DISRUPTION-TOLERANT NETWORKING PROTOCOLS AND SERVICES FOR DISASTER RESPONSE COMMUNICATION

BY

MD YUSUF SARWAR UDDIN

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science in the Graduate College of the University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Professor Tarek Abdelzaher, Chair
Associate Professor Robin Hillary Kravets
Assistant Professor Philip Brighten Godfrey
Professor Guohong Cao, Pennsylvania State University
ABSTRACT

This thesis investigates, and contributes thereto, the promise of networking technologies and data processing techniques for efficient and effective information dissemination in the context of disasters. Disasters hinder usual communication experiences of people, and thereby calls for different approaches to meet communication needs. This is because traditional communication options, e.g., the Internet, may not be available after a disaster due to infrastructure damage or power outage. As a sign of hope, average individuals these days possess numerous wireless devices, such as smart mobile phones and tablets, with various device-to-device connectivity options, such as WiFi and Bluetooth. These devices with their capabilities can be harnessed in a disaster aftermath in the form of a disruption-tolerant network (DTN). DTN copes with intermittent connectivity among devices by using persistent storage and by transporting data packets exploiting mobility of nodes. Our dissertation efforts augment DTN literature with a new set of protocols and application services by leveraging specificities that arise in disaster context.

From a boarder perspective, we divide our efforts into two main threads considering two key aspects of DTNs: i) mobile nodes and ii) moving content, that is, in DTNs node move and they generate moving content. Accordingly, in order to improve network efficiency and to design effective content dissemination protocols, we need to investigate mechanisms of two main categories: (i) exploiting physical mobility patterns of nodes, and (ii) exploiting logical properties of content generated by mobile nodes. Specifically, we leverage recurrence, as an example of exploiting mobility patterns of nodes, and propose a routing protocol, called inter-contact routing (ICR) that exploits recurrence in its core design. Later, we explore redundancy, as an example of exploiting properties of content, and propose redundancy reduction and in-network content prioritization techniques for different application services we build, namely PhotoNet, PhotoNet+ and diversity caching.
To my mother.
ACKNOWLEDGMENTS

All praises are due to Allah, the most benevolent and merciful. I am thankful to my advisor Prof Tarek Abdelzaher for his phenomenal support and encouragement during the whole PhD program, not to mention completing this dissertation work. He taught me how to develop critical thinking ability on a giver problem, which is undoubtedly one of the most important aspects of one becoming a PhD. His cooperation, advices, and suggestions have been instrumental all the way. At this moment, I feel nothing but a sincere gratitude for his being my advisor. I am also thankful to my PhD committee members: Prof Robin Kravets, Prof Brighten Godfrey, and Prof Guohong Cao. Their feedbacks and recommendation on various issues of this dissertation work have been very useful.

I am thankful to Fulbright Science and Technology Award for providing me fellowship for the first three years of my PhD program. I am particularly delighted because I was one of the first recipients of this scholarship, which has been sponsored by US Department of State since Fall 2007. Under this program, I had opportunities to travel various cities of the United States and to interact with other fellow Fulbrighters from other countries. These were very exciting moments and memorable experiences of my life. I thank the program director, state department officials and others associated with this Fellowship.

In this connection, I also remember my fellow lab-mates, some of whom are also co-authors of my various papers, such as Hossien Ahmadi, Fatemeh Saremi, Hieu Le, Dong Wang, Tanvir Al-Amin, Raghu Ganti, and Praveen Jayachandran. I also especially thankful to Mohammad Maifi Hasan Khan, one of my fellow lab-mates, for this constant encouragement for doing good and impactfull works. In October 2011, I was in a short visit to Boston to work in BBN Raytheon Technologies that gave me a new dimension in my research. I am thankful to my program mentor Dr. Alice Lueng and
co-worker William Dron, and also my room-mate at those days, Roberto De Matos.

I have been in Champaign for about five years. I would always remember the wonderful Bangladeshi Community I lived in during all these days. I am thankful to Bangladeshi Student Association at UIUC (BSA-UIUC) for all of its activities, events and occasions that made all Bangladeshi students living in Champaign feel like home. I had the privilege to serve as President of this association for two terms. Beside students, senior professional members such as Prof Salim Rashid, Prof Ali, Prof Taher Saif, Tarek bhai, Nadeem bhai, Sharif bhai, and others, accompanied us in all social events.

