the mobile users and the opportunistic mobile network formed by the co-located users, provide possible solutions to cope with the increasing usage of the networks. In the latter case, when the wireless networks are deployed in the developing world, the pace of innovation can be rapid and new communication models such as DTNs may pick adaptation since interoperability with legacy systems may be a lesser challenge.

After several years of research on the DTN architecture, the capabilities of the networking technology start to be sufficiently well understood to build applications on top of the DTNs and adapt current software to take the advantage of opportunistically available communication resources. The mobile community use cases discussed throughout this thesis rely on contacts to near by nodes to enable communication. Because of lack of deployment of opportunistic applications in the real world, it has been difficult to asses how their availability will effect the user behavior. However, new application distribution channels such as iPhone App Store, Android Market or Ovi Store, have enabled new methods to efficiently distribute applications to mobile users. Their software review process can help to solve some of the security challenges that have previously discouraged device-to-device communications. Their open development model also encourages large population of application developers to provide new types of applications further increasing the pace of innovation. Moreover, millions of users accessing the mobile application stores may finally provide sufficient population to bootstrap the opportunistic mobile communities. After all, the adaptation of the future systems will most likely be driven by the users.

References

[1] Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. 2001. Search in power-law networks. Physical Review E 64, no. 4, pages 046135+.

[2] Mikhail Afanasyev, David G. Andersen, and Alex C. Snoeren. 2008. Efficiency through eavesdropping: link-layer packet caching. In: NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, pages 105–118. USENIX Association, Berkeley, CA, USA. ISBN 111-999-5555-22-1.

[3] Bill Allcock, Joe Bester, John Bresnahan, Ann L. Chervenak, Carl Kesselman, Sam Meder, Veronika Nefedova, Darcy Quesnel, Steven Tuecke, and Ian Foster. 2001. Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. In: MSS '01: Proceedings of the Eighteenth IEEE Symposium on Mass Storage Systems and Technologies, page 13. IEEE Computer Society, Washington, DC, USA. ISBN 0-7695-9173-9.

[4] William Allcock, John Bresnahan, Rajkumar Kettimuthu, Michael Link, Catalin Dumitrescu, Ioan Raicu, and Ian Foster. 2005. The Globus Striped GridFTP Framework and Server. In: SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing, pages 1–11. IEEE Computer Society, Washington, DC, USA. ISBN 1-59593-061-2.

[5] M.S. Allen and R. Wolski. 2003. The Livny and Plank-Beck Problems: Studies in Data Movement on the Computational Grid. In: Supercomputing, 2003 ACM/IEEE Conference, pages 43 – 54.

[6] Globus Alliance. 2010. Globus Toolkit Website. Http://www.globus.org/toolkit/.

[7] Chen Avin and Carlos Brito. 2004. Efficient and robust query processing in dynamic environments using random walk techniques˙ In: IPSN '04: Proceedings of

the third international symposium on Information processing i n sensor networks, pages 277–286. ISBN 1-58113-846-6.

[8] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. 2007. DTN routing as a resource allocation problem. In: SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications, pages 373–384. ACM, New York, NY, USA. ISBN 978-1-59593-713-1.

[9] Aruna Balasubramanian, Brian Neil Levine, and Arun Venkataramani. 2008. Enhancing interactive web applications in hybrid networks. In: MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking, pages 70–80. ACM, New York, NY, USA. ISBN 978-1-60558-096-8.

[10] Aruna Balasubramanian, Yun Zhou, W. Bruce Croft, Brian Neil Levine, and Aruna Venkataramani. 2007. Web search from a bus. In: CHANTS '07: Proceedings of the second ACM workshop on Challenged networks, pages 59–66. ACM, New York, NY, USA. ISBN 978-1-59593-737-7.

[11] Aruna Balasubramanian, Yun Zhou, W. Bruce Croft, Brian Neil Levine, and Aruna Venkataramani. 2007. Web search from a bus. In: CHANTS '07: Proceedings of the second workshop on Challenged networks CHANTS, pages 59–66. ISBN 978-1-59593-737-7.

[12] A. Bassi, M. Beck, G. Fagg, T. Moore, J.S. Plank, M. Swany, and R. Wolski. 2002. The Internet Backplane Protocol: A Study in Resource Sharing. In: Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on, pages 194 – 194.

