UPPSALA
UNIVERSITET

IT Licentiate theses
2012-002

Measurements in Opportunistic Networks

FREDRIK BJUREFORS

UPPSALA UNIVERSITY
Department of Information Technology

Measurements in Opportunistic Networks

Fredrik Bjurefors
Fredrik.Bjurefors@it.uu.se

Mars 2012

Division of Computer Systems
Department of Information Technology
Uppsala University
Box 337
SE-751 05 Uppsala
Sweden

http://www.it.uu.se/

Dissertation for the degree of Licentiate of Philosophy in Computer Science

ISSN 1404-5117
Printed by the Department of Information Technology, Uppsala University, Sweden

Abstract

Opportunistic networks are a subset of delay tolerant networks where the contacts are unscheduled. Such networks can be formed ad hoc by wireless devices, such as mobile phones and laptops. In this work we use a data-centric architecture for opportunistic networks to evaluate data dissemination overhead, congestion in nodes' buffer, and the impact of transfer ordering. Dissemination brings an overhead since data is replicated to be spread in the network and overhead leads to congestion, i.e., overloaded buffers.

We develop and implement an emulation testbed to experimentally evaluate properties of opportunistic networks. We evaluate the repeatability of experiments in the emulated testbed that is based on virtual computers. We show that the timing variations are on the order of milliseconds.

The testbed was used to investigate overhead in data dissemination, congestion avoidance, and transfer ordering in opportunistic networks. We show that the overhead can be reduced by informing other nodes in the network about what data a node is carrying. Congestion avoidance was evaluated in terms of buffer management, since that is the available tool in an opportunistic network, to handle congestion. It was shown that replication information of data objects in the buffer yields the best results. We show that in a data-centric architecture were each data item is valued differently, transfer ordering is important to achieve delivery of the most valued data.

Acknowledgement

I would like to thank my supervisor Per Gunningberg for his input on my work and giving me the opportunity to work in the field of communication research. My co-supervisors I have had over time, Kaustubh Phanse for introducing me to opportunistic networks, Lars-Åke Larzon for the helpful advice about teaching and handling students, and finally Christian Rohner in the work in Haggle project.

The additional members of the Haggle Team, Daniel Aldman, Erik Nordström, and Oscar Wibling.

My fellow PhD student, Olof Rensfelt, Frederik Hermans, Laura Feeney, Volkan Cambazoglu, and Hjalmar Wennerström. I would especially like to thank my office mate until two years ago, Ioana Rodhe for all the discussions about everything and nothing.

I would also like to thank Sam Tavakoli for his implementation work on the last paper and the long evenings spent debugging code, and Edith Ngai for your insightful queries about our results.

Last but not least, Liam McNamara for the input on my thesis and the collaboration on the last paper.

Included papers

Paper A: **"A Testbed for Evaluating Delay Tolerant Network Protocol Implementations"**
Fredrik Bjurefors and Oscar Wibling
In Proceedings of the 5th Swedish National Computer Networking Workshop, Karlskrona, May 2008

Paper B: **"Haggle Testbed: a Testbed for Opportunistic Networks"**
Fredrik Bjurefors, Per Gunningberg, and Christina Rohner
In Proceedings of the 7th Swedish National Computer Networking Workshop, Linköping, June 2011

Paper C: **"Interest Dissemination in a Searchable Data-Centric Opportunistic Network"**
Fredrik Bjurefors, Erik Nordström, Christian Rohner, and Per Gunningberg
Invited paper, European Wireless 2010, Lucca, Italy, April 2010

Paper D: **"Congestion Avoidance in a Data-Centric Opportunistic Network"**
Fredrik Bjurefors, Per Gunningberg, Christian Rohner, and Sam Tavakoli
In Proceedings of ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2011), Toronto, Canada, August 2011

Paper E: **"Making the Most of Your Contacts: Transfer Ordering in Data-Centric Opportunistic Networks"**
Christian Rohner, Fredrik Bjurefors, Per Gunningberg, Liam McNamara, and Erik Nordström
In Proceedings of the Third International Workshop on Mobile Opportunistic Networking (MobiOpp 2012), Zurich, Switzerland, March 2012

Comments on my participation

Paper A: I designed and implemented the testbed.

Paper B: I designed and implemented the second generation of the testbed, as well as performed the experiments and analyzed the repeatability of experiments.

Paper C: I performed all experiments. The data was analyzed in collaboration with Christian Rohner.

Paper D: Congestion avoidance was implemented in Haggle as a Master thesis which I supervised. I performed the experiments, the analysis was done in collaboration with Christian Rohner.

Paper E: I performed all experiments. The data was analyzed in collaboration with Christian Rohner.

List of work not included in this thesis

A: **"Performance of Pastry in a Heterogeneous System"**
Fredrik Bjurefors, Richard Gold, and Lars-Åke Larzon
In Proceedings of the Fourth IEEE International Conference on Peer-to-Peer Computing, Zurich, Switzerland, August 2004

B: **"Testbed and methodology for experimental evaluation of opportunistic networks"**
Fredrik Bjurefors, Oscar Wibling, Christian Rohner, and Kaustubh Phanse
In Proceedings of the 7th Scandinavian Workshop on Wireless Ad-Hoc Networks, Johannesberg, May 2007

C: **"Search-Based Picture Sharing With Mobile Phones"**
Erik Nordström, Daniel Aldman, Fredrik Bjurefors, and Christian Rohner
Demonstration, Swedish National Computer Networking Workshop, Uppsala, May 2009

D: **"Using search to enhance picture sharing with mobile phones"**
Erik Nordström, Daniel Aldman, Fredrik Bjurefors, and Christian Rohner
Demonstration (MobiHoc 2009)

E: **"Haggle - A Search-Based Data Dissemination Architecture for Mobile Devices"**
Erik Nordström, Daniel Aldman, Fredrik Bjurefors, Christian Rohner
Demonstration (MobiSys 2009)

Contents

Chapter 1

Introduction

1.1 Wireless Mobile Networks

Wireless data communication is ubiquitous today. It is used in homes to connect computers to the Internet through a wireless router. At many places it is also possible to connect wirelessly to the Internet at cafes and restaurants. But, this still requires a fixed point where the network can be accessed and is based on fixed infrastructure. Preferably, we would like to have Internet access wherever we are. To achieve this, several challenges have to be considered depending on the type of communication and the scenario. In short range communication, based on infrastructure, nodes communicate through a wireless access point. When many nodes access the same wireless access point they all compete for the same broadcast medium. Therefore, nodes have to take precautions to avoid collisions. If the network has a connection to Internet, nodes are assigned Internet protocol (IP) addresses during the time they are connected to the network. Through the access point, nodes can connect to other networks and often the Internet.

However, if a node moves between wireless networks that are maintained by different network providers, the node has to be assigned a new address. Hence, the node can no longer be reached using the old address. Also, it cannot maintain a connection with streaming data, e.g., telephone call, video, or music. Mobile IP [31] addresses this problem by using indirection points. An indirection point is a fixed relay point in the network that keeps track of the mobile node and forwards traffic to the mobile node. A node has a globally unique address that binds to the address where the nodes is currently connected, i.e., a server maps a globally unique IP address to the current IP address the node holds. With this solution the node can be reached using the same address no matter which network it is connected to at the moment and which address it has been assigned by the local network providers.

The solutions mentioned so far all require infrastructure. To enable com-

munication where infrastructure does not exist, nodes have to cooperate and route/forward traffic on behalf of other nodes in the network. Ad hoc routing is one solution for this problem. Adding ad hoc routing on top enables the possibility to route traffic through other nodes in the network, creating a mobile ad hoc network (MANET). The MANET protocols provide support for both mobility and route discovery. When nodes move in the network, route discovery creates new paths to maintain communication between end-points. MANETs can be connected to a gateway that has access to the Internet, enabling the nodes in the MANET to communicate with nodes in the Internet. An end-to-end path is created from the source to the destination, this enables the nodes to use a TCP connection. If links are not bi-directional or there isn't an alternative path back to the source TCP breaks, since reliable communication (TCP) requires a feedback loop, i.e., acknowledgements.

To handle broken paths and long delays, data can be stored in intermediate nodes and forwarded when an opportunity occurs, this communication principal is called store-carry-and-forward. With store-carry-and-forward of data, nodes can communicate even if the network is partitioned and connections to other nodes are intermittent. This communication paradigm can bring networking to remote areas, like the N4C project [4] that carry data in the northern part of Sweden by using helicopters as relays nodes in the network. KioskNet [14] and DakNet [30] are projects in rural India where, device-to-device communication is used to bring information to parts of the country that is otherwise disconnected from the rest of the world. Store-carry-and-forward strategies are used in devices on, e.g., buses and motorcycles, to access web pages, send and receive e-mails. The physical movement of the vehicles disseminate information in the network when they reach the next village.

The focus of this thesis are measurements and a platform to experimentally evaluate the performance of a new network architecture, called Haggle [29, 34]. Haggle is a data-centric, that means nodes address the data they are interested in and not where the data comes from. Haggle is also developed to work in a scenario where connections are intermittent and unscheduled, e.g., phones, that are carried by people, communicate when two persons happens to meet each other, creating an *opportunistic network*. Data is disseminated in the network when other devices come in radio range for communication. Using our testbed we evaluate data dissemination overhead, congestion avoidance, and the importance of transfer ordering in data-centric opportunistic networks. Using Haggle, we perform the following work:

- To evaluate data-centric opportunistic networks we use emulated experiments. Our testbed, described in Paper A and B, was developed to achieve repeatable experiments based on artificial traces or contact

traces from real-world experiments.

- Data dissemination in an opportunistic network requires overhead in terms of control traffic to negotiate how data should be disseminated. The control traffic can be used to reduce the side effect of data replication in the network. In Paper C, we demonstrate that informing other nodes about their interests in specific data, the redundant data traffic in opportunistic networks can be reduced, even when including the control traffic needed for dissemination of interests.

- Congestion occurs in an opportunistic network when several nodes relay traffic through the same intermediate node. The storage capacity of the node may then be exhausted, causing long queuing delays or that nodes have to drop data. In Paper D, we investigate different data dropping strategies and demonstrate that by removing data that has been replicated many times, nodes with small relay buffers will still achieve a reasonable delivery ratio.

- The order in which data is sent during a contact in an opportunistic network is of importance if data have different relevance for the receiver. By considering the relevance of data, nodes receive higher relevant data first. In Paper E, we demonstrate that local satisfaction disseminates the most relevant data first in the short run but is outperformed by global satisfaction over time.

1.1.1 Contributions

The main contributions of this thesis are:

- Creating a testbed for evaluating opportunistic networks, with focus on evaluating the Haggle architecture.

- Showing the importance of knowing what other nodes have in their buffers in data-centric opportunistic networks.

- Evaluation of node congestion in data-centric opportunistic networks, and a set of data dropping strategies.

Chapter 2

Background and Related Work

In this chapter, the related work in the area is covered, that is, content-centric opportunistic networking. It is a branch of delay tolerant networking that addresses data instead of nodes. In the last section of this chapter the Haggle architecture is briefly described. Haggle is used for all evaluations in this thesis. We begin with describing delay tolerant networks.

2.1 Delay Tolerant Networks

Delay Tolerant Networks (DTN) [1]—also referred to as disruptive tolerant networks or ad hoc networks with intermittent connectivity—are sparse wireless mobile networks with intermittent connectivity where, most of the time, a complete path from the source to the destination never exist. The first initiative introduced to be able to cope with the violations of TCP/IP requirements was done by the inter-planetary network special interest group (IPNSIG) [3]. This type of communication has long delays and is disrupted, but has predictable communication patterns. DTN is a generalization of the inter-planetary communication and does not assume predictable communication. DTN differs in characteristics to traditional networks when it comes to: continuous connectivity, very low data loss rate, and reasonably low propagation delay, that is found in the Internet. A DTN does not require an end-to-end path between communicating nodes to reliably delivery data.

DTNs operates in three stages: neighbor discovery, data transfer, and storage management [8]. Neighbor discovery must be implemented since node contacts are, in most cases, unpredictable. A node must scan or listen for adjacent nodes to be able to find new nodes before communication can take place. Data transfer must consider loss of connectivity and may be based on the prediction of delivery probability. Priority decisions on what to send has to be made to utilize a contact efficiently. Connection loss

also requires the possibility to store data if the next hop is temporarily unavailable. The finite storage space of a node must be controlled to achieve the highest performance. Nodes are often disconnected from large portions of the network, therefore, the control can only be based on local information since a global information is not available to a DTN node.

Three types of scenarios with intermittent contact are considered in DTNs; scheduled, predicted, and opportunistic:

- *Scheduled contacts* are established at a certain point in time and the duration of contact is known in advance. Forwarding can therefore be scheduled based on nodes that are going to be encountered. After a successful transfer, sent data can be removed from the buffer. DTNs built on bus schedules [8, 14, 30] and inter-planetary communication have more or less strict scheduled contacts.

- *Predicted contacts* are not based on a schedule but rather on prediction based on history of previous contact. Predictions can also be based on movement information, e.g., geographical location. Routes can be chosen based on contact predictions [26, 9] .

- *Opportunistic contacts* are unscheduled and unpredictable. Any available contact is used for communication when it presents itself. Such contacts are used in "pocket switched networks" [34], where people are carrying mobile wireless devices that come in communication range of each other. Another example is a helicopter or hiker passing a rural area [4] and picking up data, acting as a data mule.

We focus on opportunistic contacts in this thesis.

2.1.1 Opportunistic Networks

Opportunistic networks is a subclass of DTN where contacts appear opportunistically without prediction or schedule. For example, the movement of zebras in ZebraNet [21] cannot be controlled, unlike the movement of scheduled buses and motorcycles in KioskNet [14] and DakNet [30]. Therefore, forwarding strategies that can handle unscheduled and unpredictable contacts have to be used.

Opportunistic networking is improved by replication of data. By replicating data in the network, node failures and network partitioning can be handled. Several replicas will also increase the speed of dissemination in the network. A node in an opportunistic network will have to buffer data until the node gets an opportunity to forward the data. A node does not know when it will be able to forward the data, so data must be stored, sometimes for a considerable amount of time. Therefore, a buffer management scheme must be implemented to handle stored data.

In spite of lack of information about complete historical and near future network state, a node can forward data by replicating data to nodes that it comes in contact with, in order to increase the probability of delivering the data. The most aggressive type of replication is to epidemically forward data to any neighbor that a node comes in contact with, i.e., flooding data in the network. Only using contacts that have a high probability of forwarding the data to the destination decreases overhead in the network. An abundance of forwarding schemes for opportunistic networks has been presented in [15]. Ranging from the most basic forwarding scheme that sends data to any node in contact to more elaborate schemes where forwarding decisions are based on contact probability.

In the following sections, the three major categories of forwarding methods and their basic properties are described.

Epidemic Forwarding

Epidemic forwarding [40] simply floods data to all nodes within contact range. This scheme has the fastest dissemination speed, if contention and transfer time is ignored. Epidemic forwarding also involves a lot of overhead in terms of occupied buffer space in intermediate nodes. Epidemic protocols with limited number of copies have been proposed. Two such protocols are self-limiting epidemic forwarding (SLEF) [12] and Spray-and-wait [38]. SLEF adopts dissemination speed, i.e., the number of copies, to the density of the network and the traffic load. SLEF also uses a injection control mechanism to ensure that a source node will not flood the network with new data. Spray-and-wait is a limited two hop forwarding scheme where the total number of copies that should be made in the network is defined. The sending node sends a limited number of copies. A relay carries the data until it meets the destination. This forwarding scheme reduce the utilization of the buffers in the network.

Probabilistic Forwarding

Probabilistic forwarding is used to limit data replication and buffer utilization, compared to epidemic forwarding, while striving to get close to optimal delivery ratio. Heuristic forwarding schemes use different metrics to make routing decisions. Prophet [26] uses a node's contact history to estimate a probabilistic metric for each destination in the network. MV routing [9] uses the frequency of previous meetings between peers and their visits to geographical locations. MaxProp routing [8] does not only base routing on contact history, but also use additional mechanism, including acknowledgments, priority to new packets based on hop count, and lists of intermediate nodes. Acknowledgements are flooded in the network to remove the acknowledge message from buffers in the network. Lists of intermediate nodes

are used to avoid sending the same packet to a node twice.

Social Based Forwarding

In social based forwarding, social ties between nodes are used to make decisions on how to forward data. Nodes that have many contacts or are bridging communities may be better relaying nodes than a node that historically been in contact with a destination node. Based on the assumption that social ties can help in making routing decisions. Social routing algorithms try to estimate the ties between nodes with the help of contact patterns. One aspect is to find patterns of social communities and central nodes that binds together the communities [11, 16]. Another aspect is the willingness of nodes from different groups to forward data [25].

Bubble Rap [16] uses a distributed community detection [17] algorithm that calculates individual *centrality*, identifying hubs, bridges, and communities. Social forwarding focuses on finding nodes that travel between communities. SimBet [11] use ego-centric centrality and nodes' social similarity to route data. Messages are forwarded towards nodes with higher centrality to increase the probability to find a relay node that will meet the destination. Identifying central nodes to route through could lead to congestion in the node. The central nodes have to drop messages when their buffers are full. Li et al. [25] are using social ties to calculate the willingness to forward message for other nodes. They also take contact probability and buffer space into account in their forwarding process. Taking buffer space into account lowers the risk of congesting single, central, nodes.

2.2 Content-Centric Networking

In content-centric networks, communication semantics change from "where" to "what" [27]. Traditionally, communication in networks is host-centric. Data is sent to or requested from a specific host. Furthermore, data is routed from a sender to a receiver based on the address of the end points. Content-centric networks, as the name suggests, request and route data based on the content and not who the sender or the receiver is. That allows the user to focus on the data they want rather than having to specify a location of the data.

Several content-centric architectures have been proposed over the years [10, 19, 23, 39]. DONA [23] suggest to replace DNS using a flat name-based anycast architecture. The routing infrastructure can redirect users towards the closest provider of the requested content. DONA is based on TRIAD [10] with additional security in form of authenticity. JUNO [39] is a content-centric request/reply interface that acts as a middleware for applications. JUNO addresses the diverse nature today's of content-centric networks.

Pluggable components are used to deal with the heterogeneity in applications and delivery mechanisms. In Named Data Networks [19] data is routed based on the unique name of the requested data. Interest packets which contains the names of the requested data can be propagated along multiple paths towards the data. The data is then sent along the reverse path, enabling requests to be merged and responses divided on the way back to requesting nodes. Content may be meaningful to several users, therefore routers cache content to meet subsequent requests.

Content-centric architectures [24, 29, 34] for opportunistic networks based on human carried mobile devices, have been proposed. These architectures use metadata to describe the content. The content centric Haggle architecture [29, 34] provides a generic publish/subscribe Application Programming Interface (API) to applications. Applications utilize the API to disseminate and gather data from the network. Applications are agnostic about the underlying communication technology. Haggle make use of the communication technologies at hand, that suit the need of the application. Wireless ad hoc podcast [24] uses data forwarding based on subscriptions of podcast feeds. Dissemination of data is done by pulling content from a neighboring node. The requesting node queries for entries from its subscribed feed channels and the responding node replies to the request. When the requesting node has downloaded all content from the subscribed channels, the node will download random content to improve dissemination of data that is new or has few subscribers. Non-subscribed feeds are also forwarded to maintain diversity of content in the network. Forwarding non-subscribed feeds enables new channels to be spread in the network.

2.3 Haggle

Haggle [29, 34] is a content-centric network architecture for opportunistic networks. Haggle has been used, in this work, as a platform for evaluating data dissemination and congestion avoidance. Haggle is targeted as a device-to-device communication platform, but can also use nodes as data mules [37]. Instead of traditional host-to-host communication, where data is tagged with a host address. Haggle adopts a publish/subscribe like API. In Haggle, data is published into the network as *data objects*. A data object contains the data and *metadata* describing the content. Metadata contains *attributes* that are name-value pairs that nodes can subscribe to. By expressing *interest* in an attribute a node subscribes to data. The interest can also be assigned a weight used in ranking data, which will now be described in further detail.

2.3.1 Data Dissemination

Data dissemination is based on the match between the interests expressed by nodes in the network and the metadata describing the data. A data object

that matches the interests of a node will be pushed from the current holder to the interested node. Every time a data object is sent, it is replicated. The buffer on the sending node is not affected. The effect of this dissemination scheme is that the data that nodes have a common interest in will be spread in the network. Data that are of interest for only a few nodes are less likely to be spread. To improve dissemination of data with few subscribers, nodes can also carry data objects on behalf of other nodes in the network. Data is sent to nodes that are more likely to meet the subscriber in the future, or forward data objects closer to the subscriber, see Section 2.3.2.

Local Search

The content that is going to be spread in the network is determined by a local search, i.e., searching the database on the sender with the interests of the neighboring node (receiver) as input. The local search is based on late-binding of data objects to interested nodes. This is done by using the interests stated in a neighbor's node description and searching the local data store for matching attributes in the metadata of all data objects. The returned list of matching data objects is ordered according to the neighbors interests. Data is sent in the ranked order, assuring that the best matching data objects will be received first, in case the connection is lost. This solution favor the best matching data objects. The least matching data objects will be less likely to be sent, since nodes exchange only a fixed number of data objects in the initiation phase of each contact. After the initial exchange of data objects nodes will not share more data objects during the remaining part of the contact unless new data objects are received. If a data object is received from a neighboring node, or an application, a resolution will be made to find the nodes that are interested in the data object. If there is a match, the data object will be sent.

2.3.2 Forwarding

The basic functionality of disseminating data is done through forwarding based on the interest of a neighboring node. Data is first shared when two nodes come in contact. A search is then done for data objects matching the interest of the new neighbor. During the time a node is in contact all data objects, either coming from an application or from another connected node, will be forwarded to the neighboring node if the metadata matches the interest of the node.

Data is also forwarded to nodes that are not interested in the data, but are more likely to meet an interested node in the future, thereby acting as a relaying node. Forwarding through a relaying node occurs in two events. The first event is when two nodes meet, the node determine which other nodes in the network that the new neighbor is more likely to meet then

itself. The node then matches data with the interests of the set of nodes the neighbor is more likely to meet. The other forwarding event occurs when a new data object is added in a Haggle node either from an application or another Haggle node. The new data object will be sent to a limited number of interested nodes. If the node is a neighbor the data object can be sent directly to the node. If the node is not a neighbor, the data object is sent through a third party node that is more likely to meet the interested node. This type of forwarding is only based on the content and the interests of the nodes.

Chapter 3

Data Dissemination Overhead

One of the main goals of data dissemination is to deliver data as efficiently as possible using a small amount of overhead. Overhead includes control traffic for data exchange and data replication. Control traffic consists of negotiation about what data to exchange. Replication of data uses additional bandwidth as well as storage space on intermediate nodes.

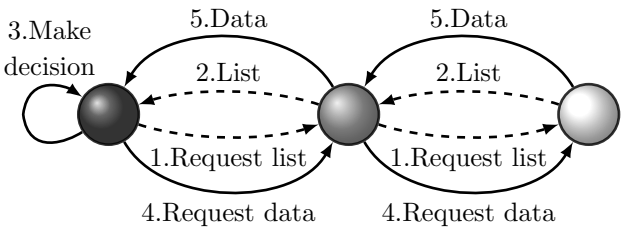


Figure 3.1: Data request over multiple hops in a pull based delivery scheme requires the destination to ask the source for a list of data from which the receiver can request data.

Data dissemination, in content-centric networks, can be divided into two main categories, pull based and push based dissemination. Disseminating data with a pull based scheme requires several steps, see Figure 3.1. Before data requests can be made, lists of available data have to be exchanged (1, 2). The black node makes a decision of which data to request (3). Next, data from the list have to be requested from the source (4) and then the source sends the data (5). Pulling data from a source node requires the destination to have an end-to-end connection with the source, or be in direct contact with the source. This type of data dissemination, called pull based, is used in ad hoc wireless podcast [24].

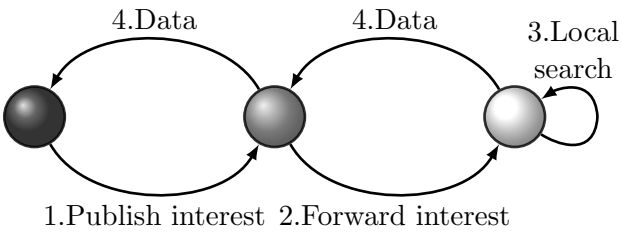


Figure 3.2: Data requests in a push based delivery scheme is done by publishing interest in data. Interests can be forwarded over several hops and a node with matching data will then send data in response to the published interest.

Haggle uses a push based data dissemination model where nodes share what they are interested in and the neighboring node push data that match the interest. It reduces overhead of sending a list, getting a response, and then requesting data. However, it shifts the decision about what to disseminate to the sending node, see Figure 3.2. Nodes are making requests by publishing interest in attributes (1). The interests of a node are spread in the network through node descriptions, that are disseminated the same way as data (2). At a later point in time, a node with data that match an interest will send the data through a node that has a higher probability to meet or forward data closer to the requesting node (3,4). The data is then delivered to the interested node (5). No direct connection is required at the time when data is requested. The requested data can be delivered from disconnected nodes using a node moving between partitions of the network. The store-carry-and-forward functionality makes delivery space and time independent, illustrated in Figure 3.3. A node does not have to be in contact with the source of the data to be able to request and receive data. The request is made when the node publish its interests and the data can be received through a node that relays the data. But, the interest of a node has to be spread in the network by disseminating copies of node descriptions. The overhead will increase with the size of the network. Furthermore, nodes must store the interest of other nodes to enable delay tolerant forwarding of data.