Last of all, I am grateful to my family members: my widow mother and my two brothers. In this occasion, I also express my thanks to my beloved wife, Rezwana Rafiq, for her inspiring accompany and being with me till the end.
# TABLE OF CONTENTS

LIST OF TABLES ........................................................................... ix

LIST OF FIGURES ....................................................................... x

CHAPTER 1 INTRODUCTION ......................................................... 1
  1.1 Challenges and Opportunities in Disaster Communication .... 4
    1.1.1 Intermittent, but Recurrent Connectivity .............. 4
    1.1.2 Redundant Data Content ................................. 5
  1.2 Dissertation Statement .................................................. 6
  1.3 Contribution of This Thesis ............................................ 6
    1.3.1 DTN Protocols Exploiting Recurrence ............. 7
    1.3.2 Diversity Maximization Protocols for DTN Apps .... 8
    1.3.3 Disaster Mobility Model and Disaster DTNs .......... 10
  1.4 Organization of the Thesis ............................................. 10

CHAPTER 2 INTER-CONTACT ROUTING PROTOCOL ............... 11
  2.1 Low-energy Inter-contact Routing (ICR) ......................... 13
    2.1.1 A Novel Network Model for DTNs ..................... 14
    2.1.2 The Routing Algorithm ................................ 16
  2.2 An Energy-differentiated Service for ICR ....................... 23
    2.2.1 Energy Model and Protocol ............................ 24
    2.2.2 Enforcement Issues ..................................... 25
  2.3 A Post-Disaster Mobility Model ................................. 26
  2.4 Evaluation and Simulation Results ............................... 27
    2.4.1 Simulator Settings ....................................... 28
    2.4.2 Performance of ICR ...................................... 29
    2.4.3 Results in Energy-constrained Environments ....... 33
    2.4.4 Impact of Protocol Parameters ....................... 37
  2.5 Conclusion and Future Work ....................................... 40

CHAPTER 3 DELAY BOUND FOR RECURRENT DTNS ............... 41
  3.1 Modeling Recurrent DTNs ............................................. 43
    3.1.1 Network Model .......................................... 43
    3.1.2 Data Flow Model ....................................... 48
  3.2 End-to-end Delay Bound for DTNs .............................. 48
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.2 Formulation without Clusters</td>
<td>125</td>
</tr>
<tr>
<td>6.3 Protocol Design and Implementation</td>
<td>129</td>
</tr>
<tr>
<td>6.3.1 Data and Query Model</td>
<td>130</td>
</tr>
<tr>
<td>6.3.2 Computing Distance Function</td>
<td>130</td>
</tr>
<tr>
<td>6.3.3 Communication Model</td>
<td>130</td>
</tr>
<tr>
<td>6.3.4 Responding to Queries</td>
<td>131</td>
</tr>
<tr>
<td>6.3.5 Computing Popularity of Objects</td>
<td>132</td>
</tr>
<tr>
<td>6.3.6 Purging Objects from Caches</td>
<td>133</td>
</tr>
<tr>
<td>6.4 Evaluation</td>
<td>133</td>
</tr>
<tr>
<td>6.4.1 Simulation Environment</td>
<td>134</td>
</tr>
<tr>
<td>6.4.2 NS3 Wireless Emulation</td>
<td>134</td>
</tr>
<tr>
<td>6.4.3 Performance Evaluation</td>
<td>135</td>
</tr>
<tr>
<td>CHAPTER 7 SIMULATING DISASTER DTNS AND EXPERIMENTS</td>
<td>148</td>
</tr>
<tr>
<td>7.1 Post-Disaster Mobility Model (PDM)</td>
<td>149</td>
</tr>
<tr>
<td>7.1.1 Disaster Area and Neighborhoods</td>
<td>150</td>
</tr>
<tr>
<td>7.1.2 Post-Disaster Relief Operation</td>
<td>151</td>
</tr>
<tr>
<td>7.2 Implementation</td>
<td>154</td>
</tr>
<tr>
<td>7.2.1 Extension to ONE</td>
<td>154</td>
</tr>
<tr>
<td>7.2.2 Configuration Parameters</td>
<td>155</td>
</tr>
<tr>
<td>CHAPTER 8 RELATED WORK</td>
<td>157</td>
</tr>
<tr>
<td>8.1 DTN Networking Protocols</td>
<td>157</td>
</tr>
<tr>
<td>8.2 Content-aware Networking and Redundancy Reduction Techniques</td>
<td>161</td>
</tr>
<tr>
<td>8.3 Caching Techniques in DTNs</td>
<td>162</td>
</tr>
<tr>
<td>8.4 DTN Mobility Models</td>
<td>163</td>
</tr>
<tr>
<td>8.5 Applications for Disaster Communication</td>
<td>164</td>
</tr>
<tr>
<td>CHAPTER 9 CONCLUSION AND FUTURE WORK</td>
<td>165</td>
</tr>
<tr>
<td>9.1 Summary of Contribution</td>
<td>165</td>
</tr>
<tr>
<td>9.2 Future Research Directions</td>
<td>166</td>
</tr>
<tr>
<td>9.2.1 Multi-Modal Communication for Disaster Response</td>
<td>166</td>
</tr>
<tr>
<td>9.2.2 Energy Efficiency in Networked Devices</td>
<td>166</td>
</tr>
<tr>
<td>9.2.3 Rich Sensors for Wide Area Sensing</td>
<td>167</td>
</tr>
<tr>
<td>9.2.4 QoS, QoE, now QoI?</td>
<td>167</td>
</tr>
<tr>
<td>9.2.5 Information-centric Networking</td>
<td>168</td>
</tr>
<tr>
<td>9.2.6 Content-aware Prioritization and Named-Data Networking</td>
<td>168</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>171</td>
</tr>
</tbody>
</table>
LIST OF TABLES