[13] B.J. Brachman and S.T. Chanson. 1988. Fragmentation in store-and-forward message transfer. Communications Magazine, IEEE 26, no. 7, pages 18 –27.

[14] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. 1999. Web Caching and Zipf-like Distributions: Evidence and Implications. In: INFOCOM, pages 126–134.

[15] John Bresnahan, Michael Link, Gaurav Khanna, Zulfikar Imani, Rajkumar Kettimuthu, and Ian Foster. 2007. Globus GridFTP: what's new in 2007. In: GridNets '07: Proceedings of the first international conference on Networks for grid applications, pages 1–5. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium. ISBN 978-963-9799-02-8.

[16] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. 2006. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In: INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pages 1 –11.

[17] John W. Byers, Michael Luby, and Michael Mitzenmacher. 1999. Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads. In: Proceedings of INFOCOM, pages 275–283. IEEE, New York, NY.

[18] John W. Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. 1998. A digital fountain approach to reliable distribution of bulk data. In: SIGCOMM '98: Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication, pages 56–67. ACM, New York, NY, USA. ISBN 1-58113-003-1.

[19] David G. Cameron, Ruben Carvajal-Schiaffino, A. Paul Millar, Caitriana Nicholson, Kurt Stockinger, and Floriano Zini. 2004. Analysis of Scheduling and Replica Optimisation Strategies for Data Grids using OptorSim. International Journal of Grid Computing 2(1), pages 57–69.

[20] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H.Weiss. 2007. Delay-Tolerant Network Architecture. In: RFC 4838, pages 1–35.

[21] Nicholas B. Chang and Mingyan Liu. 2007. Controlled flooding search in a large network. IEEE/ACM Trans. Netw. 15, no. 2, pages 436–449.

[22] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. 2001. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. Journal of Network and Computer Applications 23, pages 187–200.

[23] A. Chervenak, R. Schuler, C. Kesselman, S. Koranda, and B. Moe. 2005. Wide area data replication for scientific collaborations. In: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, page 8 pp.

[24] A.L. Chervenak, Naveen Palavalli, Shishir Bharathi, C. Kesselman, and R. Schwartzkopf. 2004. Performance and scalability of a replica location service. In: Proceedings of the 13th IEEE International Symposium on High performance Distributed Computing, pages 182 – 191.

[25] Karen Church, Barry Smyth, Paul Cotter, and Keith Bradley. 2007. Mobile information access: A study of emerging search behavior on the mobile Internet. ACM Trans. Web 1, no. 1, pages 4–4.

[26] D. D. Clark and D. L. Tennenhouse. 1990. Architectural considerations for a new generation of protocols. In: SIGCOMM '90: Proceedings of the ACM symposium on Communications architectures & protocols, pages 200–208. ACM, New York, NY, USA. ISBN 0-89791-405-8.

[27] Russell J. Clark, Evan Zasoski, Jon Olson, Mostafa Ammar, and Ellen Zegura. 2008. D-book: a mobile social networking application for delay tolerant networks. In: CHANTS '08: Proceedings of the third ACM workshop on Challenged networks, pages 113–116. ACM, New York, NY, USA. ISBN 978-1-60558-186-6.

[28] Bram Cohen. 2003. Incentives Build Robustness in BitTorrent. In: Workshop on Economics of Peer-to-Peer Systems.

[29] R.L. Collins and J.S. Plank. 2004. Downloading replicated, wide-area files - a framework and empirical evaluation. In: Proceedings of the Third IEEE International Symposium on Network Computing and Applications, pages 89 – 96.

[30] R.L. Collins and J.S. Plank. 2005. Assessing the performance of erasure codes in the wide-area. In: Proceedings of the International Conference on Dependable Systems and Networks, pages 182 – 187.

[31] PlanetLab Consortium. 2010. PlanetLab Website. http://planet-lab.org/.

[32] Marco Conti and Mohan Kumar. 2010. Opportunities in Opportunistic Computing. Computer 43, pages 42–50.

[33] F. Delmastro. 2005. From Pastry to CrossROAD: CROSS-layer ring overlay for ad hoc networks. In: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops, pages 60 – 64.