In Paper C, we show that if nodes share knowledge of what other nodes are carrying, the number of unnecessary pushes of data objects can be reduced but the amount of control traffic increases. Sharing knowledge increases the overhead involved when determining which data to send, but it reduces the total amount of overhead in the network. To be able to spread data over several hops in a delay tolerant manner, Haggle uses a mechanism of forwarding data that are not in the interests of the node carrying the data. In ad hoc wireless podcast [24] it is achieved by sending random data

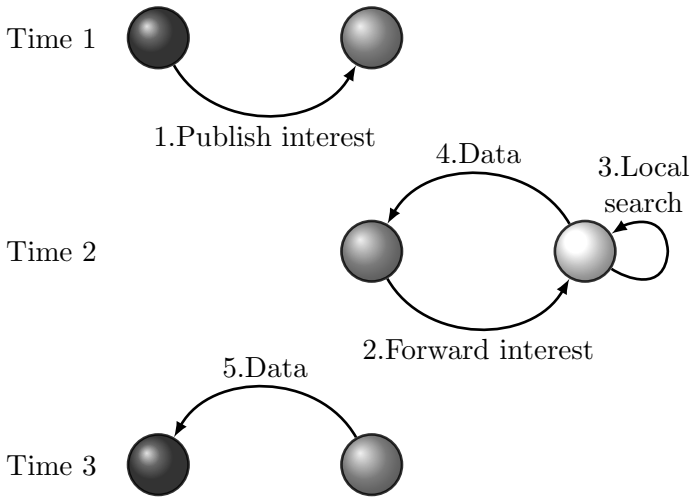


Figure 3.3: Data distribution over a disconnected end-to-end path using a push based delivery scheme. Since the source makes the decision on what to send based on the published interest of the requesting node, request and response can be decoupled in time and space.

to a node in order to increase delivery ratio for the system as a whole. As shown in Paper C, to increase delivery ratio, in a Haggle network, nodes have to know what other nodes in the network already have, otherwise only the most popular data will be replicated. All nodes in the network will do the same thing, resulting in large number of copies of the most popular data, starving the network of less popular data.

Chapter 4

Congestion Avoidance in Opportunistic Networks

In networks with end-to-end connectivity, congestion avoidance consists of two major components. A mechanism to handle *buffer overload* in intermediate nodes and end-to-end data *rate control* based on dropped packets and round trip time. Choosing what data to drop when the buffer is overloaded is decided on the, congested, intermediate node. Data rate control, on the other hand, is typically done through network feedback in the form of delayed and missing acknowledgments. Opportunistic networks are characterized by partitioning and unpredictable contacts and contact times. Data is forwarded and stored in intermediate nodes until the next forwarding opportunity occurs. End-to-end feedback is therefore unfeasible to retrieve. However, overloaded buffers in intermediate nodes can drop data to ease congestion. Buffer management is necessary in opportunistic networks since nodes have limited buffers. Redundancy, through data replication, is used to increase the probability of delivery, but it also increase buffer usage.

In an opportunistic network data dissemination is done through data replication. Multiple copies of data increases the chance for data to be delivered but it also congests the network. By limiting the number of copies of data in the network, node congestion could be avoided. When forwarding to third party nodes, the load of carried data that is not of interest, of the relaying node, increases. In a network built of devices with limited capabilities, where buffer space is one of them, a buffer management scheme has to be provided. The main issue is how to decide what should be dropped from the buffer. In Paper D, we find that it is better to drop data based on local data replication information than the accumulated interest. With accumulated interest, we mean the total interest collected from all received node descriptions. A buffer management strategy that drops data randomly performs second best in most scenarios. This indicates that one of the most important feature of buffer management is that the relayed data is

exchanged, making it possible for new data to be relayed.

As an addition to the DTN architecture [1], alternative custodians [35] were proposed to avoid congestion in a delay tolerant network. In DTN, only one copy of a custody message exists in the network. It is carried by a node that has taken responsibility to deliver the message, closer or directly, to the target node. Using alternative custodians, congestion is avoided by moving messages between nodes in the network. Less congested nodes take over custody of a message, thereby offloading the congested node.

In Paper C, we measured delivery ratio in a small DTN network and looked at the overhead involved in dissemination of data. When forwarding was used, we observed that the number of forwarded data objects increased significantly, while the delivery ratio only had a modest increase. We realized that some sort of congestion avoidance is needed to counteract over-utilization of buffer space on nodes. We also observed that the most popular data is forwarded through all nodes in the network. Creating an epidemic spread of popular data while constraining less popular data. This means each node's buffer will contain the same data, which limits delivery of less popular data and, therefore, also limits the delivery ratio for the whole network.

A buffer management strategy that provides counter measures for this problem has to be implemented. In Paper D, we evaluate a set of buffer management strategies that consider interests and replication of data. We evaluated both strategies that take the content-centric nature of the network into account as well as replication based strategies. Content-centric strategies use the interest of other nodes in the network when deciding what should be dropped from the buffer. Replication based strategies drop data based on the number of copies that the nodes has made, i.e., sent to other nodes.

Chapter 5

Transfer Ordering in Opportunistic Networks

The ordering of data transfer is often overlooked in opportunistic routing. The standard metrics, such as, delivery ratio, delay, and hop count are used to evaluate how well opportunistic forwarding protocols work. The effect of transfer ordering on these metrics has initially been investigated. RAPID [6] can either order transfers to minimizing the maximum delay, or achieve a high delivery ratio, i.e., not missing the deadline. If a deadline (TTL) is passed, the message will be dropped. The ordering is done by estimating the delay for each message using metadata received from other nodes in the network. Each node gets a global view of how messages are located in buffers in the network. With that information, the delay is estimated. Li et al. [25] have proposed Social Selfishness Aware Routing (SSAR) that takes several aspects into account when ordering data in the forwarding decision, in a network were there exists one copy of each message. The ordering is based on packet priority, that is assigned by the source node and updated when received by a relaying node. The second aspect is delivery probability, which is based on the probability that the relaying node can deliver the packet to the destination before the packets expiration time, and the probability that the packet will be dropped because the relaying node does not have enough buffer space. A social aspect is also considered, the willingness of the relaying node is exchanged between nodes in contact, if the willingness is positive the neighboring node is considered as a possible relay node.

Both RAPID and SSAR focus on a network with host-addressed, one-to-one, communication, the quality/value of the data that is delivered is not considered. Haggle is a data-centric network where data is sent from one-to-many, base on the interests of nodes. In Paper E, we evaluate how data should be ordered if each message is not of the same interest for the recipient. The assumption is that all data does not have the same importance for the

recipient. Based on that assumption, data is ordered before it is transferred. The goal to yield a short delay for the most relevant data. In Haggle, data is ordered according to how well it matches the interests of the recipient to be able to deliver the most interesting data first. The matching process, that results in a *score*, uses two components: the interests of the node and the associated weight, and the attributes (metadata) that describes the data. The interest is used to make a selection from all the data to generate a subset of data that is going to be ranked using the weight of the recipient interest. Weights are assigned by the user to each individual interest. The attributes in the data are not weighted. The weight of all matching interests are added together and normalized using to total weight of all interests resulting in a score. This is done for all data that have at least one matching attribute. All data objects are then ordered according to the score, see Figure 5.1.

Node i Interest

a:w_1 | **c**:w_2 | **d**:w_3

Node j

Data	Relevant	Score	Order
a | **c** | **d**	✔	$\frac{w_1+w_2+w_3}{w_1+w_2+w_3}$	**1**
b | **c** | **d**	✔	$\frac{w_2+w_3}{w_1+w_2+w_3}$	**2**
b | **c** | e	✔	$\frac{w_2}{w_1+w_2+w_3}$	**3**
b | e | f	×	0	-

Figure 5.1: Data ordering according to a node's interests. The recipient i has weighted interests causing the depicted ordering of j's data items.

Evaluating the effects of ordering depends on what the strategy aim to promote. Balasubramanian et al. [6] evaluate the ordering using delay and delivery ratio, depending on what the ordering is optimized for. In Paper E, we use an established metric in information retrieval, normalized Discounted Cumulative Gain (nDCG) [20], to evaluate how well the strategies perform at collecting data of interest to the users. The preferred order in which each node would like to receive the data, i.e., the order it ranks all data objects in the network, starting with the most interesting data. That is the ideal order in which the received order will be compared against using the following calculation:

$$nDCG^n[T] = n_T(s_1 + \sum_{i=2}^{T} s_i/log_b(i)) \tag{5.1}$$

where s_i represents the score for the i^{th} ranked data object of the received data. The values are discounted using the logarithm of the ranked position (order) of the score, i.e., progressively giving less value for the lower

ranked data items. The calculation yields a result between 0 and 1, where 1 represents that all the T most relevant data items have been received. Taking the average of the nDCG value for all nodes in the network gives a value for how well the T most relevant data items for each node has been disseminated at a specific point in time.

Chapter 6

Evaluation Methodology

We have chosen to use an experimental methodology in our evaluations. There are three experimental methodologies that support evaluation of network architectures and protocols. Real-world, emulation, and simulation experiments, all three have their advantages and disadvantages. The key is to find the method that best capture the properties of the scenario/protocol that is under evaluation.

With all network layers involved, real-world experiments yield the most credible results since no assumptions or simplifications have to be made when creating a model or emulating a network layer. However, in experiments with large disconnected networks it is hard to coordinate movement to create connectivity patterns and, therefore, also hard to repeat. Real-world experiments are also costly both in terms of man-hours and equipment, even with scenarios describing the mobility of the participant, giving them instructions where and when to move [28]. DTN scenarios are usually run over several hours and sometimes days, which makes it impractical to use real-world experiments. Furthermore, DTN scenarios include network partitioning and disconnectedness, which requires large areas and a lot of man-hours in setting up equipment and performing experiments. The benefits of using real-world experiments are that all layers and interaction between layers are included. But, at the same time it is a drawback when it comes to reproducibility of results when there are variations in the wireless conditions. Variations can depend on interference from other radio sources, but also obstacles in the environment. One type of variation source in an experiment are the carriers of the mobile devices, and humans in the area.

Emulation can be used to automate real-world experiments. Emulated wireless communication or another type of communication medium are typically used together with real nodes. Network changes in emulated experiments can be performed without human interaction and mobility. By automatically filtering traffic on the emulated network, connectivity patterns can be created without manual labor, i.e., people walking around with devices.

Thereby, eliminating the variation in timing that real-world node-mobility can introduce and the variation in the radio environment that nods are subjected to. Timing constraints and network conditions are easier to maintain in an emulated environment, hence it is easier generating repeatable results. Using emulated experiments running real or virtual computers, the greatest benefit is that the same implementation and operating system can be used as in the target environment for the protocol or architecture.

Emulation can be done on different levels. Emulab [41] is a distributed, scalable, emulated platform for repeatable experiments. The approach Emulab has taken is to emulate everything down to MAC, physical, and radio frequency (RF) environments which are all three simulated. MeshTest [36] is a wireless multi-hop testbed that connects real computers and emulates the RF environment. Drawbacks of using real computers are scalability and the cost of equipment. MeshTest uses the same amount of computers as real-world experiments, but it also requires an RF matrix switch, and a server that provides experimental control, for connecting all computer's Wi-Fi cards. The advantage is automation of experiments.

Both real-world and emulation experiments are limited in practical scalability. Simulations, on the other hand, scale well with number of nodes and computational capacity, and are useful when exploring different parameters in a systematic manner. Experiments with extreme parameters (e.g., mobility over large areas, long scenario duration) can be performed without complicated set-ups. However, simulations use simplified models of reality and the implementation under evaluation. If the model is too simple or incorrect, simulations can yield erroneous results. Also, re-implementing a protocol for a simulator can introduce discrepancies. Therefore, the accuracy of the results is determined by how good the abstraction model is. Simulations are suited to evaluate the high level design and principals of a protocol or architecture. The most commonly used simulators for evaluation of DTN scenarios are: DTNSim2, ns-2, and the ONE simulator [2, 13, 22]. All three are event-based and scale to hundreds of nodes. A drawback is the lack of real time simulation. Event-based simulations does not take computation time into consideration, each event will be executed before the next event is executed. Therefore, simulation makes it hard to evaluate implementations and the feasibility to run them, in real time, on real hardware. For example, if a database access used to query what data that is going to be sent to a neighboring node takes more time then the duration of a contact, it will not be detected using an event driven simulation.

Of the three experimental evaluation methods available, we have chosen an emulated testbed where unaltered code can be used. With unaltered code, we can use the same code base for real-world experiments as we do in our emulations. Using an emulated testbed, we can automate experiments and repeatably coordinate connectivity. In our emulation we focus on the data-centric forwarding done in Haggle and not the underlying layers.

6.1 Haggle Testbed

The goal with the testbed was to build an evaluation platform that is cost effective and easy to manage. The requirements were that we can control connectivity between nodes, run the Haggle implementation without code alterations, and automatically execute predefined experiments.

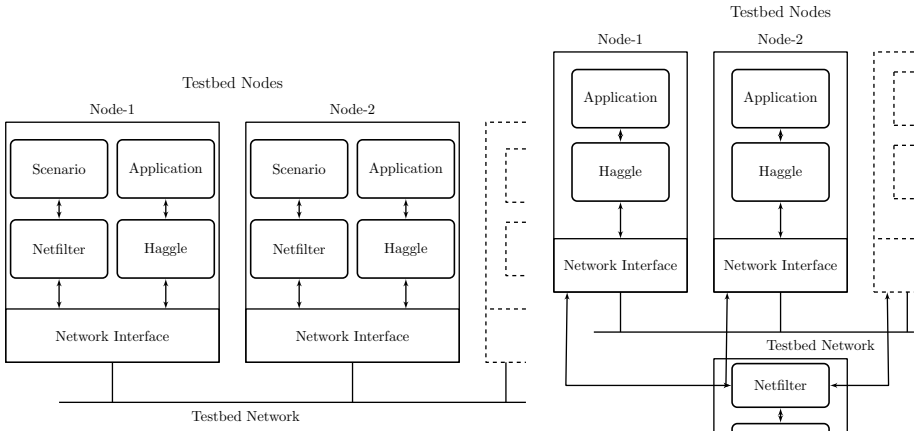
We selected to use a single physical desktop computer that hosts several virtual computers. Using one physical computer makes both deployment and experiments easy to manage. One important aspect when running several virtual testbed nodes on one physical computer is the distribution of the shared resources among the nodes. Xen [7] shows a high degree of performance isolation when it comes to network, CPU memory, and disk access [32]. At the same time providing low overhead compared to other virtualization tools.

The first generation of the testbed, described in Paper A, was based on decentralized scenario execution using a modified version of APE [28]. Execution of a scenario—including timing, application start up, and network filtering—was done decentralized on the testbed nodes, see Figure 6.1(a). The second generation, described in Paper B, moved from scenario execution on nodes to a centralized scenario execution to ease management and remove the need of a synchronization beacon to avoid clock drift between nodes, see Figure 6.1(b). A clock beacon would require a control channel, since the network often is disconnected. Network filtering was also moved from the node to the testbed host. When filtering is done on the testbed host, traffic that should't reach a testbed node will never enter the node's network stack. It also reduces network traffic on the emulated Ethernet bridge, used by all nodes.

Experiments on the testbed are run in real time, which enable us to evaluate the performance of the Haggle implementation. During stress tests of the Haggle implementation we found issues with short contact and inter-contact times, as well as other more general bugs. However, running experiments in real time also put some stress on the testbed. Due to memory, the number of nodes on the testbed is limited to 80. However, in a scenario with 4000 contacts, the testbed was found to be limited to 50 nodes or less.

All measurements on the testbed are done by logging functionalities in the Haggle implementation. All events in the Haggle kernel are logged, e.g., neighbor discovery, data sent and received. Only measurements concerning timing and the performance of the testbed are done by the testbed itself. In this manner, we can use the same logging and, therefore, the same analysis tools for real-world experiments as well as for the emulated experiments.

The data rate of the network interfaces on the testbed nodes is not limited. It would be possible to throttle the traffic between nodes, but we have not used it in any of our experiments. However, if the amount of data two nodes can exchange is restricted by the connection time and



(a) Decentralized control

(b) Centralized control

Figure 6.1: Schematic view of the testbed described in Paper A (a), and the second generation described in Paper B (b).

bandwidth, it becomes important to throttle the data rate. Focus has been on the dissemination of data according to interests. Since we send a small amount of data in each encounter, all data have time to be sent no matter the underlying medium. Inherent properties in Haggle, when it comes to communication between nodes, limits the traffic between two nodes, only a fixed number of data objects are exchanged during a contact.

A visualization tool was also added in the second generation of the testbed. The tool is based on a network monitoring tool called Vendetta [33]. It enables the user to remotely follow network topology changes and the internal state of the Haggle nodes. It is not only a tool for monitoring experiments, a user can manually control an experiment and use it as a tool for debugging the design and the implementation of the architecture.

6.1.1 Connectivity Traces

Two different types of connectivity traces are used depending on the scale of the experiment. Large scale experiments are either built on synthetic traces generated with a Markov model [18] or using available datasets that contain traces collected in a large scale [5]. Smaller experiments, up to 25 nodes, are built on contact traces collected using mobile phones. Students and faculty members, on campus, were asked to carry a mobile phone for a day. The phone was running Haggle using either Wi-Fi or Bluetooth as the communication medium. Traces were extracted from the collected data logs acquired from the experiment. These traces can be used in the testbed

to be able to reproduce the same scenario evaluating different settings in Haggle and running various applications.

6.1.2 Scalability Issues

Haggle is using a database for all data, normally stored on disk. The drawback with such a solution is that each data-query requires disk access. A situation that results in many queries is when two nodes—that are using forwarding—meet. A query has to made to resolve what data to send to the neighbor as well as a query for each of the nodes that the neighbor has a higher probability to meet.

On a testbed where all nodes use the same physical disk, running an architecture with an intense disk access workload does not scale well. Nodes mount their file system using a disk image on the testbed host's hard drive. Using several hard drives would increase the scalability until the limit of the SATA channel is reached. Another feasible solution would be to extend the testbed over several computers.

Chapter 7

Discussion

There are questions left to answer after the initial work in congestion avoidance in data-centric opportunistic networks. In Paper D, we show that, using the basic data dropping strategies, dropping data based on data replication information performs better than using the interests of nodes in the network. Probably because of stale data in the buffer. With stale data we mean data that there is no interest for or every node that is interested in the data has already received it. The interest does not change during the experiment. Hence, only incoming interests, from nodes that have not been seen before, will change the order in which data is ranked. Another undesired property that we want to avoid is that a node that drops the most forwarded data, which was the best performing strategy, will eventually end up with data in its buffer that no other nodes are interested in or has already received. Also, all nodes will probably store a similar set of data, if the amount of data in the network is small. Adding a dropping strategy that randomly drop data along with most forwarded data, should remove data that no other nodes are interested in. This could potentially increase the delivery ratio.

Congestion in an infrastructure content-centric networks could be handled in a similar manner but with the addition of communication between nodes. Copies of data will be found along a path, or several paths, in the network, towards different interested nodes. Since the nodes are connected, they can communicate what each node should store so that all nodes do not store the same data. Instead, different data could be stored on the nodes on a path and thereby distributing the cached data.

In Paper C, we look at how data dissemination is affected by the knowledge of what other nodes are carrying and how well this information is spread. However, if the information about what other nodes are carrying is not epidemically disseminated, the spread is limited by how interests are distributed in the network. As an example, for node A to receive node B's interest, node A have to meet node B or a relaying node have to forward node B's interest. But, node B's interest will only be forwarded to node A

if node A and B have a common interest. How this affects data dissemination has not been analyzed. Performing experiments with different data and interests distributions would give us a better understanding about data dissemination in Haggle, i.e., dissemination based on weighted interests.

Using the results from Paper D and E, it would be interesting to see how transfer ordering and node congestion can be handled simultaneously. Node congestion should be considered when data is ordered for transfer. Taking global satisfaction and node congestion into account could lead to less overhead, i.e, fever unnecessary transfers between node-pairs, at the same time satisfying several nodes interests per transfer.

Both Paper C and D would have benefitted from using real-world traces in the evaluation. The traces we have used in Paper C and D are small and/or artificial. This makes interpretation of the results easier but it reduces the possibility to draw more general conclusions. In future works we would like to use small artificial topologies to understand what happens in the network and in the interaction between nodes. But, we would also like to use several real-world traces that have different characteristics to be able to draw general conclusions.

Chapter 8

Summary of Papers

This thesis consists of the following papers:

Paper A

A Testbed for Evaluating Delay Tolerant Network Protocol Implementations
Fredrik Bjurefors and Oscar Wibling
In Proceedings of the 5th Swedish National Computer Networking Workshop, Karlskrona, May 2008

We evaluate the possibility of running a testbed with several nodes using virtual computers on a single physical computer. The scenario execution and network filtering are described. A set of small scenarios were run to show repeatability. The results were also compared to real-world experiments. The results were similar, but with lower RTT in the emulated testbed due to ideal network connectivity. To demonstrate scalability we ran a scaled down version of Haggle that does neighbor discovery but does not exchange data. The reason for the simplification lies in that the previous version of Haggle used a lot of processing power.

Paper B

Haggle Testbed: a Testbed for Opportunistic Networks
Fredrik Bjurefors, Per Gunningberg, and Christina Rohner
In Proceedings of the 7th Swedish National Computer Networking Workshop, Linköping, June 2011

We describe the structure of the updated version of the testbed. The testbed was implemented with the host node as the controller of experiments to reduce timing discrepancies between nodes and test runs. We evaluated

timing accuracy on three different levels: interface (Ethernet), Haggle and application. The evaluation shows that the timing is within a few milliseconds, which is reasonable when the probing interval for neighbor discovery in Haggle is five seconds. We also looked at how well the testbed scale. The limiting factor was shown to be disk access. During our evaluation we switched from a standard SATA disk to using a SSD disk. After the switch we were able to run more than twice as many nodes.

Paper C

Interest Dissemination in a Searchable Data-Centric Opportunistic Network
Fredrik Bjurefors, Erik Nordström, Christian Rohner, and Per Gunningberg
Invited paper, European Wireless 2010, Lucca, Italy, April 2010

Data dissemination in opportunistic networks is done by replicating data in the network. We evaluate the benefits/implications of sharing information about what data other nodes are carrying. This allows nodes to avoid further disseminating data that a neighbor has already received from other nodes in the network. We evaluated the performance in terms of delivery ratio and overhead. We show that the overhead in sharing this information is less then what we gain in reducing unnecessary transmissions.

It would have been interesting to evaluate the idea in a set of larger networks to be able to draw some more general conclusions. If the network has got many hops from a source to a destination, the Bloom filter—storing information about what the node is carrying—has to be sent over several hops, this would lead to an increase in overhead.

Paper D

Congestion Avoidance in a Data-Centric Opportunistic Network
Fredrik Bjurefors, Per Gunningberg, Christian Rohner, and Sam Tavakoli
In Proceedings of ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2011), Toronto, Canada, August 2011

We looked at the effect of congestion and how to avoid congestion in data-centric opportunistic networks. Node congestion means that a node cannot forward any data. Data has to be dropped to free up space in the buffer to enable new data to be forwarded. In opportunistic networks, the lack of an end-to-end path, i.e., disconnection and mobility in the network, leads to disruption in node communication. Therefore, decisions on what to drop are made by using local information. We propose a set of basic dropping strategies that use data replication and the interests of other nodes

to make a decision on what to drop. To evaluate the proposed dropping strategies we use a set of artificial network topologies where a central node connects clusters of nodes, thus leading to node congestion. We show that dropping based on data replication information performs better then using the interests of the nodes in the network.

Paper E

"Making the Most of Your Contacts: Transfer Ordering in Data-Centric Opportunistic Networks"
Christian Rohner, Fredrik Bjurefors, Per Gunningberg, Liam McNamara, and Erik Nordström
In Proceedings of the Third International Workshop on Mobile Opportunistic Networking (MobiOpp 2012), Zurich, Switzerland, March 2012

We evaluated several strategies to order the data being transferred. The selection is done by using the interest of the recipient or the accumulated knowledge of what nodes in the network are interested in, and the attributes describing the data. The ordering is then done randomly, LiFo, or using the score, calculated using the recipients specific interest per attribute. We use a quality metric (nDCG) to evaluate if nodes receive the most interesting data for each individual node in the network. We show that local satisfaction has the lowest delay when it comes to delivering the most interesting data (high quality). But, over time global satisfaction will yield a better result.

Chapter 9

Conclusions and Future Work

In this thesis, we have shown that it is feasible to create an emulation testbed where unaltered code can be evaluated. Experiments with small variations in timing can be performed in real time. Small timing variations guarantees repeatable experiments for reproducible results. The testbed provides a platform where different algorithms and applications can be evaluated under the same conditions, e.g., contact times, data load, and connectivity patterns. However, large scale experiments are not well suited for the proposed testbed as nodes are executed on the same hardware in real time. Then disk and CPU resources cannot be guaranteed and, therefore, can create variations in results between experiments.