3.1 Simulation parameters .................................. 57
4.1 Simulation parameters .................................. 82
4.2 Various timing values on Android phones ........... 86
4.3 Average feature extraction time from pictures ........ 86
5.1 Various diversity metrics used by biologists and ecologists . 96
7.1 Newly added Movement classes to ONE. ............ 155
## LIST OF FIGURES

1.1 News article showing power and communication outage in a recent US storm (featured on July 3, 2012). .................. 3

1.2 (a) Content-aware prioritization inside the network. (b) CAP runs at each node. ................................. 9

2.1 A scenario showing encounter and inter-contact graph. .......... 15

2.2 Node \( i \) is deciding message copies for \( j \). .................. 22

2.3 The disaster area. Circles are neighborhoods. Highlighted path is the part of police patrol. .......................... 27

2.4 Performance comparison, ICR, Spray-and-Wait, Spray-and-Focus, MaxProp and Prophet. ...................... 28

2.5 Comparison with RWP and City Scenario. ...................... 30

2.6 Performance of ICR in different mobility scenarios and traffic patterns. ............................................. 31

2.7 Energy consumption, lifetime and delivery ratio for various routing protocols. ............................................. 33

2.8 (a) Delivery ratio at different initial energy values (b) Dissection of energy component of ICR. ................... 35

2.9 Delivery of regular and urgent messages. ........................ 36

2.10 Total number of messages generated. ........................... 37

2.11 ICR on inter-contact and encounter graph and effect of timeout on delivery ratio. .............................. 38

2.12 Bootstrapping ICR with Spray. ................................. 38

2.13 Performance results when moving agents deviate from their recurrent patterns for a while. ......................... 39

3.1 An example of delay computation. ................................. 45

3.2 End-to-end delay components for a flow. ...................... 49

3.3 Reduction of a data flow graph by the delay composition algebra. ......................................................... 51

3.4 Linearization of transfer delay. .................................. 52

3.5 Pessimism in end-to-end delay computation. .................. 55

3.6 The data flow graph containing all flows. ...................... 56

3.7 Delay metrics of vertices and edges of the data flow graph. ................................................................. 57

3.8 Delay bound ratio for varying loads. ............................. 57
6.10 Effect of metadata noise and multiple mules. . . . . . . . . . . 141
6.11 Effect of degree of burstiness of data traffic. . . . . . . . . . . 142
6.12 Query heatmap at various degree of query pattern generation. 144
6.13 Performance at various traffic load and query distribution. . . 145
6.14 Performance results at various value of query radius. . . . . . 145
6.15 Performance results at various $\lambda$'s. . . . . . . . . . . . . 146
6.16 Performance results at various $\lambda$'s compared to LRU. . . . 147

7.1 Various components of the mobility model. . . . . . . . . . . . 151
7.2 Different mobility patterns. . . . . . . . . . . . . . . . . . . . . 152
CHAPTER 1
INTRODUCTION

This thesis investigates, and contributes thereto, the promise of networking technologies and data processing techniques for efficient and effective information dissemination in the context of disasters. Disasters, either natural such as hurricanes and earthquakes man-made (e.g., nuclear catastrophe) hinder usual communication experiences of people and therefore call for different approaches to meet communication needs. Communication becomes essential during disaster in order to conduct search and rescue operation and to minimize injury of lives and damage of properties. Various kinds of information may be needed to collect and disseminate. Information may include situational aspects of the environment (e.g., location of damages, flooded areas, infrastructure collapses) as well as the current or the last-known status of individuals affected by the disaster (e.g., health hazards, spread of diseases). First responders, rescue workers, and volunteers deployed in the area need comprehensive, reliable and high quality information in order to make well-informed decisions to coordinate relocation efforts and to manage supply materials and relief goods (e.g., food, water, shelters) to the survivors. Information is also required by people residing outside of the disaster area to know whereabouts of their dear and loved ones [1]. It is therefore important to enable data collection from inside a disaster area and then delivering it to a service point where it can be processed, actuated, and shared.

Recent years witnessed an enormous proliferation of devices capable of doing computation and communication efficiently. These devices, such as mobile phones, tablets and other digital gadgets, with a variety of data connectivity provisions such as WiFi, 3G, 4G, LTE, Bluetooth, and WiMax, make communication experiences quite pervasive, even when people are moving. This potential can be effectively harnessed during or after a disaster too. Affected people carrying these devices can gather information and share onto others as they experience various events during a disaster. Information of
various formats, such as text, images, audio and video clips, can be gathered and disseminated among affected people or can be incorporated into a common shared application.