[34] Michael Demmer and Kevin Fall. 2007. DTLSR: delay tolerant routing for developing regions. In: NSDR '07: Proceedings of the 2007 workshop on Networked systems for developing regions, pages 1–6. ACM, New York, NY, USA. ISBN 978-1-59593-787-2.

[35] Michael Demmer, Kevin Fall, Teemu Koponen, and Scott Shenker. 2007. Towards a Modern Communications API. In: Proceedings of Workshop on Hot Topics in Networks (HOTNETS).

[36] Mike Demmer and Joerg Ott. 2008. Delay Tolerant Networking TCP Convergence Layer Protocol. Internet Draft draft-irtf-dtnrg-tcp-clayer-02, Work in Progress.

[37] GridBlocks Developers. 2010. GridBlocks Source Code. Http://sourceforge.net/projects/gridblocks/.

[38] JGroups Developers. 2010. JGroups - A Toolkit for Reliable Multicast Communication. Http://www.jgroups.org/.

[39] A.G. Dimakis, P.B. Godfrey, Yunnan Wu, M.J. Wainwright, and K. Ramchandran. 2010. Network Coding for Distributed Storage Systems. Information Theory, IEEE Transactions on 56, no. 9, pages 4539 –4551.

[40] Alexandros G. Dimakis, Vinod Prabhakaran, and Kannan Ramchandran. 2006. Decentralized erasure codes for distributed networked storage. IEEE/ACM Trans. Netw. 14, no. SI, pages 2809–2816.

[41] Fahad R. Dogar, Amar Phanishayee, Himabindu Pucha, Olatunji Ruwase, and David G. Andersen. 2008. Ditto: a system for opportunistic caching in multi-hop wireless networks. In: Proc. ACM MobiCom, pages 279–290. ISBN 978-1-60558-096-8.

[42] DTNSim2. 2007. Homepage of the DTNSim2 Delay-tolerant Network Simulator. http://watwire.uwaterloo.ca/DTN/sim/.

[43] Yu Du and Sandeep K. S. Gupta. 2005. COOP - A cooperative caching service in MANETs. In: Proceedings of Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS-ICNS), page 58.

[44] Nathan Eagle and Alex (Sandy) Pentland. 2006. Reality mining: sensing complex social systems. Personal Ubiquitous Comput. 10, no. 4, pages 255–268.

[45] Frans Ekman, Ari Keränen, Jouni Karvo, and Jörg Ott. 2008. Working day movement model. In: MobilityModels '08: Proceeding of the 1st ACM SIGMOBILE workshop on Mobility models, pages 33–40. ACM, New York, NY, USA. ISBN 978-1-60558-111-8.

[46] Kevin Fall. 2003. A Delay-Tolerant Network Architecture for Challenged Internets. In: Proceedings of ACM SIGCOMM 2003, pages 27–36.

[47] Kevin Fall. 2010. DTN Scope Control using Hop Limits (SCHL). Internet Draft draft-fall-dtnrg-schl-00, Work in progress.

[48] Kevin Fall, Scott Burleigh, Avri Doria, and Joerg Ott. 2009. The DTN URI Scheme. Internet Draft draft-irtf-dtnrg-dtn-uri-scheme-00, Work in progress.

[49] S. Farrell, S. Symington, H. Weiss, and P. Lovell. 2009. Delay-Tolerant Networking Security Overview. Internet Draft draft-irtf-dtnrg-sec-overview-06.txt, Work in progress.

[50] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. 1999. Hypertext Transfer Protocol – HTTP/1.1. In: RFC 2616, pages 1–176. RFC Editor, United States.

[51] Global Grid Forum. 2003. GridFTP: Protocol Extensions to FTP for the Grid. http://www.ggf.org/documents/GWD-R/GFD-R.020.pdf.

[52] Ian Foster. 2002. What is the Grid - a three point checklist. GRIDtoday 1, no. 6.

[53] Ian Foster and Carl Kesselman (editors). 1999. The grid: blueprint for a new computing infrastructure. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 1-55860-475-8.