The testbed has been used for the three experimental studies of data-centric opportunistic networks presented in this work. In Paper C, we showed that data dissemination can be significantly improved by sharing information about what other nodes in the network store. In Paper D, buffer management strategies are investigated. Drop decisions based on data replication perform better than interest based buffer management. However, randomly dropping data from the buffer yield the second best results among all strategies. This shows that exchanging data in the buffer makes the most significant improvement. In Paper E, we investigated how the transfer order of data impact what is delivered to recipients. Satisfying local interests results, on average, that nodes receive highly relevant data with a short delay. But, over time the satisfaction of global interests yield a better result, delivering a ledger portion of the most relevant data. Also, it would be interesting to evaluate the interaction between node congestion avoidance and transfer ordering, looking at how buffer space advertisements can benefit dissemination and impact ordering.

This could help us to design forwarding strategies that take into account, not only which nodes another node is more likely to meet, but also

use the information about what other nodes in the network already store. The information could be used both when it comes to what should be disseminated and what could be removed from the buffer. But, to get a better understanding how dissemination works and what affect the distribution of interests have on data dissemination, we are going to evaluate Haggle in different topologies with varying properties. So far, in all of these evaluations, all nodes have the same behavior and are cooperating. Another area that is of interest to investigate is what happens when all nodes in the network are not willing to cooperate. In particular, is it possible to detect that a node is not cooperating.

Bibliography

[1] Delay Tolerant Networking Research Group, http://www.dtnrg.org, 2010.

[2] Homepage of the DTNSim2 Delay-tolerant Network Simulator, http://watwire.uwaterloo.ca/DTN/sim/, 2010.

[3] The Interplanetory Network Special Interest Group (IPNSIG), http://www.ipnsig.org, 2010.

[4] The N4C project, http://www.n4c.eu, 2010.

[5] http://www.crawdad.org, 2011.

[6] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. DTN Routing as a Resource Allocation Problem. *SIGCOMM Comput. Commun. Rev.*, 37(4), 2007.

[7] Paul R. Barham, Boris Dragovic, Keir A. Fraser, Steven M. Hand, Timothy L. Harris, Alex C. Ho, Evangelos Kotsovinos, Anil V.S. Madhavapeddy, Rolf Neugebauer, Ian A. Pratt, and Andrew K. Warfield. Xen 2002. Technical Report UCAM-CL-TR-553, University of Cambridge, Computer Laboratory, January 2003.

[8] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *In Proc. IEEE INFOCOM*, 2006.

[9] Brendan Burns, Oliver Brock, and Brian Neil Levine. MV Routing and Capacity Building in Disruption Tolerant Networks. volume 1, mar. 2005.

[10] David R. Cheriton and Mark Gritter. TRIAD: A Scalable Deployable NAT-based Internet Architecture. Technical report, 2000.

[11] Elizabeth M. Daly and Mads Haahr. Social Network Analysis for Routing in Disconnected Delay-Tolerant MANETs. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '07, New York, NY, USA, 2007. ACM.

[12] Alaeddine El Fawal, Jean-Yves Le Boudec, and Kave Salamatian. Self-Limiting Epidemic Forwarding. Technical report, 2006.

[13] Sally Floyd and Steve McCanne. The Network Simulator NS-2. http://www.isi.edu/nsnam/ns/, 2010.

[14] S. Guo, M. H. Falaki, E. A. Oliver, S. Ur Rahman, A. Seth, M. A. Zaharia, U. Ismail, and S. Keshav. Design and Implementation of the KioskNet System.

[15] Chung-Ming Huang, Kun chan Lan, and Chang-Zhou Tsai. A Survey of Opportunistic Networks. mar. 2008.

[16] Pan Hui, Jon Crowcroft, and Eiko Yoneki. BUBBLE Rap: Social-based Forwarding in Delay Tolerant Networks. In *Proc. ACM MobiHoc*, 2008.

[17] Pan Hui, Eiko Yoneki, Shu Yan Chan, and Jon Crowcroft. Distributed community detection in delay tolerant networks. In *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, MobiArch '07, New York, NY, USA, 2007. ACM.

[18] Seok Hwang and Dongsoo Kim. Markov model of link connectivity in mobile ad hoc networks. *Telecommunication Systems*, 34, 2007.

[19] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking Named Content. In *Proc. CoNEXT '09*, New York, NY, USA, 2009. ACM.

[20] Kalervo Järvelin and Jaana Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *ACM SIGIR*, 2000.

[21] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. *SIGARCH Comput. Archit. News*, 30(5), 2002.

[22] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *Proc. Simutools '09*, Brussels, Belgium, 2009. ICST.

[23] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A Data-Oriented (and Beyond) Network Architecture. *SIGCOMM Comput. Commun. Rev.*, 37(4), 2007.

[24] Vincent Lenders, Gunnar Karlsson, and Martin May. Wireless Ad Hoc Podcasting. In *Annual IEEE Communications Society Conference on*

Sensor, Mesh and Ad Hoc Communications and Networks (SECON), June 2007.

[25] Qinghua Li, Sencun Zhu, and Guohong Cao. Routing in Socially Selfish Delay Tolerant Networks. In *INFOCOM, 2010 Proceedings IEEE*, march 2010.

[26] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic Routing in Intermittently Connected Networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3), 2003.

[27] Michael Meisel, Vasileios Pappas, and Lixia Zhang. Ad Hoc Networking via Named Data. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*, MobiArch '10, New York, NY, USA, 2010. ACM.

[28] Erik Nordström, Per Gunningberg, and Henrik Lundgren. A Testbed and Methodology for Experimental Evaluation of Wireless Mobile Ad hoc Networks. *Testbeds and Research Infrastructures for the Development of Networks & Communities, International Conference on*, 2005.

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

[30] Alex (Sandy) Pentland, Richard Fletcher, and Amir Hasson. DakNet: Rethinking Connectivity in Developing Nations. *Computer*, 37, 2004.

[31] C.E. Perkins. Mobile IP. *Communications Magazine, IEEE*, 35(5), may 1997.

[32] Benjamin Qutier, Vincent Neri, and Franck Cappello. Scalability Comparison of Four Host Virtualization Tools. *Journal of Grid Computing*, 5(1), March 2007.

[33] Olof Rensfelt, Lars-Åke Larzon, and Sven Westergren. Vendetta - A Tool for Flexible Monitoring and Management of Distributed Testbeds. May 2007.

[34] James Scott, Pan Hui, Jon Crowcroft, and Christophe Diot. Haggle: A Networking Architecture Designed around Mobile Users. In *Proc. of the 2006 IFIP Conference on Wireless on Demand Network Systems and Services (IFIP WONS 2006)*, 2006.

[35] Matthew Seligman, Kevin Fall, and Padma Mundur. Alternative custodians for congestion control in delay tolerant networks. In *CHANTS '06: Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, New York, NY, USA, 2006. ACM.

[36] Matthew Seligman, Brenton D. Walker, and T. Charles Clancy. Delay-Tolerant Network Experiments on the MeshTest Wireless Testbed. In *Proc. CHANTS '08*, New York, NY, USA, 2008. ACM.

[37] Rahul C. Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks. may. 2003.

[38] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. 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*, New York, NY, USA, 2005. ACM.

[39] Gareth Tyson, Andreas Mauthe, Thomas Plagemann, and Yehia Elkhatib. Juno: Reconfigurable Middleware for Heterogeneous Content Networking. In *In Proc. 5th Intl. Workshop on Next Generation Networking Middleware (NGNM), Samos Islands, Greece*, 2008.

[40] Amin Vahdat and David Becker. Epidemic Routing for Partially-Connected Ad Hoc Networks. Technical report, 2000.

[41] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, Boston, MA, dec 2002. USENIX Association.

Paper A

Fredrik Bjurefors and Oscar Wibling. *A Testbed for Evaluating Delay Tolerant Network Protocol Implementations.* In Proceedings of the 5th Swedish National Computer Networking Workshop, Karlskrona, May 2008.

A Testbed for Evaluating Delay Tolerant Network Protocol Implementations

Fredrik Bjurefors Oskar Wibling

Department of Information Technology
Uppsala University

Abstract

We present our delay tolerant network testbed and methodology, for testing opportunistic networking protocols and applications, without having to modify the software. To support multiple operating systems running on a single physical computer, virtual machines are at the core of the testbed. Scripted communication opportunities are used to model connectivity in the network. A set of benchmark connectivity models are available to perform evaluations with predictable protocol behavior; dynamic topologies from real-world traces are also supported. Using this environment, we evaluate the testbed with a delay tolerant application and compare the outcome to that of a real-world experiment. Finally, we discuss our results and the limitations of the testbed.

1 Introduction

A new opportunistic networking paradigm has evolved, from a vision where heterogeneous mobile devices form dynamic clusters and take advantage of network resources when they become available. This is a deviation from the traditional end-to-end connectivity paradigm used in the Internet today.

Examples of Delay Tolerant Network (DTN) scenarios, with opportunistic use of intermittent connectivity are: community networks in rural regions, rescue operations, social networking, and wireless sensor networks. DTNs are often evaluated using simulations. These, however, tend to introduce discrepancies from the actual implementations. We present our DTN testbed and methodology for evaluating opportunistic networking software, without having to rewrite it; thereby minimizing discrepancies. The goal is to achieve reproducible experiments in a controlled and easy to manage environment.

2 Related Work

The Kasuari Framework [2] is built upon Xen and the ns-2 [8] network simulator. It is designed as a self-contained wireless network testbed that runs on a single standard computer. All nodes in the testbed are connected to two networks. One network controlling the testbed and the other network is simulated in ns-2, to create movement patterns and to log traffic. The framework is tested using a modified version of AODV with added store and forward capabilities.

Song and Kotz [7] use trace-driven simulations, based on real-world contact logs, to evaluate a number of opportunistic forwarding algorithms. They found them all inefficient; experiencing low delivery ratio, high resource usage, or a combination thereof. We also find performance problems in our studied opportunistic application; however, we provide and utilize a trace-based emulation framework instead of simulations.

3 Testbed setup

We have created a testbed to emulate a mobile opportunistic network and conduct repeatable tests in an easily manageable environment. See Figure 1 for an overview of the testbed. The Xen virtual machine monitor [1] is at the core of the testbed. Xen supports execution of multiple guest operating systems on a single physical machine; the guest systems are monitored by a Xen host system. Since virtual machines are used, it is possible to use an unmodified version of the DTN protocol under test.

3.1 Architecture

We use a Copy-on-Write (CoW) implementation to manage a large number of guest systems with minimized maintenance and hard disk space requirements. With CoW we can use one read-only file system for all nodes to ensure an identical setup. This reduces the creation time for new nodes since only a small swap file needs to be created. Each node has a difference file which the changes from the read only file system are written to. This also ensures a clean setup every time a node is rebooted.

3.1.1 Virtualized APE (Ad hoc Protocol Evaluation)

APE [4] is a Linux distribution created with the goal of making the process of performing complex real-world tests as easy as possible. It focuses on smooth deployment, high ability of customization and ability to easily run several protocol implementations for comparisons. The DTN testbed uses the main APE script to interpret the scenario file and to collect logs. Modifications have been made to make APE work with the Xen configuration; these include

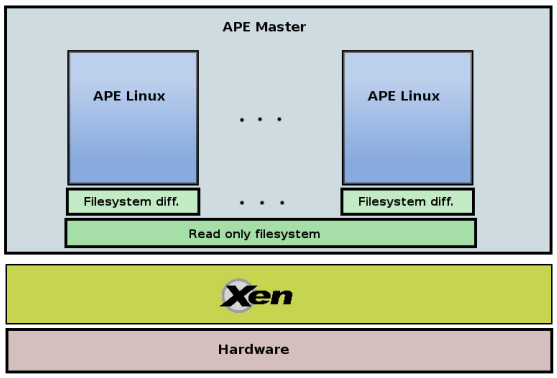


Figure 1: Testbed architecture.

adding remote initialization of scenario execution and log acquirement. The topology changes in a network are emulated by filtering traffic according to the APE scenario file.

3.1.2 Communication

The virtual nodes in the DTN testbed communicate with each other using a bridge interface, that is, a virtual Ethernet segment. Every node is assigned an IP address. The segment is visible from outside of the testbed. It is therefore possible to have several computers connected to each other to allow scaling, or having the testbed interact with real DTN nodes potentially running on mobile devices. A disadvantage is that traffic from potential DTN nodes not part of the test, interfere with the experiment. The network segment in the test thus has to filter such traffic.

3.2 Scenario Execution

In addition to defining the emulated movement of nodes, scenario files are used to initialize the nodes and to generate traffic in the network. See Figure 2 for an example of such a file. All scenario scripts begin with choreography instructions. These instructions apply to all nodes in the test.

Each node is assigned a node ID, starting from zero. The instruction on a particular line in the scenario file is executed by the node with the corresponding ID. There are three different actions defined.

```
node.0.action.3.command=mac_disable 4
```

This line will interpret `mac_disable 4` as a command. In this particular command all traffic from node ID 4 will be filtered out on node ID 0 as

if they were out of communication range. There is also a corresponding `mac_enable` command. To inform the user about what is happening on a node, a message action can be used.

```
node.0.action.3.msg="Block node 4"
```

The third action is `duration` which specifies a time (in seconds) for the node to wait. Further details regarding scenario scripting are given in the APE manual [4].

```
# generic setup/teardown instructions ---------------------------------

choreography.scenario.title=Ring 100sec app:msggen
choreography.total.nodes=5
choreography.total.duration=130
choreography.startup.command.0=startup
choreography.startup.command.1=generate_ip2mac
choreography.startup.command.2=tcpdump -i $IFNAME -s 200 -w /var/log/tcpdump.apelog &
choreography.shutdown.command.0=killproc tcpdump
choreography.shutdown.command.1=copy_files
choreography.shutdown.command.2=pack_files


 node 0   ------------------------------------------------------------------

node.0.ip=192.168.77.10
node.0.action.0.command=mac_disable all
node.0.action.1.command=msggenstarter "192.168.77 10 14 10 3 100 10 1"
node.0.action.1.duration=0
node.0.action.2.command=mac_enable 1
node.0.action.2.duration=0
node.0.action.3.command=mac_enable 4
node.0.action.3.duration=100
node.0.action.4.command=mac_disable 1
node.0.action.4.duration=0
node.0.action.5.command=mac_disable 4
node.0.action.5.duration=30
node.0.action.6.command=mac_enable all
node.0.action.7.command=exit

 node 1   ------------------------------------------------------------------

node.1.ip=192.168.77.11
```

Figure 2: Example of a scenario file.

3.3 Connectivity Control

3.3.1 Converting Traces to Scenarios

Scenarios are derived from connectivity traces by running a translation script (`trace2scenario.py`). A static or dynamic trace can be used as input. A sanity check on the trace is first performed to eliminate connections to external nodes, where overlapping connections between two nodes are found; the one with the shortest duration is removed. Each valid connection results in a bidirectional link between two nodes. The resulting output is a scenario with filter instructions for each node. The scenario is also given a name and instructions about which application and node(s) that will generate traffic.

3.3.2 Packet Filtering

Network topology changes are performed by controlling connectivity with traffic filtering. This is done by setting *iptables* [3] rules on each node in

the testbed and thereby blocking traffic from certain nodes. Connectivities are the ones registered in the trace file used to generate the scenario; during each interval, lossless communication is therefore enabled or disabled. All traffic between nodes is initially blocked, except loopback traffic and packets from the host node that enables debugging.

4 Methodology

4.1 Traffic Generation

Any application can be used to generate traffic in the network. Which application to use and how the traffic is generated is configured in the scenario script (see Section 3.2).

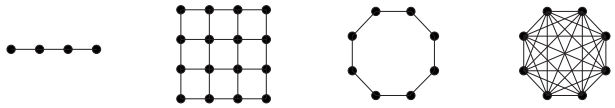


Figure 3: Basic topology types

4.2 Benchmark Tests

We use a number of static benchmark scenarios when validating protocols at an early development stage; these are designed to give a predictable protocol behavior. Four different benchmarking topologies; line, ring, star and grid, are used for this purpose (See Figure 3). For each of these topologies, the complexity is varied in terms of needed forwarding strategy, node traffic/CPU load, and number of participating nodes.

For the set of benchmarking tests we use the following.

- Four basic topology types: line, grid, ring, full (see Figure 3).
- For each of the topologies mentioned above, varying the number of nodes involved, i.e., starting with 3-4 nodes and then increasing.

The following list constitutes a comparison of the scenarios in terms of forwarding and expected average node load:

- Full connectivity: No forwarding needed, all nodes equally loaded.
- Line connectivity: Forwarding needed to reach all nodes, nodes more loaded (in terms of message throughput/CPU load) towards middle of line.
- Ring connectivity: Forwarding needed, nodes equally loaded.

- Grid connectivity: Forwarding needed, nodes more loaded towards center of mesh.

We have constructed scripts to facilitate the generation of APE scenario files for different combinations of topology and number of nodes.

For our future work, we intend to add the following.

- For each topology type, and number of nodes, vary the connectivity pattern as follows: Start by running experiments with static connectivity, then move on to varying connectivity in a controlled way and finally use arbitrary connectivity.

By varying connectivity in a controlled way, we mean to select the sequence of link changes manually, in order to test a certain behavior. As an example, consider the following ring topology: A-B-C-A. We may then use a scenario in which node A wants to send packets to node B. During the course of the scenario, the link between nodes A and B goes down during some time interval. The aim of the test is to see how the forwarding of packets from A to B is affected when the link break occurs. Varying the connectivity arbitrarily will be done using different random number distributions as well as iMote connectivity traces.

4.3 iMote Trace Tests

In order to capture realistic, complex, and dynamic topologies, we can also use real-world trace files as input to our emulations. These can be complex when it comes to predicting the outcome from a scenario built on such a trace, but the result can give an indication of what will happen in a real world situation. We have used iMote trace [5] files containing node pairs and time intervals between which the nodes have been registered to be connected. APE scenarios are generated from trace files as described in Section 3.3.1. For our experiments the *cambridge/haggle* [5] traces were used. These contain sightings of Bluetooth devices (iMotes) carried by users in office environment and conference settings. The smallest cambridge/haggle trace is one with ten nodes spanning three days. For practical reasons, during the testing of our evaluation methodology we truncated the scenario after 20 minutes.

5 Evaluation

5.1 Evaluation of the Testbed

The main goal of the emulation testbed is to enable repeatable experiments which scale to a realistic number of nodes. We comment on these properties in the sections that follow.

5.1.1 Repeatability

Since the testbed is an emulation testbed, the repeatability property follows almost by design. Connectivity changes are controlled using scripts and the execution of scenarios is therefore quite deterministic. Of course, there are still variations in the results due to interactions between, e.g.:

- Small timing variations in the initialization of nodes
- The operating system scheduling of tasks
- Packet collisions on the Xen bridge

In Figure 4 we show RTT CDF plots from a couple of individual test runs with four virtual nodes, fully connected, with each node transmitting one ping per second to each of the other three nodes for a duration of ten seconds. We see that the curves are similar, which supports the expectation that experiments are repeatable. The most significant difference is that the RTT experienced at haggle node 2 (*haggle-2*) in the second test run is higher than in the first run. Such differences can occur due to the factors described above, because of errors in the tested software, or a combination thereof. When performing production experiments we will execute a large number of test runs and take their average to reduce the impact of such individual discrepancies. The standard deviation in a series of test runs should then be small if reproducibility is to be claimed.

5.1.2 Scalability

To evaluate scalability, we performed experiments with 30 nodes running a periodic neighbor discovery used in Haggle [6]. Haggle provides opportunistic forwarding to mobile applications. We used the testbed with this software for ten minutes during which we logged CPU utilization. The measurement data is given in Figure 5. We see that the CPU utilization at all 30 nodes is maintained at a low level throughout the scenario. It rarely goes above 5% for any of the nodes. The maximum number of virtual nodes that can run on each physical machine is, however, application dependent.

5.2 Evaluation of the Haggle Architecture

As a proof of concept, we have performed a simple evaluation of Haggle.

5.2.1 Emulation Test Runs

For the benchmarking scenarios on Haggle, we have run the following tests in emulation.

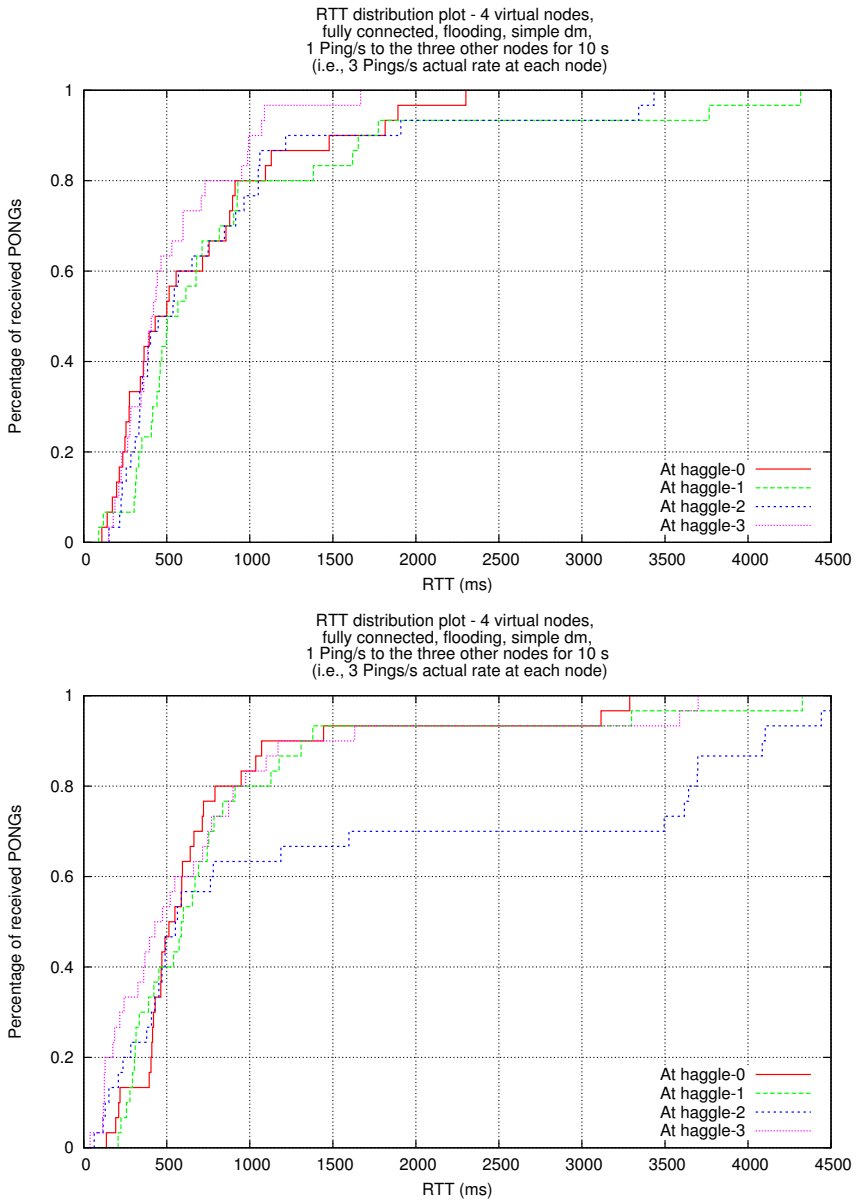


Figure 4: RTT distribution plots from two emulation test runs with four virtual nodes.

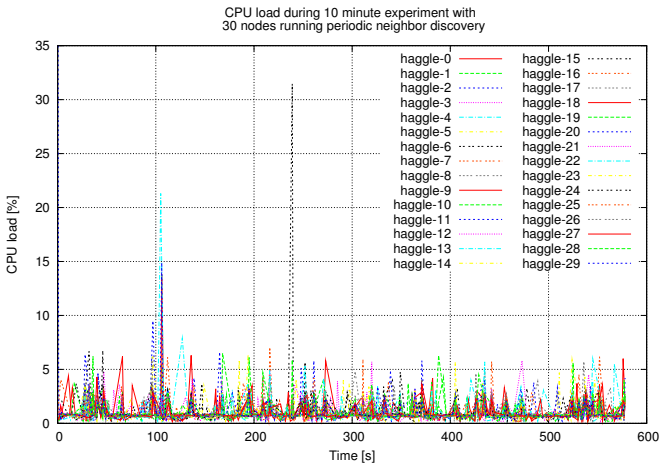


Figure 5: Example CPU load for 30 node experiment. Nodes are labeled "haggle-0" - "haggle-29"

- 2/4/5/7/10 nodes, fully connected, flooding, 1 ping/s to a number of the other nodes for 10 seconds
- 4/5/7/10 nodes, connected in a line, flooding, 1 ping/s to a number of the other nodes for 10 seconds
- 4/5/7/10 nodes, connected in a ring, flooding, 1 ping/s to a number of the other nodes for 10 seconds

These experiments uncovered scaling problems in the Haggle software that was the current one at the time. We traced the causes of the problems to both the implementation and the underlying architecture. This had the effect that an improved Haggle architecture was subsequently developed and work on a new implementation commenced.

For the experiments using iMote traces, we ran initial tests using the truncated scenario files described in Section 4.3. These tests were mainly executed to confirm that the testbed worked properly for trace-based experiments; however, we observed similar performance problems as in the benchmarking scenarios.