There are studies that provide evidence that people indeed generate and consume information during emergency moments. Numerous web applications have been in use in various emergency and civil crisis moments in near history. The web application Ushahidi (http://Ushahidi.com) is one such example that has been used during Kenyan election violences in 2008. The service allows individuals to generate reports with text and photos that are aggregated and displayed on a Google map or as a list of text reports. It has been shown that people converge to social networking and online media streaming sites, such as Twitter, during crisis moments in order to update their latest status and to know whereabouts of others. Haiti earthquake 2010 [2], Japan’s Fukushima nuclear disaster 2011 [3] and Egypt’s political protests 2011 [4] are some very recent incidents to name where such activities of people have been widely documented. In short, crisis-assistant web applications, generally called public webspace powered by the Web 2.0 and the Web 3.0 standards [5], have great promise in allowing people to communicate and share their latest conditions and events with others by micro-blogging (e.g., [6]), photo sharing (e.g., [7]), and by using other social networking sites. These are all supportive to the necessities of communication during disasters.

While Internet-based services are useful during disaster, in disasters, however, Internet connectivity may be not available or may be available only intermittently with limited capacity. This might be due to infrastructure collapse or power outage. In particular, power outage is recorded to be a commonplace consequence during or after a disaster leading to malfunction of usual communication capabilities [8, 9, 10] (Figure 1.1). The available capacity may not be sufficient to handle whatever data load is placed. Furthermore, it has been observed [11, 1] that in order to cope with anxiety or due to inherent lack of coordination among users, people in a disaster-strike area tend to communicate extensively. For example, distressed people can send the same message or the picture of their damaged house repeatedly, thereby exacerbating the demands placed upon the remaining communication resources, which have already become scarce. Challenged with the lack of communication capacity available during a disaster, disseminating redundant and noisy data could degrade or prohibit the effective operation of response
services as well as could lead to failure in producing a credible picture of disaster aftermath for people residing outside of the disaster area.

Figure 1.1: News article showing power and communication outage in a recent US storm (featured on July 3, 2012).

In recent years, the networking research community has proposed specific protocols that can cope with the challenges of intermittent connectivity. This technology is referred to as Delay- and Disruption-Tolerant Networking (DTN) (www.dtnrg.org). DTN suggests using store-carry-forward protocols where data packets can be physically carried if necessary, holding the packets for long in persistent storage when no communication link exists. For instance, if only one person in a disaster has a phone with a data connection, say 3G, others may use this phone to relay their messages. More generally, if an entire neighborhood lacks connectivity, data can be collected by devices and then physically carried towards a point (even by vehicles) where a communication infrastructure is available.

In this thesis, we consider DTN as the primary means of communication during disaster. The goal of this thesis is to investigate interesting problems that arise in disaster communication scenarios in the DTN networking paradigm and to propose solutions by leveraging available resources and op-
opportunities that prevail during the period. Our goal is to provide solutions to networking problems (e.g., routing data messages across the network) in order to disseminate high quality data (e.g., non-redundant data) so that the network can support a broad variety of services that need to operate during crises. We do not claim that all protocols and services we built are strongly tied to disaster events, it is rather the disaster that provided an application context for various problems and solutions we propose in this thesis. We believe that the results are general and have applications for other contexts too. In the following, we describe a couple of opportunities we plan to investigate.

1.1 Challenges and Opportunities in Disaster Communication

We postulate that mobility-assisted data communication techniques, the one enabled by DTN, would be of high use in a disaster aftermath. Accordingly, we plan to discover a set of key physical and logical properties of existing network elements, mainly devices and data content, that we can leverage in designing our protocols and techniques. One property we identified is recurrence observed in the mobility patterns of mobile agents deployed in a disaster response operation. Another aspect we address is the inherent redundancy in human-generated data content. In the following, we describe these two aspects in more detail.

1.1.1 Intermittent, but Recurrent Connectivity

We argue that in the realities of disaster response efforts initiated by people, rescue workers, and service vehicles, a certain kind of connectivity behavior evolves. This is recurrence. Recurrence means the repetitiveness in movement of nodes. It indicates that for a set of mobile entities moving in space, there could be a certain set of (static) points where nodes (re)visit more frequently than others. Recurrence does exists in many scenarios. In fact, no practical movement is entirely random. Instead, some regularity is observed. For example, in an university campus, students return to classrooms, labs and libraries more repeatedly than other places, say admission office. Working people prone to revisit their workplace and home again and again, along
with quite a few occasional visits to local drug store or movie club. In a disaster-response operation, medical supplies are delivered to evacuation camps, police vehicles patrol given routes, fire trucks originate at fire stations, and volunteers end up at the same gathering points, such as schools used for relief distribution. Hence, a core of moving entities exists that revisits overlapping sets of static places repeatedly. Our communication protocol takes advantage of this core to build a stable routing “backbone” by observing contact occurrences of nodes over time and identifying recurrent contacts.