[54] Christina Fragouli, Jean-Yves Le Boudec, and Jörg Widmer. 2006. Network coding: an instant primer. SIGCOMM Comput. Commun. Rev. 36, no. 1, pages 63–68.

[55] N. Freed and N. Borenstein. 1996. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. In: RFC 2045, pages 1–29. RFC Editor, United States.

[56] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google file system. In: SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pages 29–43. ACM, New York, NY, USA. ISBN 1-58113-757-5.

[57] C. Gkantsidis and P.R. Rodriguez. 2005. Network coding for large scale content distribution. In: INFOCOM 2005. 24th Annual Joint Conference of the IEEE

Computer and Communications Societies. Proceedings IEEE, volume 4, pages 2235 – 2245 vol. 4.

[58] J. Greifenberg and D. Kutscher. 2008. Efficient Publish/Subscribe-Based Multicast for Opportunistic Networking with Self-Organized Resource Utilization. In: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops, pages 1708 –1714.

[59] DTN Research Group. 2010. DTN Reference Implementation. http://www.dtnrg.org/wiki/Code.

[60] DTN Research Group. 2010. DTNRG Website. http://www.dtnrg.org/.

[61] M. Handley and C. Perkins. 1999. Guidelines for Writers of RTP Payload Format Specifications. In: RFC 2736, pages 1–9.

[62] T. Hara. 2001. Effective replica allocation in ad hoc networks for improving data accessibility. In: Procedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2001, volume 3, pages 1568 –1576, vol.3.

[63] L. Hellerstein, G. Gibson, R.M. Karp, R.H. Katz, and D.A. Patterson. 1994. Coding techniques for handling failures in large disk arrays. Algorithmica 12, pages 182–208.

[64] Andre Holzner, Pete Igo-Kemen, and Salvatoe Mela. 2009. First results from the PARSE.Insight project: HEP survey on data preservation, re-use and (open) access. http://arxiv.org/pdf/0906.0485v1.

[65] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot. 2005. Pocket Switched Networks and Human Mobility in Conference Environments. In: ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN), pages 244–251. New York, NY, USA. ISBN 1-59593-026-4.

[66] Pan Hui and Jon Crowcroft. 2008. BUBBLE Rap: Social-based Forwarding in Delay Tolerant Networks. In: 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pages 241–250. ACM.

[67] Pan Hui, J. Leguay, J. Crowcroft, J. Scott, T. Friedmani, and V. Conan. 2006. Osmosis in Pocket Switched Networks. In: First International Conference on Communications and Networking in China, ChinaCom '06, pages 1 –6.

[68] Tuomo Hyyryläinen, Teemu Kärkkäinen, Cheng Luo, Valdas Jaspertas, Jouni Karvo, and Jörg Ott. 2007. Opportunistic email distribution and access in challenged heterogeneous environments. In: CHANTS '07: Proceedings of the second ACM workshop on Challenged networks, pages 97–100. ACM, New York, NY, USA. ISBN 978-1-59593-737-7.

[69] I.Foster, C. Kesselman, and S. Tuecke. 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal on Supercomputer Applications 15(3), pages 200–222.

[70] Cleversafe Inc. 2010. Cleversafe Website. http://www.cleversafe.org/.

[71] Wuala Inc. 2010. Wuala Website. http://wua.la/.

[72] Sushant Jain, Michael Demmer, Rabin Patra, and Kevin Fall. 2005. Using redundancy to cope with failures in a delay tolerant network. In: SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, pages 109–120. ACM, New York, NY, USA. ISBN 1-59593-009-4.

[73] Sushant Jain, Kevin Fall, and Rabin Patra. 2004. Routing in a delay tolerant network. In: SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, pages 145–158. ACM, New York, NY, USA. ISBN 1-58113-862-8.

[74] Bernard J. Jansen, Amanda Spink, Judy Bateman, and Tefko Saracevic. 1998. Real life information retrieval: a study of user queries on the Web. SIGIR Forum 32, no. 1, pages 5–17.

[75] Bernard J. Jansen, Amanda Spink, and Jan Pedersen. 2005. A temporal comparison of AltaVista Web searching: Research Articles. J. Am. Soc. Inf. Sci. Technol. 56, no. 6, pages 559–570.