5.2.2 Real Device Test Runs

To validate our Haggle emulations we ran complementary experiments in the real world, using laptop computers. We performed the following real-world experiments:

- 4/5 nodes, fully connected, flooding, 1 ping/s to a number of the other nodes for 10 seconds

- 4/5 nodes, connected in a line, flooding, 1 ping/s to a number of the other nodes for 10 seconds

- 4/5 nodes, connected in a ring, flooding, 1 ping/s to a number of the other nodes for 10 seconds

We then compared our real-world results to the corresponding emulated experiments and were able to conclude that the round-trip latency curves had a similar shape, although the emulated experiments yielded lower RTTs. We interpret this result as a validator for the usefulness of our emulations to predict real-world behavior. We attribute the lower RTTs to the idealized network connectivity model which uses a Xen bridge and packet filtering.

We could further verify that the relative node load was as expected according to what is described in Section 4.2.

6 Limitations

The testbed is limited in the number of nodes that can be used on each physical machine, although this number is quite high (see Section 3.1). As more nodes are added, the average processing power per node also goes down, which can have consequences for the test execution times as well as how realistic the results will be. The question is if real devices will have a comparable processing power to the emulated ones. An option to increase the maximum number of nodes is to connect a second physical machine and run Xen on that one as well. This is possible since the virtual interfaces are reachable within the same network segment. With the upgrade to a new Linux kernel in the testbed we also expect to be able to increase the number of virtual nodes on each machine.

7 Concluding remarks and future work

We now have a testbed on which we can perform repeatable emulations of DTN software running in networks of tens of nodes.

For future work, we intend to run emulations using the cambridge/haggle traces [5] for their full duration. That way we will also investigate how Haggle behaves when run over a duration of days. When we start running production tests we will also include the grid scenarios from our benchmarking suite.

We plan to introduce bandwidth limitations in the testbed to emulate different types of interfaces, such as Bluetooth, 802.11a/b/g and RF. To reduce the risk of influencing the tests we would also like to separate the control network from the test environment.

References

[1] Paul R. Barham, Boris Dragovic, Keir A. Fraser, Steven M. Hand, Timothy L. Harris, Alex C. Ho, Evangelos Kotsovinos, Anil V.S. Madhavapeddy, Rolf Neugebauer, Ian A. Pratt, and Andrew K. Warfield. Xen 2002. Technical report, 2003.

[2] Christoph Dwertmann. A testbed framework design for protocol development and evaluation in wireless mobile ad-hoc and delay tolerant networking environments. Master's thesis, Universität Bremen, 2006.

[3] The netfilter/iptables project. http://www.netfilter.org/.

[4] Erik Nordström, Per Gunningberg, and Henrik Lundgren. A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks. In *Proc. TRIDENTCOM'05*, pages 100–109, Washington, DC, USA, 2005. IEEE Computer Society.

[5] James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle.

[6] James Scott, Pan Hui, Jon Crowcroft, and Christophe Diot. Haggle: a networking architecture designed around mobile users. In *Proc. WONS'06*, pages 78–86. INRIA, 2006.

[7] Libo Song and David F. Kotz. In *Proc. CHANTS'07*, New York, NY, USA.

[8] The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns/.

Paper B

Fredrik Bjurefors, Per Gunningberg, and Christian Rohner. *Haggle Testbed: A Testbed for Opportunistic Networks.* In Proceedings of the 7th Swedish National Computer Networking Workshop, Linköping, June 2011

Haggle Testbed: a Testbed for Opportunistic Networks

Fredrik Bjurefors Per Gunningberg Christian Rohner

Department of Information Technology
Uppsala University

Abstract

Due to node movement and connectivity changes, performing repeatable experiments with opportunistic networks is difficult using real world experiments. We present a testbed for repeatable experiments, based on virtual computers. By using structured scenarios we can control nodes, applications, and connectivity, creating a platform for repeatable experiments. Connectivity traces from real world experiments can be replayed on the testbed. Then we also have the possibility to change settings and vary the application, i.e., the content in the network. We show that we can perform experiments with negligible variations in scenario execution as well as in the obtained results.

1 Introduction

Performing real world experimental evaluation of opportunistic networks is costly, both in terms of equipment and labor. Experiments often run for hours up to days with several nodes involved. This makes it impractical to repeat as real world experiments and it is difficult to achieve similar connectivity patterns.

We present the Haggle testbed that was designed to provide a platform for experimental research and evaluation of the Haggle architecture [6]. Features include: control and management of testbed nodes, a graphical user interface (GUI), and analysis scripts. The testbed can be operated either manually by use of a GUI or automatically in batch mode by running scripted experiments.

The testbed allows for controlled experiments where different forwarding strategies, resource management methods, or other algorithms can be compared under the same conditions and in different scenarios. Controlled connectivity is essential for repeatable experiments. Performing experiments with many nodes requires a convenient management interface. The nodes

should run the same code version, follow a specific scenario, monitor, and log the experiment for later analysis.

Running Haggle on virtual machines on a single computer gives advantages in synchronized connectivity control but also allows scaling the number of nodes without requiring hardware, installation, and configuration overhead compared to running nodes on separate machines.

2 Testbed

We have created a testbed based on virtual computers. A virtual computer represents a mobile node in a dynamic network topology, which is created by controlling the connectivity between the nodes. The focus has been on ease of use, maintenance, and deployment. The testbed is controlled by scenarios, but it can also be controlled and maintained through a graphical point-and-click interface that facilitates debugging. Ease of deployment is solved by using a standard desktop computer with an automated configuration and installation.

2.1 Architecture

The testbed architecture is based on Xen [2] and Linux. Xen is a virtual machine monitor that allows concurrent execution of multiple operating system instances on a single physical computer. It has a low CPU, memory, and disk overhead and runs a single kernel per virtual machine, while ensuring resource isolation [8].

Figure 1 depicts a schematic image of the testbed structure. The *testbed host* controls the testbed and executes experiments using scenarios. Scenarios are well defined description of the participating nodes, the connectivity, and the content generated during an experiment. *Testbed nodes* run a copy of a minimal Linux installation, which only executes the Haggle architecture and an application that generates content in the network. Nodes have a full network stack and a virtual Ethernet interface connected to a testbed network.

Additionally, the testbed can also be controlled and monitored through a *remote host*. Control and execution will then be done through a middleware on the testbed host. The status of the testbed and the testbed nodes can be reported back through the middleware.

2.2 Topology control

Topology changes are emulated by filtering the traffic between testbed nodes. Netfilter [5] is used to manipulate traffic flows going in, out, and through a computer. We do not emulate wireless radio, instead we emulate connectivity opportunities. They are represented by a connectivity trace, indicating

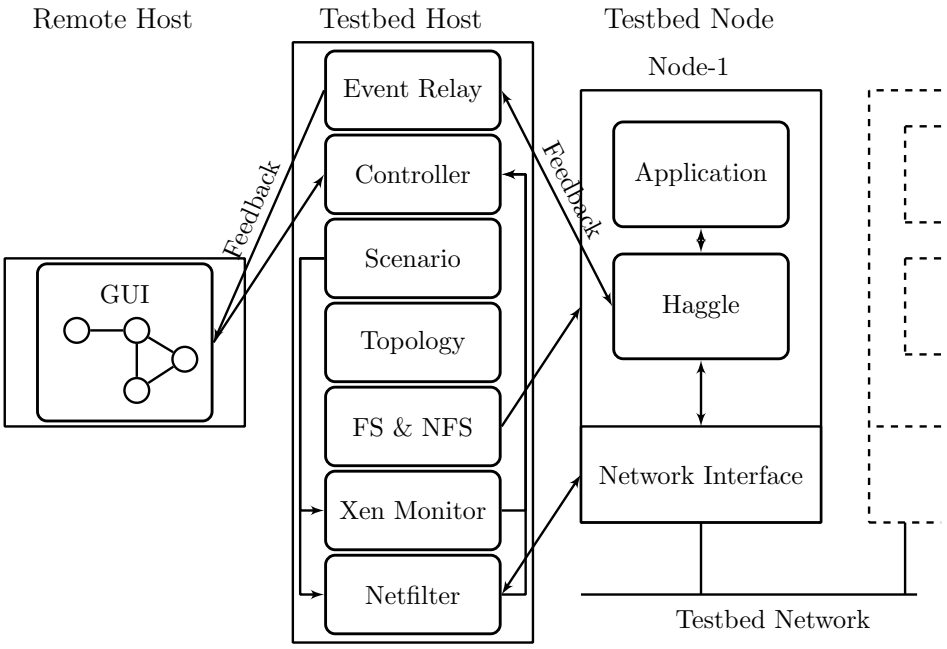


Figure 1: Testbed system structure.

which nodes are connected and at what time, such a trace can be generated using a mobility model, extracted from real world experiments, or other datasets [1]. According to a trace the testbed host filters traffic on the virtual interfaces that connect the testbed nodes to the network bridge. When filtering is done in this way, packets that are destined to a node that is out of communication range of the sender will never be received by the testbed node.

There are drawbacks of using a single physical computer, for instance, since testbed nodes use the same Ethernet bridge we can get contention on the MAC-layer even if two nodes are not in communication range of each other. However, Zhang et al. presented the hypothesis that the testbed network must have the same bandwidth as the total capacity of the ad-hoc network [9], which would also be an appropriate worst case approximation for an opportunistic network. It is possible to throttle the traffic using *tc* to ensure that the available bandwidth is sufficient to emulate the total capacity of the opportunistic network.

3 Haggle

We evaluate repeatability of the testbed using Haggle [6]. Haggle is a data-centric network architecture, designed for mobile nodes, based on a publish/subscribe model. A Haggle node can opportunistically take advantage

of any communication technology of mobile devices, such as WiFi and Bluetooth. A device runs an instance of Haggle to which applications connect.

4 Repeatable Experiments

Our methodology when comparing different protocols and algorithms is based on being able to test them in different scenarios to cover different aspects, and at the same time providing equal conditions within a given scenario. In this section we discuss the repeatability of our testbed to provide equal conditions. We evaluate the repeatability in terms of variation over a number of repetitions of the same scenario on the network topology, Haggle, and application level.

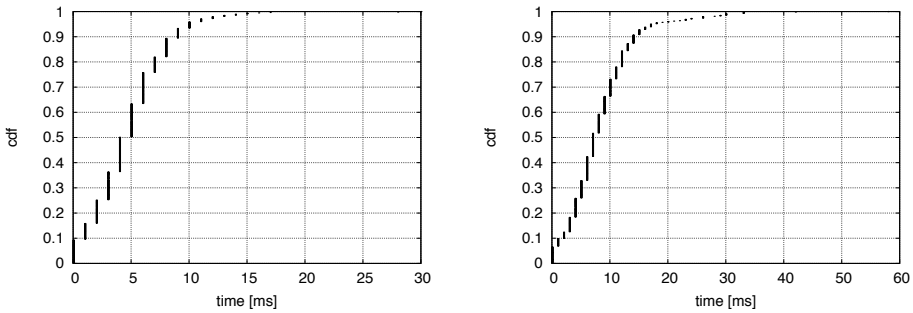
Even though exact repeatability on all levels might be desirable, it is not always possible to achieve because of the complexity of the system and the stochastic behavior of some algorithms within Haggle or applications running on top of Haggle. We limit ourselves to providing repeatability in the timing of a scenario, that is, topology control and starting Haggle and applications.

We evaluate the repeatability of the network topology and Haggle based on a scenario that takes up and down links between three pairs of nodes to test a) the timing spread and b) the ability of Haggle to discover available links. Links are discovered by listening to beacons that other Haggle nodes are sending out. The contact duration in Haggle has a resolution that is dependent on Haggle's beaconing interval. Therefore the trace consist of contact times between the three pairs of nodes of 5 seconds (i.e., the time interval of Haggle beacons), 10 seconds, and 15 seconds respectively. Between every node pair we generate 10 contacts, the time when a link goes up is not synchronized to the Haggle beaconing. The experiment is repeated 20 times.

4.1 Repeatability of the Topology Scheduling

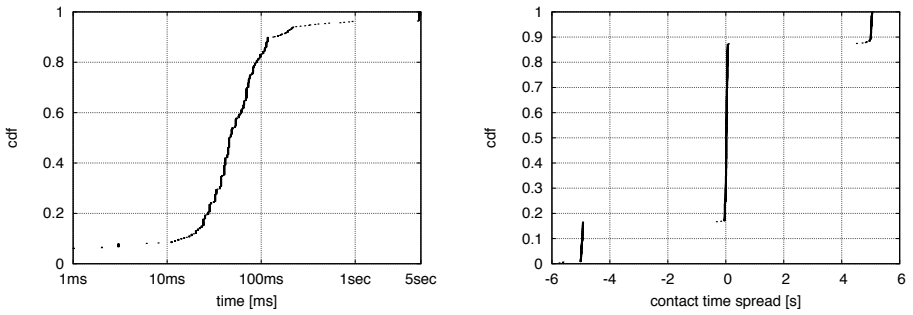
Figure 2 shows the cumulative distribution of link changes (Figure 2(a)) and contact time (Figure 2(b)). The link change distribution is measured when the testbed host changes Netfilter to enable or disable connectivity between a node pair. It gives an indication on the jitter of scenario scheduling in the testbed. The contact time distribution is measured as the time between when a connection between a node pair is enabled and when it is disabled. It gives an indication on how the scheduling jitter affects the contact time, and is expected to be roughly twice the scheduling jitter.

Figure 2(a) shows that the scheduling jitter is less than 5 ms for 50% of the link changes, and less than 10 ms for 95% of the link changes. The maximum observed delay of a link change is 30 ms.



(a) Link changes (b) Contact time

Figure 2: Cumulative distribution of link changes and contact time variation over 20 experiments at testbed level.



(a) Link changes (b) Contact time

Figure 3: Cumulative distribution of link changes and contact time variation over 20 experiments as observed by Haggle.

Figure 2(b) shows the jitter of the contact time between node pairs. Note that the shortest contact time is used as reference, accounting for 5% (20 repetitions) of 0 ms delay. 75% of all contact times are within 10 ms, 95% within 20 ms. The longest observed deviation of contact time is 58 ms.

4.2 Repeatability of Haggle Contacts

Haggle introduces stochastic behavior in many ways. While the topology scheduling is executed on the testbed host, Haggle runs in testbed nodes resulting the typical concurrency issues that are discussed earlier. Relevant for repeatability, Haggle has to be started on separate testbed nodes which involves setting up an ssh connection and starting up Haggle. We cannot control the jitter introduced by that process, however, the startup order is the same for every repetition of the experiment limiting the impact to that jitter.

Haggle itself is a complex system that is to some extent deterministic due

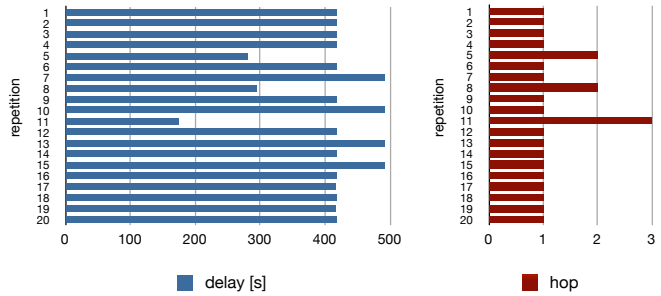


Figure 4: End-to-end delay and number of hops measured for a particular message and node over the 20 repetitions.

to internal interaction through an event queue and a common kernel, but small timing differences (e.g., introduced by context switches or discovery of other nodes) can have a significant impact. In this section we evaluate the jitter in link changes and contact time as perceived by Haggle, analogous to the evaluation of topology scheduling above.

Critical for the jitter of observed link changes is the alignment of device discovery used to discover nodes in communication range. Device discovery for Ethernet is done by sampling connectivity by sending periodic beacons, with a default interval of 5 seconds.

Figure 3 shows the cumulative distribution of link changes (Figure 3(a)) and contact time (Figure 3(b)) as measured using the Haggle log files. The jitter is obviously larger than for the testbed scheduling, with about 80% of the link changes within less than 100 ms but large deviations up to a second and 4% of the contacts that discovers only at the next interval 5 seconds later.

Figure 3(b) shows the difference of the contact time between node pairs. Here, the difference is dominated not by jitter but by scanning perception. This results in 15% of the contact times that are 5 seconds too short and another 15% of contacts that are 5 seconds too long depending on whether a beacon has been missed at the beginning of a contact or at the end of a contact.

4.3 Repeatability of the Application

Repeatability of the application is of course dependent on the timing on the testbed level and the resulting behavior of Haggle. As highlighted in the discussion on the repeatability of Haggle contacts, small timing variations on a lower level can have a significant impact. How data is disseminated through the network is of significance for the application. Every data object follows a dissemination tree that might be very different and those resulting in large differences in end-to-end delays if node contacts are missed.

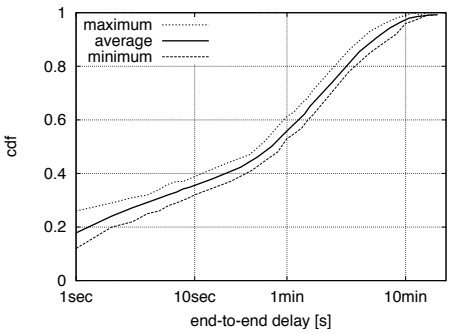


Figure 5: End-to-end delay variations over 20 repetitions of the experiment.

We conducted an experiment with five nodes each periodically creating a message, every 10 seconds, that is epidemically spread in the network. The scenario's trace was generated using a Markov model of link connectivity with a mean contact and inter-contact time of 60 seconds and 240 seconds respectively. The experiment is run over one hour.

Figure 4 shows the end-to-end delay and number of hops measured for a particular message and node over the 20 repetitions. The particular message is chosen at random and shows a typical distribution. We can observe that the delay for the majority of experiments is 417 seconds, but other messages arriving earlier (after 174, 280, and 295 seconds respectively) or later after 492 seconds.

The message was delivered over one hop in most experiments. In a few experiments the message was delivered over two or three hops, and during these experiments we also observed a low delay. This confirms our expectation that a difference in the availability of node contacts can result in significantly different dissemination topologies and hence delays. We assume in this case that there is a node contact just about to terminate at the time the message is created, whereas transmission of the message was still possible in a few experiments but not during all experiments.

The delay spread of the messages arriving after 417 seconds is 1 second, while it is 0 seconds for the messages arriving after 492 seconds. This shows good repeatability under the condition that the message take the same route. The overall delay distribution is shown in Figure 5.

As shown, variation in link connectivity can have an impact on the path a message travels. If the message size used in an experiment is small, the variations in link connectivity will have a greater impact. Messages then have time to be delivered during short contacts. If the message size is large, the transmission will be canceled and since Haggle does not fragment messages the message have to be transmitted during the next contact opportunity.

5 Testbed resource utilization

We evaluate how many nodes it is possible to scale to on a desktop computer. The limitations of the testbed come in terms of CPU and IO capacity. Memory is statically assigned by Xen, in our case 64 MB per testbed node, and the testbed host requires 512 MB according to the specification of Xen. The computer used in these experiments was a Core 2 Duo 3 GHz with 8 GB RAM, a 500 GB SATA disk, running Debian 5.0.8 and Xen 3.2.

All data was acquired using the current Haggle implementation (Haggle version 0.3). Two experiments were done to look into the limitations. The first experiment was performed to study the impact of the number of nodes in the network. In these experiments were the number of contacts fixed to 4000 contacts per hour. Figure 6(a) shows the average CPU and IO load on the testbed. Already with a network with 20 nodes is the IO load 98%. The second experiment studies the impact of the quantity of contacts. The number of nodes used in this experiment is 30. Figure 6(b) shows that an increase in contacts per time unit rise the CPU load, this is due to the fact that a contact initiates several database queries.

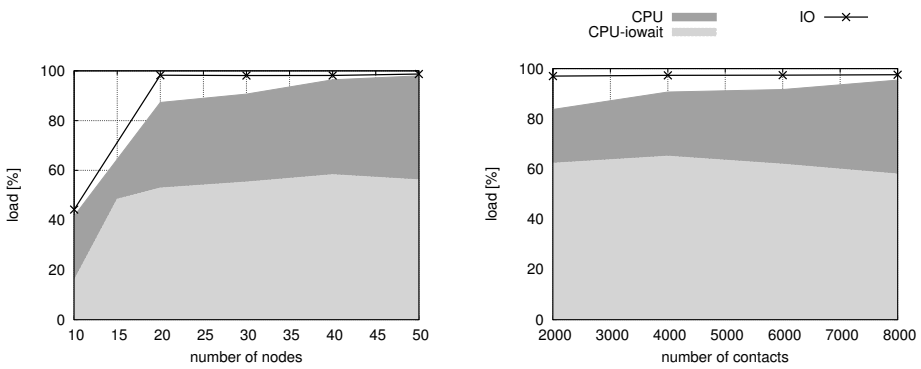
In both the experiment with a fixed number of contacts as well as the experiment with fixed number of nodes, the CPU load curve follows the IO load. This is due to that most of the CPU load is iowait. The CPU-iowait shown in both graphs is the CPU time spent waiting on IO when the CPU otherwise would be idle. With more contacts the CPU has a higher load and therefore less time is spent in iowait idling. The current version of Haggle uses a database stored on disk. This means that all database operations are done on disk and since all nodes share the same disk, it puts a lot of stress on the disk. Additionally, nodes also log all events to disk for analysis.

A second evaluation was done using the same setup except for a 128 GB SSD disk. Figure 7 shows that the IO utilization is significantly lower which means that the CPU will spend less time in CPU-iowait. The main part of the CPU time is now spent in steal (i.e., a process is waiting for CPU).

There are a few solutions to reduce the problem. Nodes could work against a RAM disk. However, this increase the amount of memory required for the testbed. The file system of testbed nodes could be distributed on several physical disks moving the limitation to the SATA channel bandwidth.

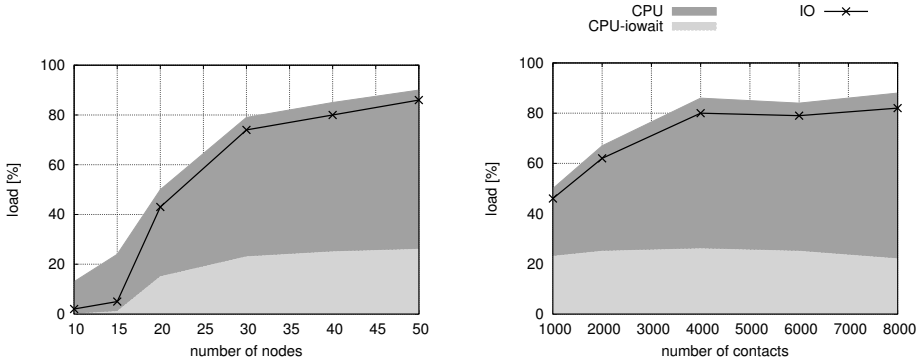
6 Related work

Several emulated network testbeds has been constructed in order to conduct experiments on wireless mobile ad-hoc protocols. The MobiEmu [9] testbed is a software platform for evaluation of ad-hoc network protocols. It uses a master controller, running on a dedicated machine, to execute scenarios. All nodes participating in the testbed runs a slave controller that receives



(a) 4000 contact/h (b) 30 nodes

Figure 6: Resource utilization in terms of CPU and IO using a standard SATA disk.



(a) 4000 contact/h (b) 30 nodes

Figure 7: Resource utilization in terms of CPU and IO using an SSD disk.

instructions, from the master controller, when the next action should be performed. In the evaluation of the platform they use a Linux Beowulf cluster of 51 machines connected to a gigabit Ethernet switch. Running experiments on such a cluster is expensive and cumbersome in terms of equipment and maintenance. However, MobiEmu can also be used in a virtualized environment similar in concept to ours. Their virtualization is based on User Mode Linux which has got a high overhead when it comes to CPU, memory, and disk. Also, the communication between virtual machines is poor [8]. Emulab [4] is based on process-level virtualization running on low-end PCs. Using FreeBSD jail they can provide file system and network name space isolation. A custom virtual Ethernet interface device was developed to support multiple virtual nodes with their own distinct IP addresses on a single physical host. They assume that CPU and memory requirements of initial network applications are modest. Therefore, no resource isolation between

virtual nodes is provided in their solution. However, both MobiEmu and Emulab can, like our solution, run unmodified software using a real operating system. Another system that can run unmodified software, and like our system does not require much resources, is Kasuari [7]. It is based on virtual machines running on top of Xen. Instead of using trace based connectivity, Kasuari is using a mobility model supplied by the network simulator ns2 [3] to simulate the motion and connection between nodes.

7 Conclusions

We have created an environment where we can evaluate different algorithms and application based on the Haggle architecture. We show that it is feasible to run experiments on a testbed built on virtual computers. In the evaluation of repeatability of topology changes, we show that the jitter in scenario execution is negligible compared to connectivity variations induced by Haggle itself. Measurements indicate that 95% of link changes are executed within 10 ms of the scheduled time. This makes us confident in stating that experiments on the testbed at different times will yield results at the application level that are credible, comparable, and reliable.

References

[1] http://www.crawdad.org, 2011.

[2] Paul R. Barham, Boris Dragovic, Keir A. Fraser, Steven M. Hand, Timothy L. Harris, Alex C. Ho, Evangelos Kotsovinos, Anil V.S. Madhavapeddy, Rolf Neugebauer, Ian A. Pratt, and Andrew K. Warfield. Xen 2002. Technical Report UCAM-CL-TR-553, University of Cambridge, Computer Laboratory, January 2003.

[3] Sally Floyd and Steve McCanne. The network simulator ns-2. http://www.isi.edu/nsnam/ns/, 2003.