1.1.2 Redundant Data Content

During disaster, a user behavior becomes prominent: generating excessively redundant and noisy content. When survivors take a picture of their damaged house and send it repeatedly to everyone they know, there is an enormous amount of redundant content created and set to transfer through the network. A similar case (or even worse) can happen when, for example, multiple people take pictures of the same damaged house and send all those pictures to a common service point, whilst sending only a few of them could have been sufficient. Since data objects in DTNs are usually stored at different nodes and data content moves inside the network, redundancy is actually multiplied. This natural behavior aggravates the problem of limited connectivity, causing inefficient utilization of shared communication, storage, and power resources. This calls for sophisticated prioritization techniques and mechanisms to identify and reduce redundant content.

In principle, the redundancy reduction problem arises due to a higher level network design objective. Namely, we postulate that the main objective of the network (i.e., its content dissemination protocol) is to maximize “information” transfer per unit cost. During disaster, people residing in the distressed area are exposed to the same physical environment and condition based on which they generate their content to share among themselves. Since multiple users generate content based on the same physical events, in this context, the volume of generated content is way multiplied than the actual “information” contained therein (for example, the set of all events happened in the area). This phenomenon attributes to redundancy.

Current DTN implementations do not support these capabilities, and we
thus propose to augment DTNs message forwarding and storing functionalities to handle these application-specific requirements. Note that while DTN protocols usually replicate (bit-wise duplicate) data packets onto several nodes to increase delivery chances (thus increases redundancy), our proposed redundancy reduction approach is rather in semantic level and is orthogonal to duplicating packets to overcome poor connectivity. While popular communication styles of today consider delivery of all bits are equally valuable, our paradigm stands in contrast. Forwarding and storing based on content is a new paradigm that calls for methodologies for assigning relative importance to data objects and associated priority assignment techniques.

1.2 Dissertation Statement

We propose the following dissertation statement:

_Leveraging recurrence in motion patterns of nodes and reducing redundancy in human-generated data will result in more efficient content dissemination in overloaded DTNs._

1.3 Contribution of This Thesis

From a boarder perspective, we divide our dissertation efforts into two main threads with the consideration of two fundamental aspects of DTNs: i) mobile nodes and ii) moving content, that is, in DTNs node move and they generate moving content. Accordingly, in order to improve network efficiency and to design effective content dissemination protocols, we need to investigate mechanisms of two main categories: (i) exploiting physical mobility patterns of nodes, and (ii) exploiting logical properties of content generated by mobile nodes. Specifically, we leverage recurrence, as an example of exploiting mobility patterns of nodes, and explore the power of diversity, as an example of exploiting properties of content. Below, we describe these two directions in more details.
1.3.1 DTN Protocols Exploiting Recurrence

We propose a DTN routing protocol that exploits recurrence. To establish the promise of recurrence as an important artifact of mobile networks, let us look closely at mobility of nodes in DTNs. In DTNs, nodes move in space and pose unavailability in communication links both in time and space. The notion of space is important because spatial vicinity in physical space is required for two nodes to communicate and pass data bits. So, in order to utilize intermittently available links for effective data communication, one needs to understand the availability as well as disruption of links as a function of both time and space. But space is a very complex physical artifact to model, which essentially makes the characterization of true spatial behavior of nodes very hard. Arguably, we may not need to model space as it is. Instead, what we do is to model the effects of such spatial behavior on the topological properties of interest (namely, connectivity and delay). Observe that complex spatial behavior models are not always necessary for this purpose. For example, in the architecture community, rather than understanding the exact “mobility patterns” of the program counter (in the space of memory locations), it was possible to achieve significant performance benefits simply by exploiting an overarching behavior principle such as locality of reference, which leads to caching. Similarly, in DTNs, to route packets, we exploit an overarching node behavior principle; namely, recurrence. In that, the network can be construed as a collection of recurrent contacts (i.e., meeting between a pair of nodes) and the associated time gaps between those contacts.

We worked on the following problems in this thread:

- **Inter-contact Routing (ICR):** An energy-efficient multi-copy routing protocol for disaster-response DTNs. ICR takes advantage of recurrent contacts in determining routes for delivering data packets. Chapter 2 describes the protocol at length.

- **Delay bound for recurrent DTNs:** We analyze our proposed DTN model and derive analytical end-to-end delay bound for prioritized flows in recurrent DTNs (Chapter 3).
1.3.2 Diversity Maximization Protocols for DTN Apps

In the second thread of our work, we propose a set of mobility-assisted content dissemination techniques that minimize redundancy in disseminated content. As we argued before, human generated content are generally prone to be redundant due to generation of nearly-duplicate content by multiple parties. For example, in a post-disaster deployment, volunteers may generate reports or capture pictures of certain event (e.g., a burned building) that needs an immediate attention and send those reports/pictures to a command base over a DTN (or share among themselves for better situational awareness).