[76] Evan P. C. Jones, Lily Li, and Paul A. S. Ward. 2005. Practical routing in delay-tolerant networks. In: WDTN'05: Proceeding of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking, pages 237–243. ACM Press, New York, NY, USA. ISBN 1-59593-026-4.

[77] Abhinav Kamra, Vishal Misra, Jon Feldman, and Dan Rubenstein. 2006. Growth codes: maximizing sensor network data persistence. ACM SIGCOMM CCR 36, no. 4, pages 255–266.

[78] Maryam Kamvar and Shumeet Baluja. 2006. A large scale study of wireless search behavior: Google mobile search. In: CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems, pages 701–709. ISBN 1-59593-372-7.

[79] J. Kangasharju, K.W. Ross, and D.A. Turner. 2007. Optimizing File Availability in Peer-to-Peer Content Distribution. In: Proceedings of the 26th IEEE International Conference on Computer Communications, INFOCOM, pages 1973 –1981.

[80] Gunnar Karlsson, Vincent Lenders, and Martin May. 2006. Delay-tolerant broadcasting. In: CHANTS '06: Proceedings of the 2006 SIGCOMM workshop on Challenged networks, pages 197–204. ACM, New York, NY, USA. ISBN 1-59593-572-X.

[81] C. A. Kent and J. C. Mogul. 1988. Fragmentation considered harmful. In: SIGCOMM '87: Proceedings of the ACM workshop on Frontiers in computer com-

munications technology, pages 390–401. ACM, New York, NY, USA. ISBN 0-89791-245-4.

[82] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. 2009. The ONE simulator for DTN protocol evaluation. In: Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, pages 1–10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium. ISBN 978-963-9799-45-5.

[83] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. 2007. A data-oriented (and beyond) network architecture. SIGCOMM Comput. Commun. Rev. 37, no. 4, pages 181–192.

[84] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. 2000. OceanStore: An Architecture for Global-Scale Persistent Storage. In: Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), pages 190–201.

[85] Gregor Von Laszewski, Ian Foster, Jarek Gawor, Peter Lane, Nell Rehn, and Mike Russell. 2001. A Java Commodity Grid Kit. Concurrency and Computation: Practice and Experience 13(89), pages 643–662.

[86] W. H. O. Lau, M. Kumar, and Svetha Venkatesh. 2002. A cooperative cache architecture in support of caching multimedia objects in MANETs. In: WoWMoM '02: Proceedings of the 5th ACM international workshop on Wireless mobile multimedia, pages 56–63. New York, NY, USA. ISBN 1-58113-474-6.

[87] Jeremie Leguay, Anders Lindgren, James Scott, Timur Friedman, and Jon Crowcroft. 2006. Opportunistic content distribution in an urban setting. In: ACM SIGCOMM workshop on Challenged networks (CHANTS), pages 205–212. ISBN 1-59593-572-X.

[88] Jeremie Leguay, Anders Lindgren, James Scott, Timur Friedman, Jon Crowcroft, and Pan Hui. 2006. CRAWDAD trace set upmc/content/imote (v. 2006-11-17). Available at http://crawdad.cs.dartmouth.edu/upmc/content/imote.

[89] Vincent Lenders, Martin May, Gunnar Karlsson, and Clemens Wacha. 2008. Wireless ad hoc podcasting. SIGMOBILE Mob. Comput. Commun. Rev. 12, no. 1, pages 65–67.

[90] Ilias Leontiadis, Paolo Costa, and Cecilia Mascolo. 2010. Extending Access Point Connectivity through Opportunistic Routing in Vehicular Networks. In: 2010 Proceedings IEEE INFOCOM, pages 1–5. IEEE. ISBN 978-1-4244-5836-3.

[91] Sunho Lim, Wang-Chien Lee, Guohong Cao, and Chita R. Das. 2006. A novel caching scheme for improving Internet-based mobile ad hoc networks performance. Ad Hoc Networks 4, no. 2, pages 225 – 239.

[92] Anders Lindgren, Avri Doria, and Olov Schelen. 2004. Probabilistic routing in intermittently connected networks. In: The First International Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR), Lecture Notes in Computer Science, pages 239–254. Springer Berlin / Heidelberg.