[4] Mike Hibler, Robert Ricci, Leigh Stoller, Jonathon Duerig, Shashi Guruprasad, Tim Stack, Kirk Webb, and Jay Lepreau. Large-scale virtualization in the emulab network testbed. In *ATC'08*. USENIX Association, 2008.

[5] The netfilter/iptables project. http://www.netfilter.org/.

[6] E. Nordström, P. Gunningberg, and C. Rohner. Haggle: a data-centric network architecture for mobile devices. In *MobiHoc S3 '09*. ACM, 2009.

[7] Jörg Ott, Dirk Kutscher, and Christoph Dwertmann. Integrating DTN and MANET routing. In *CHANTS '06*. ACM, 2006.

[8] Benjamin Qutier, Vincent Neri, and Franck Cappello. Scalability comparison of four host virtualization tools. *Journal of Grid Computing*, 2007.

[9] Yongguang Zhang and Wei Li. An integrated environment for testing mobile ad-hoc networks. In *MobiHoc '02*. ACM, 2002.

Paper C

Fredrik Bjurefors, Erik Nordström, Christian Rohner, and Per Gunningberg. *Interest Dissemination in a Searchable Data-Centric Opportunistic Network.* Invited paper, European Wireless 2010, Lucca, Italy, April 2010.

Interest Dissemination in a Searchable Data-Centric Opportunistic Network

Fredrik Bjurefors, Per Gunningberg, Erik Nordström,
and Christian Rohner
Department of Information Technology, Uppsala University
Box 337, 751 05, Uppsala, Sweden
phone: +46 18 4710000, fax: +46 18 4717244
email: name.surname@it.uu.se
web: www.it.uu.se

Abstract

We present a performance evaluation of interest dissemination in Haggle. Haggle is a data-centric, searchable, network architecture for opportunistic communication. In a data-centric network the driving force for information delivery is the interests in a particular data and other nodes' knowledge about those interests. This paper presents how interests and data are represented and distributed in Haggle. In Haggle data dissemination decisions are made on the sender side. A sending nodes' knowledge of other nodes' interest in data is consequently important for network performance. Efficient dissemination of this knowledge increases the the delivery ratio of useful data but also the overhead. We present experimental results with different dissemination strategies.

1 Introduction

Communication in opportunistic networks is based on sporadic and intermittent contacts between mobile nodes. These communication opportunities are used to forward data to other nodes. A node can itself be interested in the transferred data, or it can store-carry-and-forward it to other interested nodes. The overall forwarding issue is how efficiently data can be disseminated to particular nodes or to all nodes in an opportunistic network.

Opportunistic forwarding is used when an end-to-end communication infrastructure is lacking. Examples include rescue and recovery scenarios, or data distribution in remote areas such as mountains [3] or developing countries [12]. In these areas data is carried by users on mobile devices, such as a mobile phone, and fits applications where delay is not critical, e.g. email.

Informed decisions on what to forward to which node is therefore key to maximize delivery of data to interested nodes, while minimizing overhead. Overhead comes with the associated knowledge about the interest and data of each node which need to be spread to other nodes. Server-client models, where nodes request data, are in general not practical in opportunistic networks because of the lack of end-to-end connectivity. Such models would lead to long delays and out-dated delivery since the requests must opportunistically diffuse to the server(s). Instead, we investigate into a publish-subscribe model used in the Haggle [2] [18] architecture. The model makes progressive resolution on the sender side when to forward data and it is done hop-by-hop. The resolution is based on subscribed interests of data in a node and forwarding opportunities to other nodes. Doing the decision on the sender side means less latency until the data is sent to other nodes compared to a receiver decision.

We investigate the delivery ratio of data to subscribing nodes depending on the knowledge of what other nodes have in their data stores. Informed knowledge minimizes unnecessary transmissions and deliver fresh data. On the down side, exchange of knowledge among nodes means overhead. We analyze the overhead for different interest dissemination strategies. We believe that we are the first to experimentally evaluate the overhead of content distribution strategies in opportunistic networks.

The paper is structured as follows. In Section 2, the Haggle node description and forwarding model is described. Further, in Section 3 the methodology and the scenario used are described. Section 4 describes and discuss the results of our experiments. Related work is reviewed in Section 5. Last, we conclude our findings in Section 6.

2 Haggle

Haggle [16] is a data-centric network architecture based on a publish/subscribe style API. It can opportunistically take advantage of any communication technology of mobile devices, such as smart phones and laptops. A device runs an instance of Haggle to which several applications can connect and becomes a node in a Haggle network.

2.1 Node descriptions

A Haggle node has a database where it stores *data objects*. Data objects consists of *data* and *metadata*. Data could be anything from a small sensor data to a large file. Metadata contains *attributes* that describes the actual data. Nodes expresses their *interests* in data also in the form of *attributes*. A *node description* is a data object that consist of the interests and the metadata of a node. When two nodes are in contact they exchange their node descriptions. The interest attributes are matched against metadata

attributes. If there is a match, data is exchanged. With the exchange of node descriptions, each node builds its own *relation graph* of what other nodes wants and what they already have [15]. This graph is used to determine which data objects to push to the node of contact.

2.2 Bloom filter

Bloom filter are space efficient data structures of fixed size that can be used to determine if an element, i.e., data object, is a member in a set. Bloom filters [5] are here used in each node to avoid unnecessary transmissions of data objects that the receiving node already has. Filters are used to represent data and node descriptions of previously met nodes. The Bloom filter representation is included in the node description exchanged with other nodes.

Before sending a data object, the sending node checks its internal Bloom filter for the corresponding receiving node. If that data object is within the Bloom filter that data is hold back. The receiving node also looks into its internal Bloom filter to check if the data object is already in its data store. If the receiver already has the data object, it will send a *reject* reply to indicate that it already possess the data object and that the sender should not send the complete data object. In case it does not have the data the receiver will instead send an *accept* reply. The sender will then send the complete data object. The Bloom filter mechanism reduces the size of exchanged node descriptions as well as the number of unnecessary deliveries of duplicate data objects.

Two different techniques for spreading node descriptors are compared with respect to delivery ratio and overhead. Both are combined with or without Bloom filters.

2.3 Original spreading of Node Descriptors

Node descriptions are exchanged the first time two nodes meet. By comparing interest attributes with data attributes the nodes will rank the best matches. Note that the best matches may be to other nodes beyond the one in contact with. The nodes will also identify for which nodes the contact node would be a good choice for carrying data objects to, i.e. the nodes it will have a high probability to meet. Hence, node descriptions are exchanged among those nodes that share enough interests so that the rank of the match is high enough to be among objects exchanged. This is the original mode of operation in Haggle.

There is a limit on the number of data objects that are sent to a node during a contact encounter. The default limit setting, ten data objects, is used in this evaluation. The next time these two nodes meet again, ten new data objects will be exchanged, since each node update their respective

relation graph (if changes have been made to the description). A limit is used to avoid a single node from congesting other nodes.

2.4 Epidemic spreading of Node descriptions

An additional feature investigated in the experiments is epidemic spread of node descriptions; when two nodes meet they will exchange all their third party node descriptions on the assumption that all nodes are interested in all node descriptions. Technically, this is achieved in the original technique by putting a common interest in all nodes. If nodes share an interest, node descriptions will be spread epidemically by default since they always will match the interest of other Haggle nodes.

3 Methodology

We experimentally investigate data dissemination with Haggle. In particular we are interested in the trade-off of spreading knowledge about what another node already has in its database and the benefits of having that knowledge. We use different scenarios with and without data forwarding, epidemic spread of node descriptions, and different interest overlap among the nodes.

To perform our experiments we simulate an opportunistic network according to predefined scenarios. A scenario is a set of instructions that configures Haggle, define which application to use, and the connectivity between the nodes. Connectivity is coordinated according to a connectivity trace. We run the experiments on a testbed that is based on virtual computers running Linux. One virtual machine hosts one Haggle node. The virtual computers are running on top of Xen [4], a virtualization software connecting the nodes through an Ethernet bridge on which we can filter traffic.

3.1 Metrics

There are several metrics to investigate when it comes to the efficiency and cost of maintaining what other nodes possess. We will look into three metrics:

3.1.1 Delivery ratio

The main goal with data dissemination in an opportunistic network is to achieve a high delivery ratio of data the nodes are interested in. We measure the delivery ratio for every node as the number of relevant data received over the total number of relevant data stored on any node in the network. The

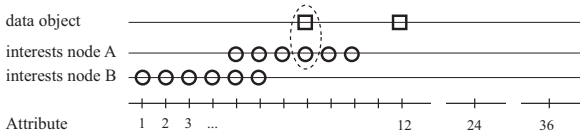


Figure 1: Example distribution of attributes in nodes' interests and a data object.

overall delivery ratio for the network is thus the average delivery ratio over all nodes.

The achievable delivery ratio depends on the connectivity patterns and when data is generated. Nodes that get isolated will not be able to deliver or receive relevant data objects. One can thus only compare the delivery ratio in the same scenario with different types of interest spreading schemes and settings.

3.1.2 Node description overhead

We define node description overhead as the number of node descriptions passed between nodes during the whole experiment. Node description overhead is measured to understand how much additional traffic is created when node descriptions are updated each time the Bloom filter has been altered between node encounters.

3.1.3 Metadata overhead

A receiver sends a reject or accept message for each data object a sending node is trying to push to the receiver. The decision to reject or accept is based on the information in the metadata. Metadata overhead is the number of reject messages that are received over time during an experiment. Rejected attempts to send a data object will happen more often if the knowledge about what other nodes possess is poorly updated, since the sending node will try to send data objects the receiver already has. The number of rejects is a measure of how much additional traffic is created by sending metadata.

3.2 Scenarios

The connectivity pattern used in the evaluation is based on a connectivity trace collected using eight mobile phones that faculty members carried around during a regular working day. The trace is about seven hours long. The characteristics of the trace is described in more detail in the beginning of Section 4.

We use an application on top of Haggle that adds interests to the node and periodically generates data. Interests and metadata attributes are cho-

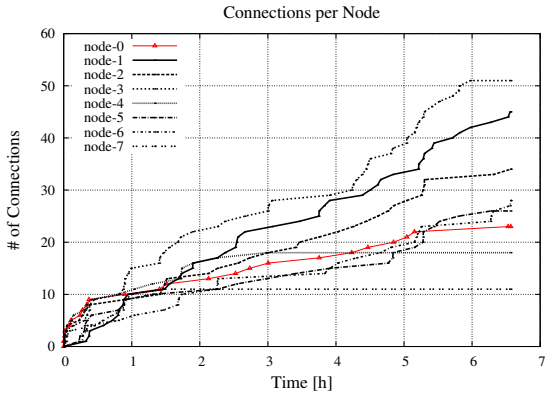


Figure 2: Connections per node over the time of an experiment.

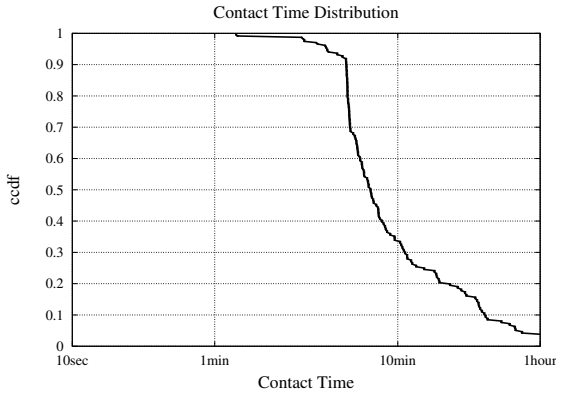


Figure 3: Complementary cumulative distribution of contact times.

sen from a pool of attributes. Each node is given six consecutive attributes as interests, starting with a random attribute (see Figure 1). The interest does not change during the experiment. By varying the attribute pool size we can control the interest overlap between the nodes. In our experiments we choose the size of the attribute pool to be 12, 24, or 36; corresponding to a probability that two nodes share at least one interest of 0.92, 0.46, and 0.31, respectively.

Each node will generate data with two attributes in the metadata, picked randomly from the pool of attributes. In the example shown in Figure 1, the data object has one attribute in common with node A's interests but none with node B's interests. Data objects are generated every two minutes on each node and is generated until the end of the experiment. The last generated data objects will therefore not have the chance of being delivered to all interested nodes.

To compare the effect of updated knowledge about what data objects

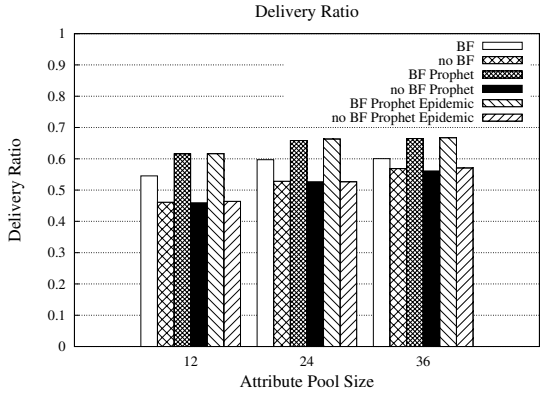


Figure 4: Delivery ratio using different interest overlaps.

nodes possess, we performed the experiments using two modes of Haggle. In one mode Haggle updates the Bloom filter in the node description for each data object that the node receives or creates (see Section 2.1). In the other mode it does not, leaving the Bloom filter empty. This behavior is equivalent to not including a Bloom filter in the node description, with the consequence that nodes only exchange their node descriptions at first encounter but not during later contacts because the node description did not change since then.

All experiments were performed with and without data object forwarding. Forwarding means that a third party node is given data objects that itself is not interested in, but is expected to be likely to deliver to nodes interested in the object. With forwarding we expect a higher delivery ratio. We use the forwarding protocol PRoPHET [14] that calculates delivery probabilities based on contact history.

4 Results and discussion

In this section we describe and discuss the results from our experiments. First, we look at the characteristics of the network trace used for the experiment. Figure 2 shows the number of contacts for each node over the duration of the experiment. The number of contacts varies among the nodes from over 50 contacts within 7 hours for node-3 to only 10 or 18 for node-4 and node-7, respectively, which became isolated after about two hours into the experiment. A small increase of the number of contacts does in general not mean that nodes are isolated, it can also mean that they have long connections ongoing. Figure 3 shows the distribution of contact times. We can see that the median contact time is seven minutes and the longest contacts last for more than one hour. In summary, the office environment offers many contact opportunities to exchange data.

4.1 Delivery ratio

Figure 4 shows the delivery ratio in experiments with different interest overlap. We observe a delivery ratio between 46 percent and 67 percent depending whether Bloom filters are used and forwarding is enabled.

4.1.1 Influence of Bloom filter

An increase in delivery ratio is seen when Bloom filters are used (BF) compared to when they are not used (no BF). The reason for the increase is that nodes know what data the other nodes already have and therefore can send new data instead. This is important as we limit the number of data objects to be exchanged during a new node contact.

4.1.2 Influence of forwarding

Forwarding (BF Prophet) and additional epidemic spread of node descriptions (BF Prophet Epidemic) further increases the delivery ratio. Data objects are forwarded through third party node on the basis that the third party node is likely to be able to deliver to a node. Forwarding decisions are also based on the knowledge about what the end node already possess and can therefore be taken advantage of when data objects are sent through a third party node. To get this knowledge a node relies on receiving node descriptions through third party nodes, either epidemically or because of shared interests. Forwarding itself without Bloom filter (no BF Prophet), interestingly enough, does not increase the delivery ratio. Data objects selected to be forwarded are the best matching to the node's interest. All nodes will therefore be forwarding the best matching data objects to all other nodes and will likely be rejected in the last hop since all nodes are forwarding the same data objects.

4.1.3 Influence of interest overlap

The delivery ratio is slightly higher for experiments with a sparse interest overlap (e.g., attribute pool size 36). With a large attribute pool nodes are less likely to be interested in a data object, as well as other nodes node descriptions. Therefore, if the overlap is sparser the limited number of data objects shared will more likely not be node descriptions. Furthermore, the pool of attributes is larger and data objects still contains two attributes. A node will therefore be interested in a smaller set of data objects. Hence, the delivery ratio is increased. But, compared to an experiment where the interest overlap is large, the improvement Bloom filters give in delivery ratio is smaller. This because the overhead has less influence when a contact is not fully utilized.

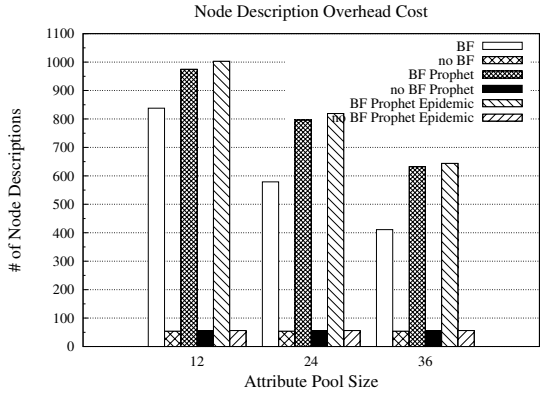


Figure 5: Node descriptions exchanged during an experiment using different interest overlaps.

4.2 Node description overhead

Keeping information about data that other nodes are carrying comes at the cost of exchanged node descriptions. In Figure 5 the amount of node description exchanged during an experiment is shown. In the case without Bloom filters, nodes exchange their node description only once with every peer, resulting in 8*7=56 node descriptions to be exchanged assuming that every node gets in contact with every other node. This number is independent of interest overlap and forwarding strategy.

Including Bloom filters in the node description results in a significant increase of node description exchanges. In the scenario with larger interest overlap (i.e., attribute pool size 12), 838 objects are exchanged without forwarding, and up to 1003 objects with forwarding. As the interest overlap decline (i.e., attribute pool size 24 and 36), we observe a decline in exchanged node descriptions as well. This can be explained by the fact that node descriptions also get exchanged because they match other nodes' interests. Having similar interests results therefore in more matching node descriptions.

If forwarding is used in combination with Bloom filters, there is an increase in node description exchanges due to forwarded third party node descriptions. The increase is more prominent when the interest overlap is smaller. Nodes that have a high probability to share a common interest will exchange node descriptions anyway.

4.3 Total overhead

The total overhead is the node description overhead combined with the amount of rejected data objects. With an increased spread of knowledge about other nodes, the number of rejected data objects is expected to de-

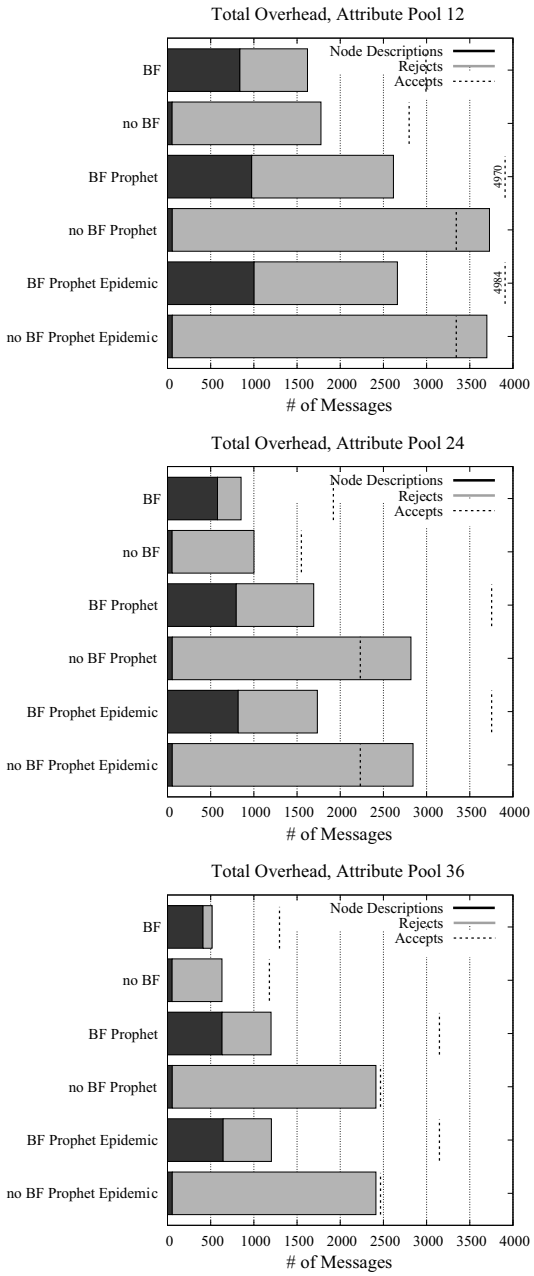
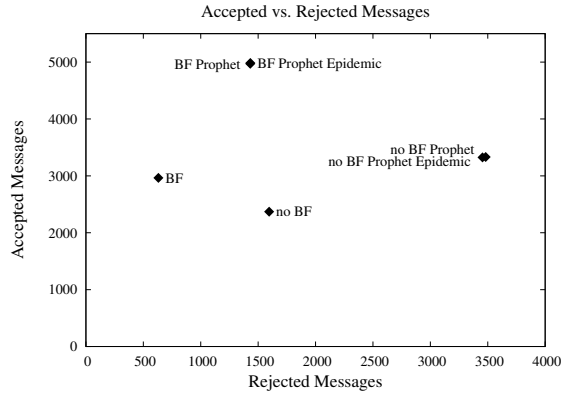
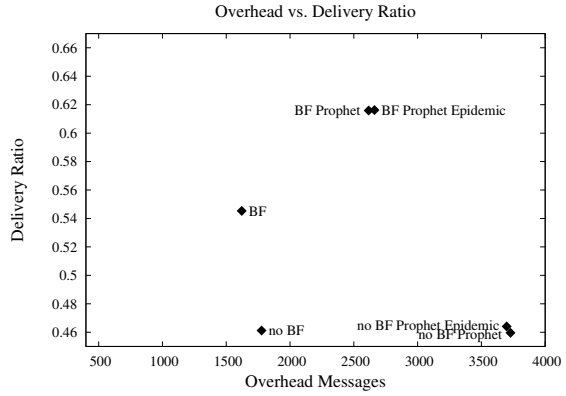


Figure 6: Total overhead (node descriptions and rejects) using different interest overlaps. The dashed line shows the number of accepted packet, also including accepted data object by third party nodes.

crease. The total amount of overhead is depict in Figure 6. These graphs show that the cost in node description overhead is compensated with a reduction in rejected data objects. Bloom filters reduce rejected data objects



(a)



(b)

Figure 7: a) shows the increase of accepted message over rejected messages when forwarding us used. b) Delivery ratio over the cost in overhead. Attribute pool 12.

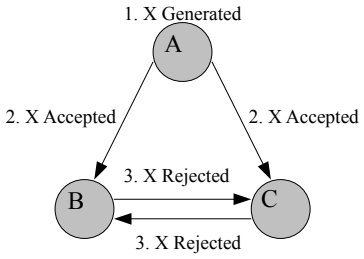
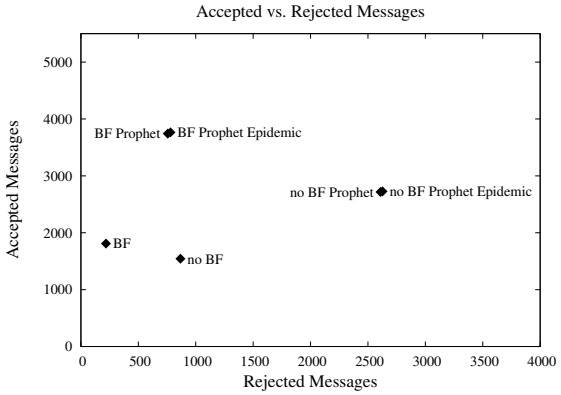
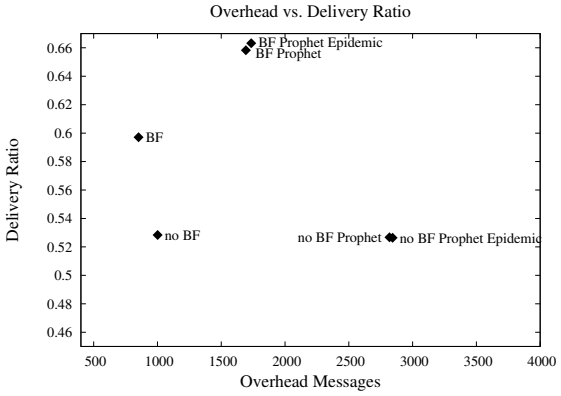


Figure 8: Illustration of a situation where messages are going to be rejected even when Bloom filters are used.

little more than 50 percent, the improvement is more significant when the overlap in interest is less. The amount of rejected data objects decrease



(a)

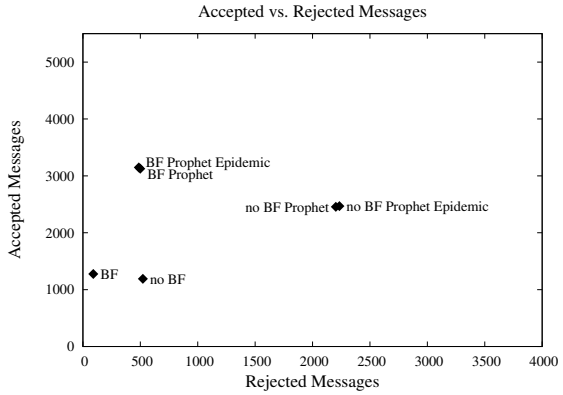


(b)

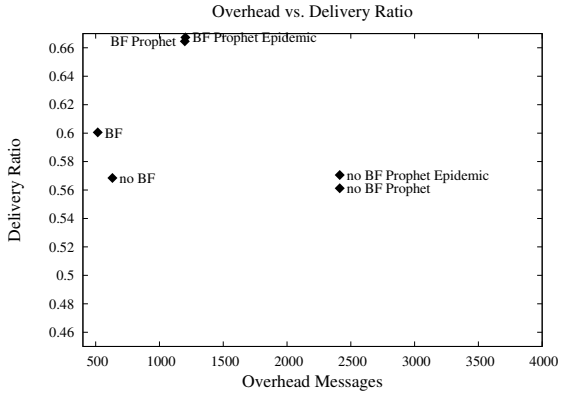
Figure 9: a) shows the increase of accepted message over rejected messages when forwarding us used. b) Delivery ratio over the cost in overhead. Attribute pool 24

with a smaller overlap of interests since fewer data objects will match the receivers interest. However, even when Bloom filters are used there are data objects that are rejected. This is because of situations like the one illustrated in Figure 8. The three nodes are in communication range of each other. Node A generates data object X that both B and C is interested in. The data object is then sent to both B and C. Consequently, B will then make a resolution and match the received data object to C's interest and vice versa. Since neither B nor C have data object X in the respective Bloom filter, both B and C will send the data object to each other and both nodes will reject it.