Assuming DTNs are highly resource constrained in terms of storage and transfer capacity, carrying and delivering all these generated pictures is not possible. Transferring all of these content is not required either. Because, for example, lots of volunteers due to poor coordination among themselves may visit the same event and report a set of “similar” pictures to the base. That means sending at least one from each different group might be sufficient.

Generally, this raises an interesting question as how to reduce the volume of content to be delivered without causing much degradation to the perceived utility at the application. A redundancy minimization technique is the key to answer this. We further recognize that minimizing redundancy of data objects can be regarded as maximizing “diversity” of the same in the sense that the collection retains as much “dissimilar” or “different” (non-redundant) content as possible. In that, redundancy minimization problems are computationally analogous to diversity maximization problems.

To disseminate non-redundant information, the core techniques we use are in-network prioritization schemes. Prioritization techniques identify which content is more useful (non-redundant with respect to the current collection) than others so that data packets can be accordingly transferred or stored inside the network. We develop a set of application-specific content prioritization protocols, termed as content-aware prioritization (CAP), that try to minimize redundancy in content dissemination. These prioritization protocols are mostly optimization problems that are solved to determine the order in which data objects are required to be transferred and stored so that it maximizes a certain utility metric, called diversity measure, computed over the content collection. The techniques also require to eliminate redundant content inside the network (say, dropping multiple pictures of the same burned...
house). In order to do that, it is important (and challenging) to recognize that two or more data objects have redundant content. Recognizing redundancy can be done with data mining techniques such as clustering, depending on the specific type of the data object. We use results from existing literature to handle this part of our work.

![Diagram of network components](image)

Figure 1.2: (a) Content-aware prioritization inside the network. (b) CAP runs at each node.

The diversity-maximizing protocols that we proposed can be applied in numerous different applications. Each protocol has four logical components: i) content representation, ii) distance function between objects, iii) diversity metric, and iv) storage and transfer prioritization policies (CAP rules). Based on certain application requirements, the application developer needs to specify an appropriate object representation and a suitable distance function. They also need to choose one from a set of proposed diversity metrics. Once the diversity metric is chosen, prioritization rules are derived from the optimization formulation.

While choosing an appropriate diversity metric for an application, we further identified that diversity maximization becomes tricky when content has noises and outliers, because outliers are naturally favored by the scheme. Since applications, such as situation awareness, are sometimes participatory in nature and the participating sources may have different degree of reliability in generating content, noisy content can be introduced. If the application requires special attention to outliers, the associated diversity metric needs to be revisited by incorporating associated handling cases for outliers based on the application context.

We develop the following three protocols:

- **PhotoNet:** First we propose PhotoNet, an application service that
collects pictures from a disaster area to maximize (mainly) geographic coverage for better situational awareness. It maximizes diversity of picture collection by favoring pictures from distant locations and pictures that are visually different pictures but originated from the same location. We describe PhotoNet in Chapter 4.

- **PhotoNet**: It turned out that PhotoNet is susceptible to noises and outliers. To fix that, we propose PhotoNet+, which attempts to eliminate outliers from the collection. PhotoNet+ is described in Chapter 5.

- **Diversity Caching**: We extend our redundancy reduction techniques for DTN caching policies. We propose a distributed data caching system that consists of a bunch of caches at different network locations where data objects are stored and user queries are responded. We describe this problem in Chapter 6.

### 1.3.3 Disaster Mobility Model and Disaster DTNs

Current DTN literature lacks a suitable mobility model as well as suitable real traces that capture movements of agents during a disaster aftermath. We propose a novel mobility model, Post-Disaster Mobility (PDM), comprising of different mobile agents with different roles in an urban disaster area (Chapter 7). In not otherwise stated, all our experiments are conducted using this mobility model.

### 1.4 Organization of the Thesis

The next chapter (Chapter 2) details our proposed inter-contact routing protocol followed by its delay analysis in Chapter 3. Chapter 4 introduces our redundancy minimization protocols for media content as well as describes one of schemes, PhotoNet. Chapter 5 describes PhotoNet+. We describe diversity caching techniques in Chapter 6. Chapter 7 outlines the post-disaster mobility model we develop for conduct our simulation experiments. Chapter 8 enumerates recent literature related to work presented in this thesis. Chapter 9 concludes the thesis with closing remarks and a set of future research directions.
CHAPTER 2