[93] Anders Lindgren and Pan Hui. 2009. The quest for a killer app for opportunistic and delay tolerant networks: (invited paper). In: CHANTS '09: Proceedings of the 4th ACM workshop on Challenged networks, pages 59–66. ACM, New York, NY, USA. ISBN 978-1-60558-741-7.

[94] Witold Litwin, Rim Moussa, and Thomas Schwarz. 2005. LH*RS—a highly-available scalable distributed data structure. ACM Trans. Database Syst. 30, no. 3, pages 769–811.

[95] Ellert M., Gronager M., Konstantinov A., Konya B., Lindemann J., Livenson I., Nielsen J. L., Niinimaki M., Smirnova O., and Waananen A. 2007. Advanced Resource Connector middleware for lightweight computational Grids. Future Generation Computer Systems 23, pages 219–240.

[96] R.K. Madduri, C.S. Hood, and W.E. Allcock. 2002. Reliable file transfer in Grid environments. In: Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN), pages 737 – 738.

[97] Bruce Maggs. 2010. Tutorial on Content Delivery Networks.

[98] M. Mealling. 2002. Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database. In: RFC 3403, pages 1–14. RFC Editor, United States.

[99] Erik Nordström, Per Gunningberg, and Christian Rohner. 2009. A Search-based Network Architecture for Mobile Devices. Technical Report 2009-003, Uppsala University.

[100] Jörg Ott. 2006. Application Protocol Design Considerations for a Mobile Internet. In: Proceedings of the 1st ACM MobiArch Workshop, pages 75 – 80.

[101] Jörg Ott, Teemu Kärkkäinen, and Mikko Juhani Pitkänen. 2007. Application Conventions for Bundle-based Communications. Internet Draft draft-ott-dtnrg-dtn-appl-00.txt, Work in progress.

[102] Jörg Ott and Dirk Kutscher. 2006. Bundling the Web: HTTP over DTN. In: Proceedings of WNEPT 2006.

[103] Mayur R. Palankar, Adriana Iamnitchi, Matei Ripeanu, and Simson Garfinkel. 2008. Amazon S3 for science grids: a viable solution? In: DADC '08: Proceedings of the 2008 international workshop on Data-aware distributed computing, pages 55–64. ACM, New York, NY, USA. ISBN 978-1-60558-154-5.

[104] J. Palme, A. Hopmann, and N. Shelness. 1999. MIME Encapsulation of Aggregate Documents, such as HTML (MHTML). In: RFC 2557, pages 1–28.

[105] Craig Partridge. 2005. Authentication for Fragments. In: Proceedings of HotNets-IV.

[106] David A. Patterson, Garth Gibson, and Randy H. Katz. 1988. A case for redundant arrays of inexpensive disks (RAID). In: SIGMOD '88: Proceedings of the 1988 ACM SIGMOD international conference on Management of data, pages 109–116. ACM Press, New York, NY, USA. ISBN 0-89791-268-3.

[107] Lauri Peltola. 2007. DTN-based Blogging. Special Assignment, Helsinki University of Technology, Department of Co mmunications and Networking.

[108] Lauri Peltola. 2008. Enabling DTN-based Web Access: the Server Side. Master Thesis, Helsinki University of Technology, Department of Communi cations and Networking.

[109] Anna-Kaisa Pietiläinen, Earl Oliver, Jason LeBrun, George Varghese, and Christophe Diot. 2009. MobiClique: middleware for mobile social networking. In: WOSN '09: Proceedings of the 2nd ACM workshop on Online social networks, pages 49–54. ACM, New York, NY, USA. ISBN 978-1-60558-445-4.

[110] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz André Barroso. 2007. Failure trends in a large disk drive population. In: FAST '07: Proceedings of the 5th USENIX conference on File and Storage Technologies, pages 2–2. USENIX Association, Berkeley, CA, USA.

[111] J. S. Plank. 2005. Erasure Codes for Storage Applications. Tutorial Slides, presented at *FAST-2005: 4th Usenix Conference on File and Storage Technologies*, http://www.cs.utk.edu/∼plank/plank/papers/FAST-2005.html.