When forwarding is added it is evident that the total overhead increase. In Figure 7(b) the delivery ratio is compared with the total overhead. It is apparent that the increase in accepted data objects (see Figure 7(a)), that



(a)



(b)

Figure 10: a) shows the increase of accepted message over rejected messages when forwarding us used. b) Delivery ratio over the cost in overhead. Attribute pool 36.

forwarding induce cannot be taken advantage of if Bloom filters are not used. This because the same data objects are sent through different paths to the end node resulting in multiple rejects. We also observe that Bloom filters increase delivery ratio at the same time as it reduces the overhead. Compared to the results without forwarding, it comes with a cost of an increased number of rejected packets. Table 1 shows that the amount of accepted data objects also increase. The additional accepts are the result of accepted data objects in intermediate hops. The same trend is seen in experiments with a sparser overlap of interest, shown in Figure 9 and Figure 10.

Epidemic spread of node descriptions has a negligible effect on the total amount of overhead throughout all interest densities and regardless of Bloom filters are used or not. This is a similar finding as for delivery ra-

Attribute pool size	12		24		36	
Action	Accept	Reject	Accept	Reject	Accept	Reject
BF	2964	631	1809	218	1275	90
BF Prophet	4970	1431	3742	756	3126	499
BF Prophet Epidemic	4984	1432	3765	780	3148	487
no BF	2370	1596	1543	867	1190	522
no BF Prophet	3332	3482	2716	2609	2455	2203
no BF Prophet Epidemic	3325	3454	2728	2627	2468	2234

Table 1: Number of accepted and rejected data objects (exclusive node descriptions).

tio. In a scenario with a high occurrences of connectivity among all nodes, epidemic node description spreading have a small impact on the increase in dissemination. Node descriptions are exchanged when nodes meet and also sent to third party nodes if interests are shared.

5 Related work

The Content Centric Networking (CCN) [11] communication model is driven by consumers of data. Like Haggle's interest (defined in node descriptions) and data objects, CCN have two types of packets, interest and data. Unlike CCN, where packets transmit data in response to an interest packet and the interest packet is then consumed, Haggle is constantly searching the network by local searches on each node in the network.

In CCN interest packets are forwarded toward potential sources. Each hop maintains a table of pending interests. If an interest packet is received that exactly matches an entry in the table, the packet will be discarded since an interest has already been sent. Data sent in response to an interest is always sent back the same route it arrived in. A data packet, on its way back, will be copied to all interfaces that the interests was received from. Therefore, data cannot loop in CCN, but in a highly connected topology interests can loop. A random nonce has been added to the interest packet to detect and prevent duplicated interest packets. In Haggle Bloom filters are used to detect and prevent data objects from being delivered multiple times to a node as well as forwarded through other nodes.

The Delay Tolerant Networking Research Group (DTNRG) [1] is working on a standardization of protocols for networks where continuous end-to-end connectivity cannot be assumed. The proposed architecture [9] use bundling of data and late binding addressing based on end-point identifiers. Haggle

also use late binding but use application layer framing with searching to bind data to nodes.

A number of DTNs have been deployed. DieselNet [6] provides urban internet access through the use on 30 buses carrying 802.11 devices in the urban area of Amherst, MA. Another application is Internet service for remote third-world regions were there is a lack of infrastructure based service, such as KioskNet [10], DakNet [17].

Haggle contains a publish/subscribe API for dissemination of data. Subscribers are temporally and spatially decoupled from the publisher in the publish/subscribe communication paradigm [8]. Haggle is using filters for local demultiplexing of data to applications, and use ranked searches for dissemination. Traditional publish/subscribe systems either disseminate based on exact matching filters [7], or channels of topics [13].

6 Conclusions

We experimentally evaluated the overhead of different interest dissemination strategies in opportunistic networks. Haggle's relation graph of what other nodes wants and what they already have allows for informed decisions what to forward to other nodes in contact. For dense connectivity like in the trace used for the experiments, we find that exchanging information about data stored on the nodes is worth the involved overhead of additional node descriptions. We achieve a significant increase of the delivery ratio and reduce the total overhead. We can also conclude that this information is in particular important when using forwarding in order to keep the number of (unnecessary) copies of the same data object in the network limited. Moreover, forwarding without information about what data other nodes already have does not increase the delivery ratio. We expect the findings be valid also for more sparsely connected networks because additional information, if available, allows to make more informed decisions and improves delivery.

7 Acknowledge

This research is funded by the Haggle project under the EU grant IST-4-027918.

References

[1] Delay Tolerant Networking Research Group, http://www.dtnrg.org.

[2] The Haggle project, http://www.haggleproject.org.

[3] The N4C project, http://www.n4c.eu.

[4] Paul R. Barham, Boris Dragovic, Keir A. Fraser, Steven M. Hand, Timothy L. Harris, Alex C. Ho, Evangelos Kotsovinos, Anil V.S. Madhavapeddy, Rolf Neugebauer, Ian A. Pratt, and Andrew K. Warfield. Xen 2002. Technical Report UCAM-CL-TR-553, University of Cambridge, Computer Laboratory, January 2003.

[5] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[6] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *In Proc. IEEE INFOCOM*, 2006.

[7] Antonio Carzaniga and Alexander L. Wolf. Forwarding in a content-based network. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 163–174, New York, NY, USA, 2003. ACM.

[8] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

[9] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, New York, NY, USA, 2003. ACM.

[10] S. Guo, M. H. Falaki, E. A. Oliver, S. Ur Rahman, A. Seth, M. A. Zaharia, U. Ismail, and S. Keshav. Design and implementation of the kiosknet system.

[11] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *CoNEXT '09: Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12, New York, NY, USA, 2009. ACM.

[12] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. *SIGOPS Oper. Syst. Rev.*, 36(5):96–107, 2002.

[13] Vincent Lenders, Gunnar Karlsson, and Martin May. Wireless ad hoc podcasting. In *Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, June 2007.

[14] Anders Lindgren, Avri Doria, and Olov Schelen. Probabilistic routing in intermittently connected networks. In *SIGMOBILE Mobile Computing and Communication Review*, page 2003, 2004.

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

[16] Erik Nordström, Per Gunningberg, and Christian Rohner. Haggle: a data-centric network architecture for mobile devices. In *MobiHoc S3 '09: Proceedings of the 2009 MobiHoc S3 workshop on MobiHoc S3*, pages 37–40, New York, NY, USA, 2009. ACM.

[17] Alex (Sandy) Pentland, Richard Fletcher, and Amir Hasson. Daknet: Rethinking connectivity in developing nations. *Computer*, 37:78–83, 2004.

[18] J. Scott, P. Hui, J. Crowcroft, and C. Diot. Haggle: A networking architecture designed around mobile users. In *Proc. of the 2006 IFIP Conference on Wireless on Demand Network Systems and Services (IFIP WONS 2006)*, 2006.

Paper D

Fredrik Bjurefors, Per Gunningberg, Christian Rohner, and Sam Tavakoli. *Congestion Avoidance in a Data-Centric Opportunistic Network.* In Proceedings of ACM Workshop on Information-Centric Networking (ICN 2011), Toronto, Canada, August 2011.

Congestion Avoidance in a Data-Centric Opportunistic Network

Fredrik Bjurefors, Per Gunningberg, Christian Rohner, and Sam Tavakoli

Department of Information Technology
Uppsala University

Abstract

In order to achieve data delivery in an opportunistic network, data is replicated when it is transmitted to nodes within communication reach and that are likely to be able to forward it closer to the destination. This replication and the unpredictable contact times due to mobility necessitate buffer management strategies to avoid buffer overflow on nodes. In this paper, we investigate buffer management strategies based on local forwarding statistics and relevance of the data for other nodes. The results obtained on our emulation platform for opportunistic networks show that strategies with a high data refresh rate achieve the most efficient delivery and generate the smallest overhead on our community and mobility scenarios.

1 Introduction

Communication in opportunistic networks is based on sporadic and intermittent contacts between mobile nodes. These contacts are used to exchange data with nodes that are likely to move data closer to the destinations. The lack of end-to-end paths makes data-centric networking with a focus on content rather than on location an attractive communication model for opportunistic networks.

Considerable work has been done on forwarding strategies, deciding what data to replicate and exchange during contacts. Knowledge about movement pattern, social relations in communities and user interests can be used for efficient data dissemination within and between communities. Identification of central nodes with many contacts, called hubs, and nodes that have frequent contacts with two or more communities, called bridge nodes, have been shown to be efficient in data dissemination [5].

Data replication in general leads to "node congestion," i.e., filling up buffers with replicated data. Hubs and bridges in particular are likely to

face the issue of evicting data to leave space for new data. In this paper, we emphasize the importance of efficient congestion avoidance. We investigate into buffer management and data dropping strategies based entirely on local information, and show that strategies with a high data refresh rate achieve the most efficient delivery and generate the smallest overhead in our community and mobility scenarios.

The paper is structured as follows. In Section 2, the data-centric architecture used in this work is described. In Section 3, congestion avoidance strategies in terms of buffer management are discussed. Section 4 describes the methodology and the scenarios. Section 5 describes and discusses the results of our experiments with different buffer management strategies. Related work is reviewed in Section 6. Last, we conclude our findings in Section 7.

2 Haggle

We evaluate congestion avoidance using the Haggle [10, 11] test-bed. Haggle is a data-centric network architecture, designed for mobile nodes, based on a publish/subscribe model. A Haggle node can opportunistically take advantage of any communication technology of mobile devices, such as Wi-Fi and Bluetooth. A device runs an instance of Haggle to which applications connect. The instance becomes a node in an opportunistic Haggle network.

2.1 Data Objects and Node Descriptions

A Haggle node has a buffer management system where it stores *data objects*. Data objects consist of *data* with associated *metadata.* Data could be anything from a couple of bytes to a large video clip of many Mbytes. Metadata uses *attributes* that describes the actual data. Nodes also express their *interests* in data in the form of these attributes. A *node description* consists of metadata describing what the node is interested in and contains other node-specific information. An application on a node generates new data objects as well as consumes data objects that it is interested in.

2.2 Forwarding

When two nodes are getting in contact, they first exchange their node descriptions. The interest attributes of the two nodes are matched against metadata attributes of the data objects stored on the nodes. If there is a match of interests, data is exchanged. In this manner, data objects are eventually spread among all nodes with common interests.

In addition, nodes exchange data that the other node is not interested in by itself, thus acting as a relay node. The nodes calculate which other nodes in the network the the neighboring node is more likely to meet in the future.

If the other node is more likely to be able to forward the data towards the destination than the node itself, the node pushes the data objects to the neighboring node.

The relayed data objects will occupy space in the *relay buffer*. The objects will stay there and will be replicated and exchanged when meeting other nodes with interest in these data objects. Data objects are replicated when they are forwarded, i.e., forwarding does not mean that a data object is deleted from the sending node's data store. Eventually all buffers will be filled up and there must be a strategy to evict data objects to create space for new data objects.

3 Congestion

In an opportunistic network, data dissemination is done through data replication. Multiple copies of data increase the chance for data to be delivered, but it also congests the network. The remedy in place to avoid congestion is to limit the number of copies of data in the network.

A node receiving many data objects during a short period of time will drain its storage resources. When storage is depleted, a node will not be able to receive or relay objects, which means that the whole network forwarding efficiency will be hampered by these blocked nodes. To get the delivery system to work properly under short overload periods congestion avoidance methods are needed, and when the buffers are filled up, there is a need for an algorithm for dropping objects.

Traditional congestion avoidance systems, like TCP use feedback information from the end destination to the source (e.g., ACKs) and the sender transmission rate is throttled to the capacity of the end-to-end path. TCP is complemented with an algorithm to selectively drop segments in routers, e.g., RED [4]. Opportunistic networks lack an end-to-end path and the whole data object is forwarded to the destination on a store-carry-forward basis without an ACK back to the sender. Therefore, traditional congestion avoidance techniques cannot be used.

The objective with congestion avoidance in this type of opportunistic system is to take decisions on how much data to replicate, and what to drop from the relay buffer and when, with minimum impact on the overall delivery ratio. Since there is no feedback information, the decision to drop must be based on local information that a node can gather when it is in contact with other nodes.

The information that is available to a node is forwarding statistics and node descriptions. Forwarding statistics consist of the number of times a node has replicated a data object. Also, the forwarding probability of the node that the data object is sent to is collected. A highly replicated object is a candidate for dropping since there are already many of them in the

network. The node descriptions received from other nodes give information about the subscribers' interests. An object in which there is little interest is also a candidate for dropping.

3.1 Buffer Management

Haggle selects the data objects to be dropped from the relay buffer based on age. Data objects older than a certain threshold will be dropped. Aging is performed periodically. The chosen age threshold and periodicity, in relation to the rate of incoming data defines the occupation in the relay buffer and may fluctuate significantly over time.

In this work, we take buffer size in consideration and introduce a limit on the relay buffer. If the relay buffer on a node gets full, data objects will be evicted from the buffer. Resource management in a node considers all data objects that are "not of interest" and removes them. The size of object is not taken into account in this first evaluation but it could also be used as a factor. A large object releases more buffer space but a large object takes more time to spread due to limited bandwidth and contact times.

3.1.1 Evaluated dropping strategies

We evaluate data object dropping strategies based on interests and on the degree of replication.

LI – Least Interested: Drop the data object that the least number of neighbors are interested in. This strategy has two side effects: (a) it reduces the diversity of content since the object will eventually be extinct if no one is interested in it for a long time, (b) but it increases the overall delivery ratio of other objects since the overall interest matching is increasing.

MI – Most Interested: Drop the data objects that most neighbors are interested in. As opposed to the previous strategy, this will maintain data object diversity in the network, but may have a negative effect on the delivery ratio. The strategy will reduce the number of copies of data objects that are most looked for. On the other hand, a large number of copies of the same data were shown not to increase the delivery ratio [3].

The following strategies drop data objects based on local forwarding history, i.e., replication of data.

Max – Max Copies: Drop data objects after a max number of copies have been made at the local node. Data objects are removed after they have been copied *Max* number of times. This makes the relaying node reject relayed data objects until space has been cleared in the buffer, no preemption is done.

MF – Most Forwarded: Drop the data object with the highest number of replications made at the local node.

LF – Least Forwarded: Drop the data object with the lowest number of replications made at the local node.

Random: Drop randomly chosen data objects from the relay buffer. We compare the other five strategies against this reference case.

Infinite Buffer: Reference best case. It is the overall best case, represented by nodes with an infinite relay buffer. No object needs to be dropped

No Buffer: Reference worst case. The worst case is represented by a node strategy, which only accepts data that the node itself is interested in, i.e., there is no relay buffer.

3.2 Flow Control

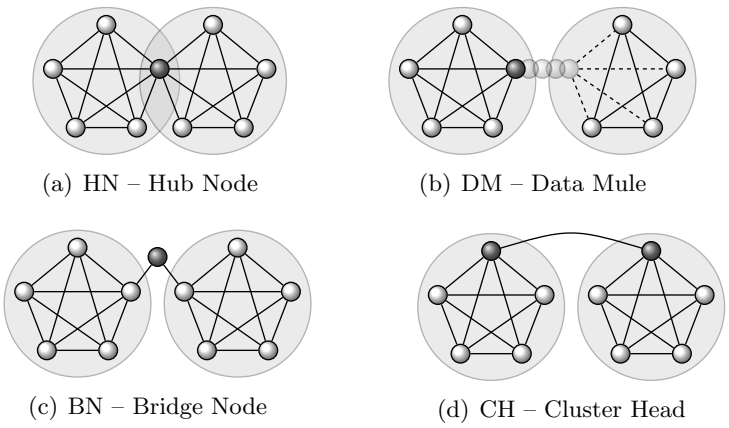
Flow control prevents a sender from overwhelming a receiver with data by having the receiver signal its ability to handle incoming data. Haggle uses two combined mechanisms to control the flow of objects between nodes. The first is to limit the number of data objects exchanged. A receiving node must have at least that buffer space. The second is to limit the exchange of data objects to only the highest ranked with respect to interest. In general, a fixed limit will decrease the overall dissemination of data objects. It will have the same effect on dissemination as a network with fewer contact opportunities. If the network is sparsely connected, only the highest ranked data objects are likely to be exchanged.

The maximum number of data objects possible to receive and a ranking threshold can be advertised in the node description. However, node descriptions are only exchanged during the initial phase of a contact and there is no dynamic feedback during the actual transfer that can throttle the sender.

By limiting the number of shared data objects, we can only reduce the risk of congestion. It is not possible to avoid or recover from congestion. For that purpose, dropping strategies are needed.

4 Evaluation

The performance evaluation is performed on a test-bed based on virtual computers running Linux. One virtual machine hosts one Haggle node. The virtual computers are running on top of Xen [2], a virtualization software that connects the nodes through an Ethernet bridge where we can filter traffic between the nodes. The mobility, community graphs and contact times are emulated and are filtered according to scenarios and connectivity traces. The user data production and consumption for each node is identical for each scenario in order to be able to compare strategies.



(a) HN – Hub Node (b) DM – Data Mule

(c) BN – Bridge Node (d) CH – Cluster Head

Figure 1: In each scenario five communities are connected in a chain. Depicted are the four different hub and bridge nodes (shown in dark gray) used in the scenarios. Lines between nodes represent intermittent connects.

4.1 Scenarios

Four scenarios are used in the evaluation. In each scenario, five communities are connected in a chain, unless mentioned otherwise. A community is a group of nodes that have many contact opportunities between each other [5]. In the scenarios we vary the way neighboring communities are connected to each other, as depicted in Figure 1, and we study the impact of different congestion strategies.

For all scenarios, we use a community size of five nodes where each node have intermittent contact with all other nodes in the community. Contacts occur according to a Markov model with a mean contact time of 60 seconds and an inter-contact time of 240 seconds. The total length of an experiment is one hour.

The following hub or bridge node(s) scenarios are used:

HN – Hub Node: Two communities are connected by a hub node. All data objects exchanged between two communities have to go through this hub.

DM – Data Mule: A "data mule" node belongs to two communities but is only present in one community at a time. A data mule receives all packets destined to a neighboring community. During a stay in one community, the mule will receive packets destined to nodes in a neighboring community. The buffer may become full before it leaves for the other community, therefore, data must be dropped. A "data mule" node stays in a community for 600 seconds before it moves over to the other community.

BN – Bridge Node: In this scenario communities are connected by a bridge node between each community pair. Three nodes are likely to be congested, the bridge node and the two nodes connecting to the bridge. In

this scenario we interconnect four communities instead of five to keep the number of nodes comparable with the other scenarios.

CH – Cluster Head: One node from each community is connected to a node in the neighboring community, forming a link between them. Congestion occurs by the fact that a node within a community that wants to send data to a node in any other community must go through the cluster head node of the community, thus a bottleneck occurs.

4.2 Scenario parameters

Each node is set to be interested in five attributes. They are chosen from a pool of $N = 100$ attributes. The attributes are assigned to the nodes according to a Zipf law distribution of $\alpha = 0.368$ [8]. Nodes are set to have 200 data objects stored in their buffer, marked as received from the node's application. Each data object is also assigned five attributes from the same pool of 100 attributes. These attributes are chosen from a uniform distribution.

In our experiments, we vary the size of the relay buffer. During each contact the nodes exchange up to 20 data objects matching their own interest and up to 20 data objects that are relayed. When the relay buffer reaches 80% utilization, data objects will be removed according to the dropping strategy. We limited the buffer to 80% in order to let the chain of events in the Haggle kernel to take place without risking a buffer overflow. When a relayed data object also is of interest for the relaying node the data objects will not take up space in the relay buffer. Instead, it will be put in another buffer that is considered infinite. Aging is not used in our experiments.

5 Results and Discussion

In this section, we present performance results from our experiments. We report on the metrics such as delivery ratio, dissemination speed, and forwarding overhead.

5.1 Delivery ratio

We measure the delivery ratio per node. With this, we mean how many of the nodes in the network that have an interest in the object, will actually get a copy within a time frame. The *delivery ratio* for the whole network is thus the delivery ratio over all nodes and all objects. Figure 2 shows the delivery ratio in all four scenarios and the dropping strategies. As can be obsverd, the Hub Node scenario is performing the best for all dropping strategies. This shows the importance of good and frequent connectivity of the hub node within and between communities. The other scenarios perform significantly worse with respect to delivery ratio with the Cluster Head scenario slightly

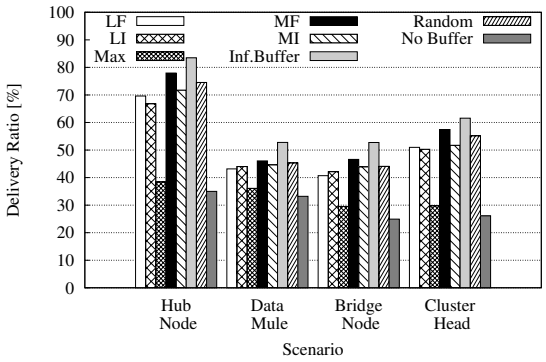


Figure 2: Delivery ratio using six object-dropping strategies in four scenarios. The dropping strategies are: Least Forwarded (LF), Least Interested (LI), Max Copies made of data, Most Forwarded (MF), Most Interested (MI), and Random. The cases, Inf. Buffer do not drop any packets and No Buffer which represents the case when nodes do not relay objects.

better than Data Mule and Bridge Node scenarios as it provides a more direct connection between the communities.

The relative performance between the strategies are similar over the different scenarios. The strategies Most Forwarded (MF) and Random are performing closest to the reference strategy Infinite Buffer. The MF strategy (which drops the data object with the highest number of replications made at the local node) performs the best. An explanation for this is that it favors diversity so that the not so popular objects get a chance to be spread to the interested nodes. The Least Forwarded (LF) strategy that drops the data object with the lowest number of replications has the opposite effect by favoring popular objects so that they are quickly spread in the network. On the other hand, it creates many copies of the same objects, which takes space in buffers and is not efficient from the whole network-spreading point of view. Hence, the delivery ratio, which is calculated over all objects and all nodes is hampered. The performance of MI, to drop the objects with most interest, follow the same discussion as for the MF—the objects with the most interest in are likely to be the most copied as well. That Random dropping is doing so well can again be argued with the same rationale—it allows less popular objects to get space in the buffers. The dropping strategy Max Copies is performing close the worst case strategy No Buffer. This is an indication that it pays to really use a dropping strategy that often exchange data in the buffer.

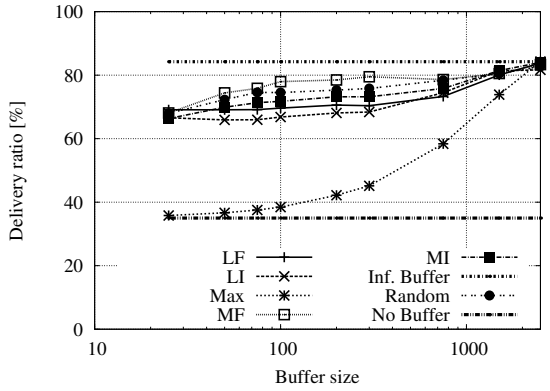


Figure 3: Delivery ratio using different dropping strategies, in the Hub Node scenario.

5.1.1 Hub Node scenario

The Hub Node scenario has the highest delivery ratio because of the well-connected hub node that resides within two communities. We will take a closer look at Hub Node with respect to its impact on buffer size and dissemination speed, as well as forwarding overhead. The other scenarios show similar results, but with a lower delivery ratio.

Figure 3 shows the delivery ratio for the different dropping strategies with varying buffer sizes. We observe a clear difference between the Max Copies and the other strategies. While the other strategies achieve a fair delivery ratio already with small buffer size, the Max Copies strategy scales linearly with the buffer size between the two reference strategies No Buffer and Infinite Buffer. For the other strategies, it is observed that they scale with buffer size up to 100 data objects from which point a larger buffer size does not improve the delivery ratio until it approaches the maximum needed buffer occupation.

For this particular contact scenario, our interpretation is that a buffer of 100 data objects is enough if there is a dropping strategy. In other words, above this buffer size the dropping strategy is more significant than an increase in buffer size. Furthermore, the Max Copies strategy is not suited for scenarios where nodes only have a few possible neighbors that may share the interests. What happens in this strategy is that objects in the relay buffer will not be discarded until they have been replicated and sent to other nodes. The consequence is that objects that few nodes are interested in will occupy buffer space, blocking new objects that may have a better chance to be spread.