INTER-CONTACT ROUTING PROTOCOL

In this chapter, we describe an energy-efficient multi-copy routing protocol for disaster-response networks that exploits recurrence to achieve improved resource economy \(^1\). In the aftermath of a natural disaster (such as a hurricane or a strong earthquake), regular communication services are disrupted due to infrastructure damage and power outages. Disaster-response networks are thus mainly DTNs comprising of battery-operated wireless devices carried by people (survivors, rescue workers and volunteers). These devices could be cell-phones in an ad hoc communication mode and wireless routers powered by vehicular batteries (may be at households or in vehicles). A key performance objective that we investigate in designing our routing protocol is to minimize energy consumption per message in order to prolong the lifetime of battery-operated devices until infrastructure is restored. There are many ways energy-economy can be achieved. For example, nodes can be duty-cycled, low-power listening modes can be used, and transmission energy can be modulated depending on channel conditions and distance from receiver. Our work focuses only on the routing layer. Since the main “knobs” under the control of multi-copy routing are the number of message copies made and the path of each message, we consider the total number of message replica transmissions in the network as a quantification of energy-efficiency of routing decisions. In that sense, routing protocols proposed for general DTNs are often not suitable for disaster-response networks, as they concern themselves more with other performance objectives. For example, they often achieve a high delivery ratio at the expense of generous message copying and

\(^1\)The work presented in this chapter has been published in two research articles:
forwarding which consumes significant amounts of communication energy.

In contrast, our protocol significantly reduces the need for large numbers of copies by discovering recurrent contacts and forming a network view (a routing table) that uses such recurrent contacts to deliver a message (possibly across multiple hops) to the destination. By exploiting recurrence to increase delivery probability, the degree of message replication is reduced and the number of message transmissions drops. Hence, the protocol contributes to saving energy while maintaining a high delivery ratio.

We argue that recurrence exists in disaster-response networks because movement of entities in disaster scenes is not entirely random. Instead, some regularity is observed: medical supplies are delivered to evacuation camps, police vehicles patrol given routes, fire trucks originate at fire stations, and volunteers end up at the same gathering points, such as schools used for relief operation management. Furthermore, a number of static points exist that can be used for message handover. For example, the evacuation camps, schools, fire stations, and vehicles stranded on common routes may all serve as stepping stones to deliver data from one moving entity to the next. Hence, a core of moving entities exists that revisits overlapping sets of static places repeatedly. Our routing protocol takes advantage of this core to build a stable routing “backbone” by observing contact occurrences of nodes over time and identifying recurrent contacts.

The protocol uses the new notion of inter-contact routing to uncover patterns that reduce the DTN to a multihop data-muling network. Inter-contact routing (ICR) allows more accurate estimation of delivery delay and delivery probability, identifies reliable multi-mule paths when they exist, and informs message replication and forwarding decisions. Since each relief operation is unique, the identities of data mules and their sources and destinations are learned rather than hardcoded into the protocol. Evaluation shows that the protocol is effective at reducing the total number of message transmissions in the network while maintaining a high delivery ratio and only a slightly larger delay compared to other DTN protocols in recent literature.

ICR offers low-energy operation by creating fewer number of replicas per message. In addition to the base multi-copy replication scheme, we augment the protocol with an extension, namely energy-differentiated service, that allows DTN nodes to manage their energy consumption more economically depending on the available (low) energy and the relative urgency of
messages. The proposed service classifies messages into two priority classes, namely ‘urgent’ and ‘regular’. In addition, we propose a differentiated energy allocation scheme that accounts for energy usage of respective traffic and reserves energy for more important traffic whenever required. Inter-contact routing along with this service achieves a commendable performance in an ultra-low energy condition, where other traditional routing protocols drastically fail.

The rest of this chapter is organized as follows. The following Section 2.1 introduces inter-contact routing followed by Section 2.2 where we describe energy-differentiated service based upon ICR. Section 2.3 describes a novel mobility model that enables simulation of a post-disaster scenarios. Section 2.4 describes evaluation results and comparison with other known protocols. Section 2.5 concludes the chapter with some feature research directions.

2.1 Low-energy Inter-contact Routing (ICR)

We describe Inter-contact Routing (ICR), which improves energy efficiency by forwarding fewer messages when paths have a higher probability of delivery. The protocol exploits information about different paths, when available, to estimate message delivery probability. It then forwards messages on the most reliable paths to eliminate unnecessary message replication. When no information is available it defaults to a prior state of the art protocol Spray-and-Wait [12]. Therefore, the resulting overall number of message transmissions and energy consumption are decreased, when possible, while keeping the delivery ratio high.

There are two key challenges associated with opportunistic exploitation of paths with a higher delivery probability; namely, (i) what information to maintain in routing tables to estimate delivery probability, and (ii) how to make packet replication and forwarding decisions based on that information (as well as what to do when no information is available to tell how to reach a destination).

To address the first challenge (estimation of delivery probability), we observe that since nodes move in a finite space, under broad mobility assumptions, all packets will be delivered given an infinite amount of time. To obtain a more meaningful metric, we define successful delivery as one that occurs
within a given latency bound, called the delivery timeout. If delay extends beyond that timeout, delivery is said to have failed. Nodes gather statistics of their contacts with other nodes. In particular, inter-contact delay statistics are collected. Flooding these data through the network makes it possible to add up inter-contact delays along paths and compute path delay distributions and hence delivery probability within the timeout.