[112] J. S. Plank and M. G. Thomason. 2004. A Practical Analysis of Low-Density Parity-Check Erasure Codes for Wide-Area Storage Applications. In: DSN-2004: The International Conference on Dependable Systems and Networks, pages 115 – 124. IEEE.

[113] J. S. Plank and M. G. Thomason. 2007. An Exploration of Non-Asymptotic Low-Density, Parity Check Erasure Codes for Wide-Area Storage Applications. Parallel Processing Letters 17, no. 1, pages 103–123.

[114] J. S. Plank and L. Xu. 2006. Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Network Storage Applications. In: NCA-06: 5th IEEE International Symposium on Network Computing Applications, pages 173 – 180. Cambridge, MA.

[115] Michael O. Rabin. 1989. Efficient dispersal of information for security, load balancing, and fault tolerance. J. ACM 36, no. 2, pages 335–348.

[116] B. Ramsdell and S. Turner. 2010. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. In: RFC 5751, pages 1–45. RFC Editor, United States.

[117] K. Ranganathan and I. Foster. 2002. Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications. In: Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), pages 352–358. ISBN 0-7695-1686-6.

[118] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. 2001. A scalable content-addressable network. In: SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pages 161–172. ACM, New York, NY, USA. ISBN 1-58113-411-8.

[119] Luigi Rizzo. 1997. Effective erasure codes for reliable computer communication protocols. SIGCOMM Comput. Commun. Rev. 27, no. 2, pages 24–36.

[120] Antony Rowstron and Peter Druschel. 2001. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In: SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles, pages 188–201. ACM Press, New York, NY, USA. ISBN 1-58113-389-8.

[121] F. Sailhan and V. Issarny. 2002. Energy-aware Web caching for mobile terminals. In: Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops, pages 820 – 825.

[122] Françoise Sailhan and Valérie Issarny. 2003. Cooperative Caching in Ad Hoc Networks. In: MDM ’03: Proceedings of the 4th International Conference on Mobile Data Management, pages 13–28. Springer-Verlag, London, UK. ISBN 3-540-00393-2.

[123] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. 2003. RTP: A Transport Protocol for Real-Time Applications. In: RFC 3550, pages 1–104.

[124] James Scott, Pan Hui, Jon Crowcroft, and Christophe Diot. 2006. Haggle: A Networking Architecture Designed Around Mobile Users. In: Proceedings of IFIP WONS, pages 86–78. Les Menuires, France.

[125] Keith Scott and Scott Burleigh. 2007. Bundle Protocol Specification. In: RFC 5050, pages 1–49.

[126] Matthew Seligman. 2006. Storage Usage of Custody Transfer in Delay Tolerant Networks with Intermittent Connectivity. In: ICWN, pages 386–392.

[127] Matthew Seligman, Kevin Fall, and Padma Mundur. 2006. Alternative custodians for congestion control in delay tolerant networks. In: CHANTS ’06: Proceedings of the 2006 SIGCOMM workshop on Challenged networks, pages 229–236. ACM, New York, NY, USA. ISBN 1-59593-572-X.

[128] Sergio D. Servetto and Guillermo Barrenechea. 2002. Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks. In: WSNA ’02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications pages 12–21. ISBN 1-58113-589-0.

[129] Aaditeshwar Seth, Patrick Darragh, Suihong Liang, Yunfeng Lin, and Srinivasan Keshav. 2005. An Architecture for Tetherless Communication. In: Marcus Brunner, Lars Eggert, Kevin Fall, Jörg Ott, and Lars Wolf (editors), Disruption Tolerant Networking, number 05142 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany.

[130] Huaping Shen, Sajal K. Das, Mohan Kumar, and Zhijun Wang. 2004. Cooperative Caching with Optimal Radius in Hybrid Wireless Networks. In: NETWORKING 2004. Lecture Notes in Computer Science, volume 3042/2004, pages 841–853.

[131] Amin Shokrollahi. 2003. Raptor Codes. Technical report, Laboratoire d'algorithmique, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. Available from: http://algo.epfl.ch/.

[132] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. 1999. Analysis of a very large web search engine query log. SIGIR Forum 33, no. 1, pages 6–12.