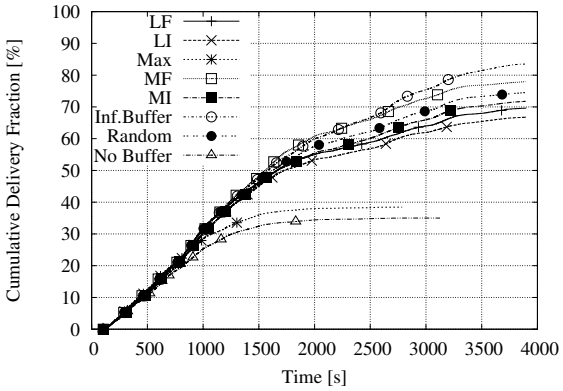


Figure 4: Cumulative delivery fraction over time in the Hub Node scenario using a buffer size of 100 data objects.

	Inf.Buffer					MF buffer=25					MF buffer=100				
Community No.	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Hub Node	93	51	39	26	22	100	46	34	20	17	86	50	41	31	31
Data Mule	90	46	17	7	2	95	40	12	4	0	94	43	16	5	2
Bridge Node	100	23	6	2		100	10	2	1		100	13	4	0	
Cluster Head	100	36	18	7	8	100	29	18	12	9	100	27	16	12	12

Table 1: Delivery ratio per source community for nodes in community one. Community number five represents the community furthest away from community one.

5.2 Dissemination speed

Table 1 shows the delivery ratio from community 1-5 respectively, to nodes in community 1. It shows how many data objects that nodes in community 1 will get from the other communities given the nodes' interests in the objects. Since objects are forwarded from community to community via relay nodes, the delivery rates are implicit indicators of the dissemination speed. As can be seen in the table, the nodes of community 1 in the Hub Node scenario receive 93% of all objects of interest in community 1 and 22% of the objects of interest in community 5. The Infinite Buffer strategy provides the fastest dissemination as discussed earlier, but even for a small buffer size (25), the dissemination speed is not too far from the best case. When comparing the performance of interconnection alternatives, the Hub Node is doing better than the others in terms of the delivery ratio, i.e., it has the fastest dissemination speed.

Figure 4 shows the cumulative delivery ratio for all dropping strategies with respect to time. The MF strategy comes very close to the Infinite Buffer reference case for all times. Thus, it is the best dropping strategy when optimizing for dissemination speed or shortest average delay.

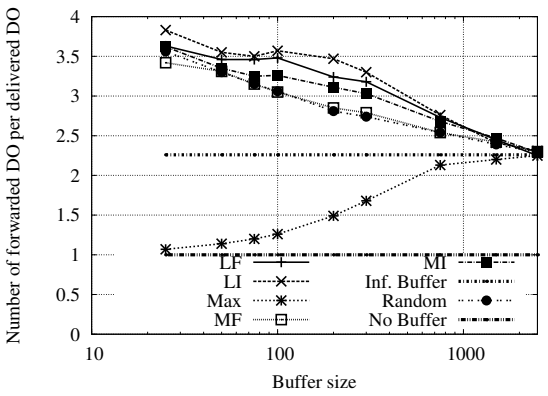


Figure 5: Overhead in terms of average number of forwarded data objects per successfully delivered data object, in the Hub Node scenario.

If a No Buffer strategy is used, data objects will only be exchanged when nodes are in contact and share interests. There is no relaying of objects. As can be seen in the figure, objects will eventually stop being exchanged when buffers for shared interests are exhausted. Even with a small relay buffer and a dropping strategy, data objects will get through to the interested nodes, but at the cost of forwarding overhead.

5.3 Overhead

With overhead, in this paper, we mean the number of times a data object has to be forwarded per delivery to an interested node. The system creates copies of objects that are distributed to the interested nodes. The more copies of an object that are circulating the more likely it is that at least one of them will reach a node interested in it. Likewise, the average delay to a node will be shortened with more copies if there is enough capacity in the system to accommodate all copies. Therefore, it is interesting to understand how many times forwarding takes place per efficient delivery of an object, and which dropping strategy is the most efficient.

In contrast to flooding and epidemic spreading where nodes exchange copies every time they meet, an interest-based system only exchanges copies when there are matching interests or when there is a relaying opportunity. The nodes will store the copies until the right time and not resend them as soon as possible, which is the case in flooding and epidemic spreading. This conserves both bandwidth and energy but requires larger buffers.

Figure 5 depicts this overhead per dropping strategy and buffer size. The most efficient strategy and the reference case is that of Infinite Buffers. In this strategy, a node always has buffer space to receive an object and can store everything it receives. There is no resending of objects due to

dropping of objects. Here we can see that a data object is copied on average 2.2 times to reach nodes that are interested in it. When studying the graphs for dropping strategies, we observe that they use between 3.8 and 2.2 copies depending on strategy and buffer size. To achieve a high delivery ratio, a data object has to be resent after it has been dropped from the buffer. The difference to the Infinite Buffers case of 2.2 is due to the dropping strategy, when the dropped objects have to be copied again. With a smaller buffer size, more objects have to be dropped and replaced than for bigger buffer sizes. The No Buffer strategy does not have any relay buffer. The object is copied directly to the interested party. MF has the lowest overhead compared to other dropping strategies with a high delivery ratio. MF is also the strategy with the highest delivery ratio. To summarize, storage space is traded against forwarding overhead and with less objects dropped, less objects have to be resent.

6 Related Work

Related congestion management work has been done in the area of host-centric Delay Tolerant Networking. In DTN, a node guarantees to forward a bundle (message) to a new custodian or deliver it to the destination. When a bundle is accepted, it cannot be dropped since there is no copy left at the sender. Seligman et al. propose to handle storage congestion at custody nodes by migrating bundles to alternative custodians [12]. Instead of migrating already stored bundles, Zhang et al. [13] propose a congestion management scheme that takes active decisions on whether to accept a custody bundle to avoid congestion. In our Haggle system, there is no commitment of guaranteed delivery, data objects are replicated in each forwarding step.

Balasubramanian et al. [1] and Krifa et al. [7, 6] have studied routing in a DTN as an allocation problem to avoid congestion and possible dropping of a bundle. Joint scheduling is done by using global information. Global information is either propagated through the network [1] or acquired by statistical learning [7, 6]. Both proposals can optimize dissemination to a specific metric, e.g., average delay or delivery probability.

Lindgren et al. [9] evaluate queuing policies based on local forwarding counters and transient forwarding probabilities, using PRoPHET. They show that probabilities can be used in buffer management to increase the delivery ratio. With weights reflecting the number of copies they also show how PRoPHET can be made more energy efficient per packet delivery. An evaluation is performed in a host-centric and intermittently connected network.

7 Conclusions

We have evaluated dropping strategies at congested nodes for data-centric opportunistic networks in different community scenarios. In such networks congestion avoidance, buffer handling, and choice of data object to drop must be based on local information. Nodes do not have complete information of the interests of other nodes in the network since Node Descriptions will only spread between nodes that share interests.

From the experimental evaluation, we conclude that the MF strategy (which drops the data object with the highest number of replications made at the local node) performs overall best with respect to delivery ratio, delay, and overhead at a congested node, i.e., when there is not enough buffer space for relaying objects. Furthermore, using local replication information when dropping gives a higher delivery rate compared to using local interest information. The strategy to randomly select a data object to drop comes surprisingly close to the best cases in our measurements. This shows that already with a small relay buffer, performance gains can be achieved with a dropping strategy where data objects often are exchanged.

8 Acknowledgments

This research was funded by the ResumeNet project under the EU grant FP7-224619 and the Haggle project under the EU grant IST-4-027918.

References

[1] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. *SIGCOMM Comput. Commun. Rev.*, 37(4):373–384, 2007.

[2] P. R. Barham, B. Dragovic, K. A. Fraser, S. M. Hand, T. L. Harris, A. C. Ho, E. Kotsovinos, A. V. Madhavapeddy, R. Neugebauer, I. A. Pratt, and A. K. Warfield. Xen 2002. Technical Report UCAM-CL-TR-553, University of Cambridge, Computer Laboratory, Jan. 2003.

[3] F. Bjurefors, P. Gunningberg, E. Nordström, and C. Rohner. Interest dissemination in a searchable data-centric opportunistic network. In *European Wireless Conference, 2010*, pages 889–895, 2010.

[4] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1:397–413, August 1993.

[5] P. Hui and J. Crowcroft. Bubble rap: Forwarding in small world DTNs in ever decreasing circles, 2007.

[6] A. Krifa, C. Barakat, and T. Spyropoulos. Optimal Buffer Management Policies for Delay Tolerant Networks. In *SECON*, page 10, 2008.

[7] A. Krifa, C. Barakat, and T. Spyropoulos. An optimal joint scheduling and drop policy for delay tolerant networks. In *WoWMoM*, pages 1–6. IEEE Computer Society, 2008.

[8] V. Lenders, G. Karlsson, and M. May. Wireless ad hoc podcasting. In *SECON*, June 2007.

[9] A. Lindgren and K. Phanse. Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks. In *COMSWARE, 2006*, pages 1–10, 2006.

[10] E. Nordström, P. Gunningberg, and C. Rohner. Haggle: a data-centric network architecture for mobile devices. In *MobiHoc S3 '09*, pages 37–40. ACM, 2009.

[11] J. Scott, P. Hui, J. Crowcroft, and C. Diot. Haggle: A networking architecture designed around mobile users. In *IFIP WONS 2006*, 2006.

[12] M. Seligman, K. Fall, and P. Mundur. Alternative custodians for congestion control in delay tolerant networks. In *CHANTS '06*, pages 229–236. ACM, 2006.

[13] G. Zhang, J. Wang, and Y. Liu. Congestion management in delay tolerant networks. In *WICON '08*, pages 1–9. ICST, 2008.

Paper E

Christian Rohner, Fredrik Bjurefors, Per Gunningberg, Liam McNamara, and Erik Nordström. *Making the Most of Your Contacts: Transfer Ordering in Data-Centric Opportunistic Networks.* In Proceedings of the Third International Workshop on Mobile Opportunistic Networking (MobiOpp 2012), Zurich, Switzerland, March 2012.

Making the Most of Your Contacts: Transfer Ordering in Data-Centric Opportunistic Networks

Christian Rohner[1], Fredrik Bjurefors[1], Per Gunningberg[1], Liam McNamara[1], and Erik Nordström[2]

[1]Department of Information Technology , Uppsala University
[2]Princeton University, USA

Abstract

Opportunistic networks use unpredictable and time-limited contacts to disseminate data. Therefore, it is important that protocols transfer useful data when contacts do occur. Specifically, in a data-centric network, nodes benefit from receiving data *relevant* to their interests. To this end, we study five strategies to *select and order* the data to be exchanged during a limited contact, and measure their ability to promptly and efficiently deliver highly relevant data.

Our trace-driven experiments on an emulation testbed suggest that nodes benefit in the short-term from ordering data transfers to satisfy local interests. However, this can lead to suboptimal long-term system performance. Restricting sharing based on matching nodes' interests can lead to segregation of the network, and limit useful dissemination of data. A non-local understanding of other nodes' interests is necessary to effectively move data across the network. If ordering of transfers for data relevance is not explicitly considered performance is comparable to random, which limits the delivery of individually relevant data.

1 Introduction

In opportunistic networks, mobile nodes exchange data when they are within wireless communication range. Data items are disseminated via neighbours to enable them to reach interested parties, with nodes using the *store-carry-forward* principle to collaboratively establish a data relaying service.

Due to the unpredictability of node contact duration it is not known how many successful item transfers will be performed during a contact. Therefore, a data dissemination protocol must judiciously decide *what* data to exchange and in *which order*, to make the best use of a time-limited contact and to achieve the best "value" for the network at minimal cost.

Transferring an item of data represents a selection to spread that particular item instead of another. The receiver then gains the utility from the transferred items and can also spread it to other nodes in the network. When choosing a dissemination system, a key consideration is balancing local benefit to the receiver and potential global benefit for the entire system.

For the rest of the paper, we shall use the following terminology to refer to two aspects of opportunistic data dissemination. Deciding whether an item should be transferred to a given host is a *selection* decision, deciding which order the selected items should be sent in is called an *ordering* decision.

Numerous selection algorithms have been proposed to efficiently choose nodes that will carry data to other nodes in the network [10, 12, 18, 21, 26]. These decisions are often made based on node properties, for example, contact history or role in community structures. Deciding which items to drop from a finite data item buffer has been extensively studied, with strategies based on FIFO, LIFO, TTL, or age [19, 25] of items. By design, however, these strategies see each delivered data item as contributing the same value to the network, and therefore strive to make a similar effort in delivering each item without worrying about ordering. In reality, however, an item's contributed value to different nodes will vary depending on some measure of *relevance*.

In this paper, we claim that all data items are not created equal, and that dissemination systems can benefit from reflecting this reality. One mechanism for a host to determine the relevance of a data item is based on the match between its interests and the item metadata. A node could then decide which items to select for transfer based on the perceived benefit to the recipient or the network as a whole. As the amount of selected data that can be transferred between two nodes is limited by the duration of their contact, the order of the data transferred plays an important role for the performance of the dissemination.

With these observations in mind, we study five data-centric ordering strategies and their ability to deliver relevant data to nodes. By matching user interests against content metadata, recipients assign each data item a relevance score. We then judge a dissemination strategy mainly by its ability to deliver the higher scored data before the lower scored in an efficient manner.

We evaluate each dissemination strategy using the Haggle network architecture [23] running on a trace-driven emulation testbed. Our main results are not specific to Haggle; we simply take advantage of the functionality offered by the platform. We use three different contact traces to study the behaviour of each strategy under varying environments, in terms of connectivity and community structure. Our results show that data transferring based on the relevance leads to a more efficient dissemination of data.

The rest of the paper is organised as follows: Section 2 considers related work, Section 3 presents our formulation of relevance-driven data dissemination. Our experimental details are specified in Section 4 followed by the results in Section 5. Finally, our conclusions are discussed in Section 6.

2 Related Work

Nodes cooperating to provide a data dissemination system need to follow mutually beneficial policies in order to achieve their common aims, such as message ferrying, item replication and searching. Opportunistic sharing actions can be divided into four main groups: i) choosing who to *peer* with; ii) *selecting* files to share with peers; iii) *ordering* the transfer of those files; iv) *dropping* files when buffer space runs out. Nodes can follow a wide variety of policies for each of these actions aiming to maximise different properties of the dissemination system.

The problem of choosing who to peer with is fundamental in opportunistic networks, especially in dense or high churn networks. Initiating communication over extremely short-lived connections may prove fruitless, while very long-lived ones need not be prioritised. RAPID is a protocol that considers DTN routing as a resource allocation problem, translating a selected routing metric into per-packet utilities that determine how packets should be selected for replication in the system [2]. Through the learning of node contact patterns they show it is possible to minimise the chosen routing metric. This work allows for specifying the policy for all actions listed at the beginning of this section, though it does not define how other metrics will be affected.

Numerous algorithms have been proposed that select items to forward based on context or social knowledge to enhance the dissemination [5, 6, 10, 24]. However, the goal of such protocols is to compute good "data mules", and are otherwise agnostic to the relevance of the data forwarded.

Forwarding decisions and buffer management in opportunistic networks have been extensively studied. It has been shown that buffer management strategies, e.g., FIFO, LIFO, "most forwarded" or "oldest" can have a large impact on the delay and overhead in a DTN [19]. However, these strategies optimise for high delivery ratio or low delay, and have at most an *incidental* effect on the relevance of the data delivered. Krifa et al. propose a distributed buffer dropping algorithm (the aspect we do not tackle) that uses the theory of encounter-based message dissemination and statistical learning to approximate an optimal approach with global knowledge, improving average delivery rate and delivery delay [14].

A number of data-centric dissemination systems have previously been proposed that use "content channels" as their underlying dissemination mechanism [8, 15, 16]. Data items are classified into channels that users subscribe to and then synchronise when in contact. Content channels provide a coarse definition of relevance, and there is no intentional ordering of data within channels or when synchronising. We use a more specific notion of relevance based upon the metadata of files being shared. Moghadam et al. [20] proposed an interest-aware content distribution protocol with a more fine grained definition of relevance, similar to our relevance score only selecting whether to forward or not, and not how to order data.

Global knowledge of other nodes naturally helps achieve a more efficient dissemination [12], but perfect global knowledge is impractical to achieve. However,

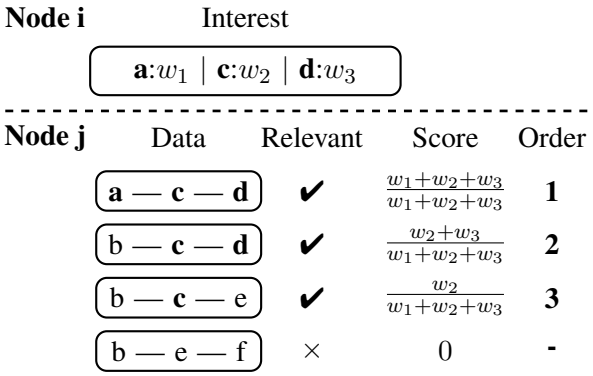


Figure 1: Data ordering according to a node's interests. The recipient i has weighted interests causing the depicted ordering of j's data items.

strategies that learn as much as possible about other nodes in the system may benefit in the long run, as we show in Section 5.

3 Relevance-Driven Data Dissemination

We now introduce the basic concept of relevance-driven data dissemination, which uses *relevance scores* to decide whether to select items for transfer and which order to send them. The process delivers data items to nodes based on their interests rather than their network identifiers (like traditional networking).

To compute a specific node's relevance score for a data item, we assume that each data item's metadata is in the form of a set of attributes $\mathcal{D}$. The use of metadata is widely adopted, for example in music files' *ID3* tags or image files' *Exif* standard. Likewise, each node possesses a set of interests $\mathcal{I}$ that correspond to data attributes. Interests are forwarded to other nodes through direct contacts or through relaying them via third parties. A node's interests are weighted to emphasise particular interests of a node.

We classify data items as *relevant* to a node if the intersection $\mathcal{R} = \mathcal{D} \cap \mathcal{I}$ is non-empty. The strength of a node's interest in a data item is based on the matching interests $a \in \mathcal{R}$ and their respective weights w_a, expressed as a score:

$$score = \frac{\sum_{a \in \mathcal{R}} w_a}{\sum_{a \in \mathcal{I}} w_a} \tag{1}$$

The relevance score can be used by a dissemination system to determine the ordering in which items to be forwarded should be transferred. Figure 1 shows a transfer schedule from node j to node i, based on i's relevance score of the data items carried by j. A high relevance score implies that j should transfer the item early in a contact to ensure its successful delivery. Note that this relevance-driven ordering does not determine which encountered nodes are good "data mules", this is the task of a forwarding algorithm.

Although the above example represents a relatively straightforward approach to ordering that locally optimises for the nodes involved in the exchange, other more collaborative approaches, such as exchanging items also relevant to other nodes, may prove more beneficial for the network as a whole and in the long run. Ensuring a data item is maximally relevant to a receiver is an obvious greedy approach for increasing local interest satisfaction. However, in many cases, when the global interest in a data item is strong—although the local interest is weak—it may be preferable to prioritise that item's dissemination over one that maximises the local interest satisfaction. To study these various trade-offs in data dissemination, we define the following strategies for exchanging data items during a node contact:

ScoreLocal — Only locally relevant data items are exchanged, ordered by their relevance scores relative to the receiver of a data transfer, with the most relevant data item first.

ScoreGlobal — Only globally relevant data items are exchanged, ordered by their aggregate relevance scores relative to all nodes known at the time the data transfer occurs. The purpose of this strategy is to promote the dissemination of globally relevant items over locally relevant ones.

RandomLocal — Only locally relevant data items are exchanged, but they are ordered randomly. This is a baseline for evaluating the value of using the relevance score to order data items.

Random — Any buffered data items are exchanged in random order, irrespective of relevance to the nodes involved in the data exchange. This strategy provides a baseline for evaluating the value of using the relevance score to both *select* and *order* data items.

LiFoLocal — Only locally relevant data items are exchanged, ordered based on buffer time with the "freshest" item first. The purpose of this strategy is to promote fresh data, a property that has been shown to be efficient in buffer management in sparsely connected networks [4].

We shall refer to ScoreLocal, LiFoLocal and RandomLocal as "local strategies" as they select items for transfer based only upon the recipient's interests.

4 Experimental Methodology and Setup

In this section we describe the experimental methodology and setup we use to compare dissemination strategies. The strategies are implemented in Haggle [1] and we evaluate them through a set of experiments executed on a trace-driven emulation testbed, where contact traces decide the connectivity between Virtual Machine (VM) nodes [3]. We use three traces with different properties to minimise the bias on our results due to the configuration of a particular trace (Section 4.1), and

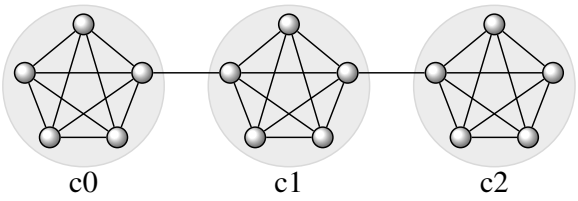


Figure 2: Clusters in Line topology, with three K_5 graphs connected by bridge links. Each edge is an intermittently connected Markov link.

understand transfer ordering across a range of network densities and churn. We distribute interests and data according to a well-known model of data and interests on the web (Section 4.2). A relevance metric from the area of information retrieval allows us to evaluate the relevance of the data delivered to nodes in our experiments (Section 4.4).

4.1 Contact Traces

We use three contact traces, generated from topology/mobility models or real measurements, which each aim to subject our dissemination strategies to different network topologies:

Clusters in Line — This is a synthetically generated trace that configures nodes in clusters with the goal of studying dissemination in a segmented network, as shown in Figure 2. The links in-between nodes in each cluster, and links between bridging nodes, are modelled by a two state Markov link model [11], with an expected contact duration of 60 seconds and inter-contact time of 240 seconds. Each of the three clusters is comprised of five nodes.

HCMM:SO — This trace is generated using the Home-cell Community based Mobility Model with Social Overlay [9], which aims to accurately model both intra- and inter-community contact patterns. The intra-community pattern is achieved by assigning nodes to home cells (or caves) according to a caveman social graph [22], as illustrated in Figure 3. Nodes then move to another cell C with a probability proportional to the number of their "friends" (in the social graph) that have C as their home cell. The inter-community contact pattern is modelled through periods of social activity, in which community *bridge nodes* convene at common locations according to a complementary social graph overlay. This model achieves weak non-homogenous mixing. We use a HCMM configuration with 30 nodes, in 6 communities a 5 nodes distributed over 10x10 cells, and a social overlay with inter-community movement of 2 hours but otherwise the same parameters as in [9]. The nodes have a mean contact duration of 58s, intercontact time of 166s and 38% of the nodes meet each other.

Real World — A real-world trace, collected from an experiment with 10 mobile phones, complements our two synthetic traces. The mobile phones were handed out to colleagues in our office corridors, who carried them during their

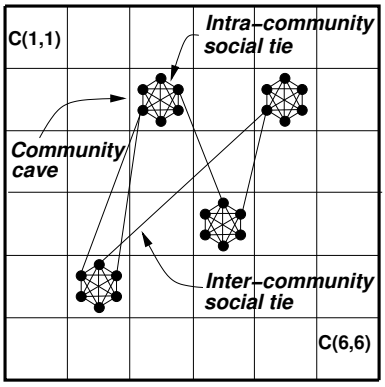


Figure 3: HCMM:SO contact model example: Each node is assigned a home cell and moves to other cells according to its social relations there (source [9]).

work day. The phones used WiFi to send beacons every five seconds, allowing them to detect each other when in range. The WiFi interfaces used a low transmit power of 2 dBm to reduce battery consumption. The experiment lasted around six hours and people attended their normal work schedules; they sat in their offices, chatted with colleagues, had lunch, and participated in meetings. The nodes have a mean contact duration of 135s, intercontact time of 409s and 82% of the nodes meet each other.

4.2 Interest Modeling and Data Distribution

To model a data-centric network, interests and data should be distributed in a way that reflects the non-uniformity of item popularity in real life. Studies on user interest distribution have been shown to exhibit Zipf-like behaviour [7], we therefore model our distribution of interests and content metadata accordingly. To this end, we form node interests by assigning each node a set of five attributes $\mathcal{I}$ from a global attribute pool $\mathcal{A}$ of cardinality 100. Each user's interest attributes are drawn from $\mathcal{A}$, which is a Zipfian distribution with skewness parameter α. A higher skewness leads to a higher overlap of the interest between the nodes, while $\alpha = 0$ corresponds to a uniform distribution of the interests. This method allows to control the strength of the interest segregation among nodes. Each attribute $a \in \mathcal{I}$ is also given a weight w_a, proportional to the likelihood of picking a from $\mathcal{A}$, and normalised such that the sum of a node's interest weights equals 100.

We assign each data item a set of five attributes $\mathcal{D}$ from the attribute pool $\mathcal{A}$ with the same probability distribution as above. The node interests and item attributes use the same distribution, as libraries generally reflect the tastes of the owners. We chose to inject all data items at the beginning of the experiments to enable complete analysis of a single wave of item distribution and so that each data item has equal conditions (i.e., node contacts) to disseminate.

Parameter	Symbol	Default Size
Attribute pool	$\mathcal{A}$	100
Data attributes	$\mathcal{D}$	5 (Zipf, $\alpha = 0.5$)
Interest attributes	$\mathcal{I}$	5 (Zipf, $\alpha = 0.5$)
Data items per node		200

Table 1: Parameters used when generating (meta)data and interests.

We explore different values of α for the attribute pool $\mathcal{A}$, unless otherwise specified we use $\alpha = 0.5$ as a default. This reflects a balanced attribute overlap, avoiding a situation where all nodes have interest in every data item. Instead, with our configuration, a node has an interest in $\sim 22\%$ of all data items. The experiment parameters are summarised in Table 1.

4.3 Data Transfer

To focus on the transfer ordering effect when forwarding, rather than whether an item is selected for forwarding, we artificially limit the number of transfers per node contact. This to emphasise the impact of limited contact duration, and to abstract away from Bluetooth or WiFi bandwidth offered by mobile phones, data size, and the distribution of contact duration in the given contact traces.

The results presented in the following section are derived with a limit of 10 data objects transferred per node contact. Different configurations with up to 100 data objects per node contact showed comparable relations between the different strategies, we therefore concentrate the analysis on one configuration. Furthermore, a low number of transfers reflects resource conservative nodes, a likely situation in real life.

4.4 Evaluation Metrics with Relevance Focus

To measure the effectiveness of relevance-driven dissemination we use the normalised Discounted Cumulative Gain (nDCG) [13], an established metric in the information retrieval community. The nDCG assigns a given result set a quality valuation between 0 and 1, and assesses the ordering of data items in a set compared to an ideal order. In our case, the ideal ordering is obtained for a node by taking the top T sorted (according to relevance) data items out of all possible items, while the experienced ordering is the top T sorted sequence of items that have been delivered to the node in the experiment. The nDCG metric thus gives a measure of how successful a strategy is at delivering data of high relevance. The nDCG is computed as follows:

$$nDCG^n[T] = n_T(s_1 + \sum_{i=2}^{T} s_i / log_b(i)) \tag{2}$$

where s_i is the score of the i^{th} relevant received data item (i.e., its local order is i) and $1/n_T$ is the $nDCG^{n-ref}[T]$ for the relevant data available in the system used for normalisation. This measure will only consider how well the system provides a node's T most desired data items, items below rank T will not contribute utility to the node. The network-wide nDCG is calculated as the mean of the per-node nDCG^n, and gives a measure of a strategy's performance for the network as a whole. It can be seen that all nodes are weighted equally to this systemic measure, that is, nodes with low interest in available items contribute equally to the network-wide score.

The nDCG does not give a complete picture of the performance of a strategy, revealing nothing about the cost to achieve a certain nDCG or whether less relevant data is unnecessarily delayed in order to achieve a high nDCG. We therefore complement our analysis with traditional metrics, such as *delivery ratio*, *delay* and *overhead*. However, these metrics are well understood and require no further introduction.

5 Results

In this section we look at the ability of dissemination strategies to deliver relevant items, the delay items suffer and how these metrics evolve through the experiment. This allows us to understand the learning process of globally oriented strategies, which require time to learn about the interests of other nodes. The cost of dissemination (i.e., overhead) also reveals important trade-offs between performance and transfer efficiency.

5.1 Delivery Ratio

Each node is interested in a subset of the data distributed across the nodes in the network. A single node's delivery ratio is the fraction of relevant items that have been received by the node. The network-wide delivery ratio is the mean node-specific delivery ratio. It gives an indication of how successful the network has been at delivering data items of relevance to all nodes throughout the network.

Dissemination Across Segmented Networks The distribution of data items and interests, the contact patterns, and the length of the experiment all constrain the delivery process. In particular, interest and data distributions affect strategies that only make local considerations. These strategies suffer when networks are segmented, because nodes isolated from each other must rely on other nodes to transfer the data they desire from other parts of the network. With local strategies, such transfers only occur when the relevant data items are also of interest to the recipient.

The *Clusters in Line* topology helps us to study the effect of segmented networks on local strategies. The delivery ratio across clusters is bound by the in-

Zipf α	0.25			0.5			0.75		
dst\src	c0	c1	c2	c0	c1	c2	c0	c1	c2
c0	100	13	0	100	19	1	100	30	17
c1	15	100	17	14	100	24	23	100	64
c2	0	16	100	0	20	100	13	65	100

Table 2: Fraction of inter-cluster data delivered for different Zipf α in the *Clusters in Line* topology, using any local strategy.

terests of the bridge nodes; only data of interest for the bridge node in cluster c0 will be disseminated from cluster c1 to cluster c0, for example. By changing the α parameter of the Zipf distribution, we can study how the node interests affect the dissemination. Table 2 shows the fraction of items delivered from one cluster to another. About 15% of relevant data items are able to traverse to adjacent clusters when $\alpha = 0.25$. No items traverse between the two non-adjacent clusters (c0 and c2). Increasing α to 0.5 allows one percent of the items to traverse from c0 to c2, also slightly improving adjacent traversal. Raising α to 0.75 improves the non-adjacent dissemination further.

As local strategies only transfer locally relevant data, every transferred item contributes to the delivery ratio independent of the ordering strategy. While the strategies eventually achieve the same delivery ratio, their progress is different as the availability of relevant data items becomes scarce. The next section discusses this aspect.

Delivery Ratio Progress The rate at which relevant items are delivered gives an indication of the strategies' ability to prioritise the network-wide delivery ratio. HCMM:SO periodically exposes nodes to new neighbours, allowing us to examine the effect of the local saturation of neighbours' items. We can see this in Figure 4, where the progress of delivery ratio is given for all strategies. Local strategies perform well initially but start to experience limited availability of items left to share, reducing progress until new nodes are encountered (as seen at hours 2, 6, and 8). This pattern is repeated throughout the experiment. Random and ScoreGlobal strategies select more data items, and therefore suffer less from this local saturation effect.

Successful delivery of new items will be constrained when all contacts are used to distribute the same data, the network needs variety to ensure all interests are satisfied. This can be seen in the slow growth of ScoreGlobal, which aims to have all nodes swapping the same "best" items, starving delivery of less globally relevant items. For this reason, Random begins to outperform the others after 7 hours, as it enforces variety in dissemination among nodes.

Variety in data among the nodes is in particularly important when local saturation affects the ability to progress in data delivery. LiFoLocal performs worst of the

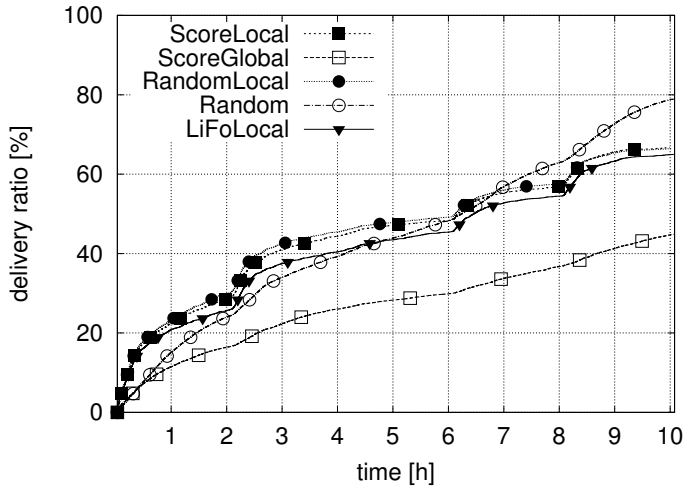


Figure 4: HCMM:SO: Delivery ratio progress over time, $\alpha = 0.5$.

Local strategies in terms of delivery ratio because promoting fresh data does not improve variety in data among the nodes the same way as random or score-based ordering do.

5.2 Received Item Relevance

The delivery ratio by itself does not give any information about the ability of a strategy to disseminate the *most* relevant data. We measure the relevance of the data received by nodes compared to the most relevant data available in the system using the nDCG metric. Figure 5 shows the network-wide nDCG progress evaluated over the 10 most relevant received data items for the three topologies. At the beginning of all experiments, ScoreLocal consistently delivers the most relevant data, as the strategy was designed to exchange the most relevant data first. In the *Clusters in Line* topology we see the ScoreGlobal strategy eventually exceed the performance of ScoreLocal due to network segmentation from which local strategies suffer. It is also interesting to note that ScoreGlobal develops its knowledge about other nodes over time, at the first node contact it is equivalent to ScoreLocal, as it does not have previous knowledge about other nodes. Thus the strategy initially achieves a higher nDCG than random or freshness based strategies but loses this behaviour over time as it meets more nodes and develops a global view. The HCMM:SO results begin high due to the large amount of early contacts.

In terms of relevance, using data freshness as an ordering criteria does not give an advantage over random ordering, so the promotion of newly received items does not have the same beneficial effect as it does in buffer management [4]. Moreover, its tendency to distribute the same data to many nodes can lead to worse results than random ordering due to excessive redundancy.

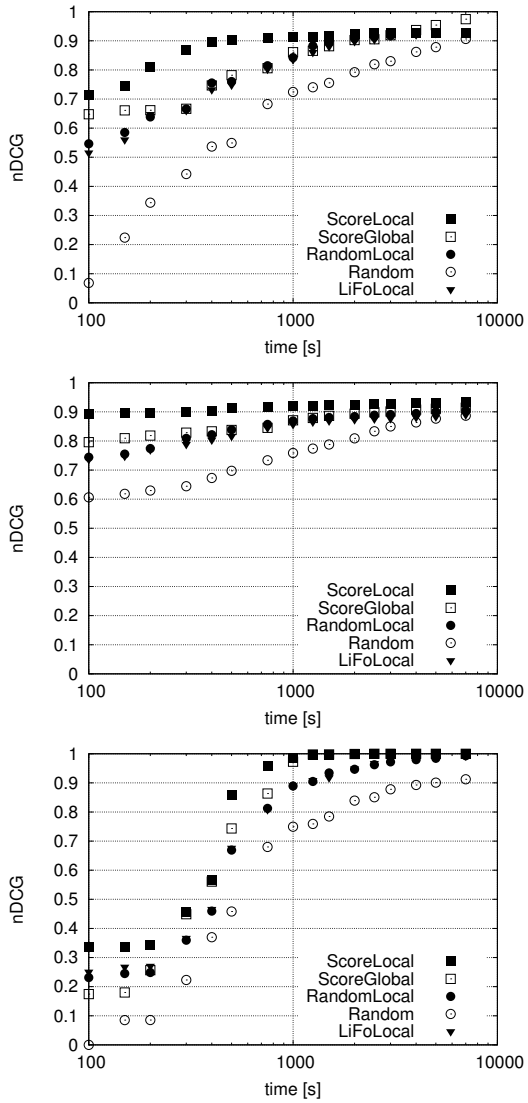


Figure 5: Normalized Discounted Cumulative Gain nDCG evaluated over the 10 most relevant received data over time of the experiment for (a) Clusters in Line (b) HCMM:SO (c) Real World

Interest Segregation The effect of varying α for the *Clusters in Line* topology is depicted in Figure 6. For high α, most nodes and items will share the more likely attributes. When α is low, attributes are evenly distributed and so it will be unlikely that two adjacent nodes find the same item relevant. This lack of shared relevance limits items' ability to spread with a local strategy. We can see the effect of this in Figure 6, where the local strategies do not reach 100%. This is due to the segregation of nodes with a matching interest in items. Whereas non-local strate-

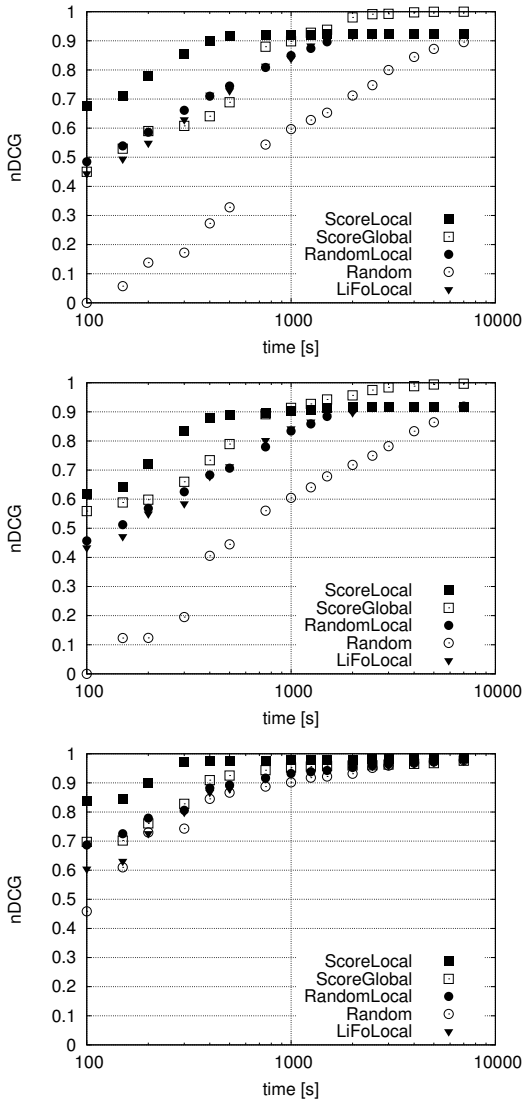


Figure 6: Normalized Discounted Cumulative Gain nDCG on *Clusters in Line* topology evaluated over the 10 most relevant received data over time of the experiment for varying values of α (a) $\alpha = 0$ (b) $\alpha = 0.25$ (c) $\alpha = 1$

gies can overcome these limitations, as transferred items do not need to interest the recipient.

With sparse topologies the path length to other nodes will be longer than in dense topologies. Longer paths are less likely for all adjacent nodes to share interest in data items. Thus shorter path lengths will not suffer from as much segregation.

As ScoreGlobal develops a more informed view of other nodes' interests, it

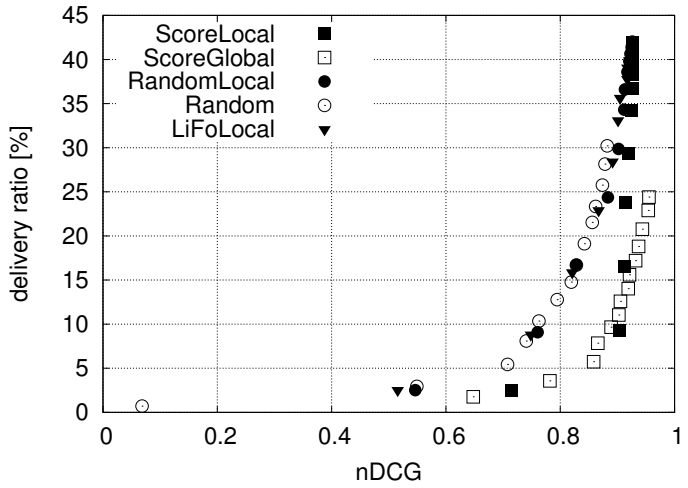


Figure 7: Clusters in Line: Normalized Discounted Cumulative Gain nDCG versus delivery ratio.

will gain a more uniform view of interest distribution. A lack of strong views on what should be shared avoids segregation.

Informed ordering of data items is of particular concern when nodes have particularly segregated interests (low α). Whereas with highly skewed interests (high α) the Random strategy does not suffer from its simplicity.

Relevance versus Delivery Ratio Comparing relevance against the delivery ratio gives an indication about the efficiency of the strategies. While, achieving a high nDCG with low delivery ratio is likely to involve transfer of irrelevant data, it shows that the data delivered actually is the most relevant one.

In Figure 7, we plot the nDCG against the delivery ratio. Each data point represents periodic instances in time as both delivery ratio and nDCG monotonically increase over the experiment. More separation between points thus shows better progress towards delivery and relevance.

Random and freshness-based ordering show the same behaviour, while the Random strategy has slowest progress. Relevance-based ordering naturally achieves higher nDCG for the same delivery ratio than other orderings, with the global selection strategy ScoreGlobal overcoming the high relevance/low delivery property slower. This indicates the increased cost of achieving its higher relevance score at the end of the experiment. ScoreLocal on the other hand cannot take advantage of the increasing delivery ratio to improve its nDCG score due to the previously discussed interest segregation.

Zipf α	0	0.25	0.5	0.75	1
*Local	1.00	1.00	1.00	1.00	1.00
ScoreGlobal	3.09	2.98	2.41	1.51	1.08
Random	3.77	3.63	2.97	1.86	1.24

Table 3: Clusters in Line: Overhead in terms of number of transferred data items per received relevant data item.

5.3 Transfer Efficiency

We use the overhead in the number of transferred data items per received relevant data item as measure of efficiency. Table 3 summarises the overhead for different data an interest distributions for the *Clusters in Line* topology. While the overlap in data and interest distribution affects the overhead needed to disseminate the data, we found that the results are similar for the three topologies. Local strategies achieve optimal overhead of 1.0 as every transferred data item is relevant to its recipient.

Both ScoreGlobal and Random try to benefit the whole system and therefore use contact opportunities to transfer data that is not necessarily relevant to its recipient. As the interests of the recipient are included in the aggregated interests used to select data, the chances of transferring relevant data items are higher for ScoreGlobal than using the Random strategy. Looking at the data and interest distribution, we observe that increased skewness leads to a reduction in overhead. This is because more nodes are interested in the same data item, increasing chances that a recipient happens to be interested in the item. The overhead of the ScoreGlobal strategy almost reaches optimality with a skewness of $\alpha = 1$, reflecting that the individual node interests are highly overlapping given that data and interest distribution.

6 Conclusions

This paper proposed that the ordering of data items in transfer limited data-centric opportunistic networks has a large impact on the relevance of received data. We experimentally compared five strategies that select and order data to be transferred and evaluated their ability to deliver the most relevant data, measured using the normalised Discounted Cumulative Gain (nDCG). The trace-driven experiments on an emulation testbed included contact traces with a range of densities and mixing.

The selection and ordering of transferred items significantly affects the achieved delivery ratio and its rate of increase, respectively. If nodes desire highly relevant data, careful ordering of limited transfers can satisfy these desires quicker than order agnostic approaches. With local strategies, if it is unlikely that neighbours are interested in the same content, segregation can occur where nodes are unable to

disseminate items through the network. Interestingly, considering the relevance of data provides a way to overcome this segregation.

In the case of densely connected networks, just local decisions can achieve system properties. However, in particularly sparse networks we need to explicitly consider non-local information to successfully promote system concerns. We saw how selecting items to transfer using global information eventually outperforms local approaches. However, this does not mean that changing strategies in the experiment achieves the best of both approaches, as the early transfers dictate later performance.

We investigated limited transfer capabilities in data-centric opportunistic dissemination systems. We propose that the ordering of item transfers is a behaviour that, while rarely explicitly considered, has a large impact on the timely delivery of the most relevant data. A fully developed dissemination system should still consider more aspects than just relevance. Variable item size, contact duration and buffer size would complicate ordering decisions. It may then be advantageous to consider how best limited capacity and storage should be used [17].

For future work, we plan to consider injecting data into nodes continuously, the correlation between node topology and interest structures, and an analysis of item diversity and fairness aspects.

Acknowledgements

This research was funded by the ResumeNet project under the EU grant FP7-224619. We thank Theus Hossmann for making the HCMM:SO code and Figure 3 available to us.

References

[1] Haggle code project page, http://haggle.googlecode.com.

[2] BALASUBRAMANIAN, A., LEVINE, B., AND VENKATARAMANI, A. DTN Routing as a Resource Allocation Problem. In *ACM SIGCOMM Computer Communication Review* (2007), vol. 37, ACM.

[3] BJUREFORS, F., GUNNINGBERG, P., AND ROHNER, C. Haggle Testbed: a Testbed for Opportunistic Networks. In *7th Swedish National Computer Networking Workshop* (2011).

[4] BJUREFORS, F., GUNNINGBERG, P., ROHNER, C., AND TAVAKOLI, S. Congestion Avoidance in a Data-Centric Opportunistic Network. In *ACM SIGCOMM ICN Workshop* (2011).

[5] BOLDRINI, C., CONTI, M., IACOPINI, I., AND PASSARELLA, A. HiBOp: a History Based Routing Protocol for Opportunistic Networks. In *IEEE WoWMoM* (2007).

[6] BOLDRINI, C., CONTI, M., AND PASSARELLA, A. ContentPlace: Social-Aware Data Dissemination in Opportunistic Networks. In *ACM MSWiM* (2008).

[7] BRESLAU, L., CUE, P., CAO, P., FAN, L., PHILLIPS, G., AND SHENKER, S. Web Caching and Zipf-like Distributions: Evidence and Implications. In *INFOCOM* (1999).

[8] HELGASON, O. R., YAVUZ, E. A., KOUYOUMDJIEVA, S. T., PAJEVIC, L., AND KARLSSON, G. A Mobile Peer-to-Peer System for Opportunistic Content-Centric Networking. In *MobiHeld* (August 2010).

[9] HOSSMANN, T., SPYROPOULOS, T., AND LEGENDRE, F. Putting Contacts into Context: Mobility Modeling beyond Inter-Contact Times. In *MobiHoc* (Paris, France, May 2011).

[10] HUI, P., CROWCROFT, J., AND YONEKI, E. Bubble Rap: Social-based Forwarding in Delay Tolerant Networks. In *MobiHoc* (2008).

[11] HWANG, S., AND KIM, D. Markov model of link connectivity in mobile ad hoc networks. *Telecommunication Systems 34* (2007).

[12] JAIN, S., FALL, K., AND PATRA, R. Routing in a Delay Tolerant Network. *ACM SIGCOMM CCR 34*, 4 (2004).

[13] JÄRVELIN, K., AND KEKÄLÄINEN, J. IR evaluation methods for retrieving highly relevant documents. In *ACM SIGIR* (2000).

[14] KRIFA, A., BARAKA, C., AND SPYROPOULOS, T. Optimal Buffer Management Policies for Delay Tolerant Networks. In *SECON* (2008), IEEE.

[15] KRIFA, A., BARAKAT, C., AND SPYROPOULOS, T. MobiTrade: Trading Content in Distruption Tolerant Networks. In *CHANTS* (September 2011).

[16] LENDERS, V., KARLSSON, G., AND MAY, M. Wireless Ad Hoc Podcasting. In *IEEE SECON* (June 2007).

[17] LI, Q., ZHU, S., AND CAO, G. Routing in Socially Selfish Delay Tolerant Networks. In *INFOCOM* (2010), IEEE.

[18] LINDGREN, A., DORIA, A., AND SCHELÉN, O. Probabilistic Routing in Intermittently Connected Networks. *SIGMOBILE MCCR* (2003).

[19] LINDGREN, A., AND PHANSE, K. Evaluation of Queueing Policies and Forwarding Strategies for Routing in Intermittently Connected Networks. In *Comsware* (2006).

[20] MOGHADAM, A., AND SCHULZRINNE, H. Interest-Aware Content Distribution Protocol for Mobile Disruption-Tolerant Networks. In *WoWMoM* (2009).

[21] MTIBAA, A., MAY, M., DIOT, C., AND AMMAR, M. PeopleRank: Social Opportunistic Forwarding. In *IEEE INFOCOM* (2010).

[22] NEWMAN, M. E. J. The Structure and Function of Complex Networks. *SIAM REVIEW 45* (2003).

[23] NORDSTRÖM, E., GUNNINGBERG, P., AND ROHNER, C. A Search-based Network Architecture for Mobile Devices. Tech. Rep. 2009-003, Department of Information Technology, Uppsala University, Jan. 2009.

[24] SOLLAZZO, G., MUSOLESI, M., AND MASCOLO, C. TACO-DTN: A Time-Aware Content-based Dissemination System for Delay Tolerant Networks. In *MobiOpp* (2007).

[25] SPYROPOULOS, T., PSOUNIS, K., AND RAGHAVENDRA, C. S. Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks. In *ACM SIGCOMM WDTN Workshop* (2005).

[26] VAHDAT, A., AND BECKER, D. Epidemic Routing for Partially-Connected Ad Hoc Networks. Tech. rep., 2000.

Recent licentiate theses from the Department of Information Technology

2012-001 Gunnika Isaksson-Lutteman: *Future Train Traffic Control – Development and deployment of new principles and systems in train traffic control*

2011-006 Anette Löfström: *Intranet Use as a Leadership Strategy*

2011-005 Elena Sundkvist: *A High-Order Accurate, Collocated Boundary Element Method for Wave Propagation in Layered Media*

2011-004 Niclas Finne: *Towards Adaptive Sensor Networks*

2011-003 Rebecka Janols: *Tailor the System or Tailor the User? How to Make Better Use of Electronic Patient Record Systems*

2011-002 Xin He: *Robust Preconditioning Methods for Algebraic Problems, Arising in Multi-Phase Flow Models*

2011-001 David Eklöv: *Efficient Methods for Application Performance Analysis*

2010-005 Mikael Laaksoharju: *Let Us Be Philosophers! Computerized Support for Ethical Decision Making*

2010-004 Kenneth Duru: *Perfectly Matched Layers for Second Order Wave Equations*

2010-003 Salman Zubair Toor: *Managing Applications and Data in Distributed Computing Infrastructures*

2010-002 Carl Nettelblad: *Using Markov Models and a Stochastic Lipschitz Condition for Genetic Analyses*

2010-001 Anna Nissen: *Absorbing Boundary Techniques for the Time-dependent Schrödinger Equation*

UPPSALA
UNIVERSITET

Department of Information Technology, Uppsala University, Sweden