[133] Giuseppe Sollazzo, Mirco Musolesi, and Cecilia Mascolo. 2007. TACO-DTN: a time-aware content-based dissemination system for delay tolerant networks. In: MobiOpp '07: Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking, pages 83–90. ISBN 978-1-59593-688-2.

[134] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. 2005. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In: WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, pages 252–259. ACM, New York, NY, USA. ISBN 1-59593-026-4.

[135] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. 2008. Efficient routing in intermittently connected mobile networks: the Multiple-copy case. IEEE/ACM Trans. Netw. 16, no. 1, pages 77–90.

[136] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. 2008. Efficient routing in intermittently connected mobile networks: the single-copy case. IEEE/ACM Trans. Netw. 16, no. 1, pages 63–76.

[137] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM '01: Proceedings of the 2001 conference on Applications,

technologies, architectures, and protocols for computer communications, pages 149–160. ACM, New York, NY, USA. ISBN 1-58113-411-8.

[138] Susan Symington. 2010. Delay-Tolerant Networking Metadata Extension Block. Internet Draft draft-irtf-dtnrg-bundle-metadata-block-09, Work in progress.

[139] Wesley W. Terpstra, Jussi Kangasharju, Christof Leng, and Alejandro P. Buchmann. 2007. Bubblestorm: resilient, probabilistic, and exhaustive peer-to-peer search. In: SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protoc ols for computer communications, pages 49–60. ISBN 978-1-59593-713-1.

[140] Yi-Wei Ting and Yeim-Kuan Chang. 2007. A Novel Cooperative Caching Scheme for Wireless Ad Hoc Networks: GroupCaching. In: International Conference on Networking, Architecture, and Storage (NAS) 2007, pages 62–68.

[141] A. Vahdat and D. Becker. 2000. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University.

[142] Narseo Vallina-Rodriguez, Pan Hui, and Jon Crowcroft. 2009. Has Anyone Seen My Goose? Social Network Services in Developing Regions. In: CSE '09: Proceedings of the 2009 International Conference on Computational Science and Engineering, pages 1048–1053. IEEE Computer Society, Washington, DC, USA. ISBN 978-0-7695-3823-5.

[143] Sudharshan Vazhkudai. 2004. Distributed Downloads of Bulk, Replicated Grid Data. Journal of Grid Computing 2, pages 31–42.

[144] Yong Wang, Sushant Jain, Margaret Martonosi, and Kevin Fall. 2005. Erasure-coding based routing for opportunistic networks. In: WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, pages 229–236. ACM, New York, NY, USA. ISBN 1-59593-026-4.

[145] Forrest Warthman. 2003. Delay-Tolerant Networks (DTNs). A Tutorial.

[146] Jörg Widmer and Jean-Yves Le Boudec. 2005. Network coding for efficient communication in extreme networks. In: WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, pages 284–291. ACM, New York, NY, USA. ISBN 1-59593-026-4.

[147] Mike P. Wittie, Khaled A. Harras, Kevin C. Almeroth, and Elizabeth M. Belding. 2009. On the implications of routing metric staleness in delay tolerant networks. Comput. Commun. 32, no. 16, pages 1699–1709.

[148] Bo Yang and Ali R. Hurson. 2005. Content-aware search of multimedia data in ad hoc networks. In: MSWiM '05: Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wi reless and mobile systems, pages 103–110. ISBN 1-59593-188-0.

[149] L. Yin and G. Cao. 2004. Supporting cooperative caching in ad hoc networks. In: Proceedings of IEEE INFOCOM.

[150] Liangzhong Yin and Guohong Cao. 2006. Supporting Cooperative Caching in Ad Hoc Networks. IEEE Transactions on Mobile Computing 5, no. 1, pages 77–89.

[151] Z. Zhang. 2006. Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks: Overview and Challenges. IEEE Communications Surveys and Tutorials 8, no. 4, pages 24–37.

[152] Wenrui Zhao, Yang Chen, Mostafa Ammar, Mark Corner, Brian Levine, and Ellen Zegura. 2006. Capacity Enhancement using Throwboxes in DTNs. In: Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS), pages 31 –40.

ISBN 978-952-60-3507-9
ISBN 978-952-60-3508-6 (PDF)
ISSN 1795-2239
ISSN 1795-4584 (PDF)