To address the second challenge (making replication decisions), we set a virtual number of copies for each message that bounds the number of actual copies which can be created for it. A similar idea has been used in Spray-and-Wait. Using delivery probabilities contained in our routing information, we partition the virtual copies, putting more copies on routes with a higher confidence. In the absence of information on a destination, we resort to a blind spray of messages.

In the rest of this section, we shall first detail our disruption-tolerant network model (Section 2.1.1), then elaborate on the aforementioned two challenges (Section 2.1.2).

2.1.1 A Novel Network Model for DTNs

Each node in our protocol maintains a set of neighbors, which are nodes that it encounters recurrently. Encounters with a node are considered recurrent if they occur more often than a given number of times within a specified time window. Intermediate nodes along a message’s path hold the message from an encounter with the previous node to an encounter with the next along the path to the destination.

In a classical network, node delay depends only on the conditions of outgoing links. Packets are buffered until such links become available. In particular, the delay does not depend on which input port a packet came from. DTNs are fundamentally different in that the delay a packet experiences at a node depends not only on the next node the packet is forwarded to, but also on the previous node it came from. It is the delay between the above two contacts (i.e., the inter-contact delay) that determines the packet’s local residence time. Our network model is inspired by that observation. Hence, rather than thinking in terms of encounter graphs, where each vertex corresponds to a node and edges connect nodes that have frequent encounters, we
use an *inter-contact graph*, where a vertex represents an encounter between two nodes. An edge between two vertices represents the delay between the (beginnings of) two encounters. Each vertex is labeled with the corresponding node names, such as $ij$ for an encounter between node $i$ and node $j$ (Figure 2.1). For example, consider a police patrol, $i$, that makes 30 minute rounds along some city loop. The patrol passes building $j$ then building $k$ at two successive street intersections, approximately 3 minutes apart. Given the above parameters, it will be 27 minutes after seeing $k$ that the patrol sees $j$ again. Hence, a directed edge of delay 3 minutes exists in the inter-contact graph between vertices $ij$ and $ik$. Similarly, a directed edge of delay 27 minutes exists between vertices $ik$ and $ij$. This asymmetry explains why edges in the inter-contact graph are directed.

The novelty of inter-contact graphs lies in capturing the dependencies of delay on both the previous and next hops on the path of a message in a DTN. The delay asymmetry mentioned above is not captured in a regular encounter graph that plots the period or frequency of encounters between individual nodes. In such a graph, since node $i$ sees each of $j$ and $k$ every half hour, an edge of delay 30 minutes might exist between $i$ and each of $j$ and $k$. When $i$ encounters $j$, it might seem that sending a message from $j$ to $k$ via $i$ would take 60 minutes, whereas in fact it takes only 3 minutes, as captured in our inter-contact graph. Figure 2.1 illustrates the above mobility scenario together with its representation in an encounter graph and an inter-contact graph.

In the following discussion, to avoid ambiguity, we shall use the term *node* to refer to a physical device capable of receiving and forwarding messages in the network. We shall use the term *vertex* to refer to a vertex in the inter-contact graph that represents an encounter between two nodes. We use the notation, $c_1 \rightarrow c_2$ to denote a directed edge between two contacts (i.e., vertices) $c_1$ and $c_2$. Each edge, $ij \rightarrow ik$, in the inter-contact graph

![Figure 2.1: A scenario showing encounter and inter-contact graph.](image-url)
maintained is annotated by a tuple of two values, \((\delta(ij \rightarrow ik), \sigma^2(ij \rightarrow ik))\), where \(\delta(ij \rightarrow ik)\) is the average delay elapsed on node \(i\) between meeting node \(j\) and node \(k\), and \(\sigma^2(ij \rightarrow ik)\) is the corresponding delay variance. We denote a path in the inter-contact graph as \(\text{contact} \leadsto \text{node}\). For example, \(ij \leadsto w\), is a path from contact \(ij\) to (some contact with) node \(w\). Due to the linearity of expectation and under the assumption of edge independence, we can define path delay, denoted \(d(ij \leadsto w)\), and path variance, denoted \(\sigma^2(ij \leadsto w)\), as the sums of the edge delays and variances along the path. If there are multiple paths from an initial contact to the destination, the parameters of the ‘best’ path will be stored for routing purposes. Since, in general, not all nodes have the same view of the network, it is useful to define path delay and variance from the perspective of a given node. Hence, we define \(d_i(ij \leadsto w)\) and \(\sigma_i^2(ij \leadsto w)\) to denote path delay and variance computed by node \(i\).

### 2.1.2 The Routing Algorithm

As observed above, DTNs are different from regular networks in that message delay at a node depends not only on the node downstream but also on the one upstream. A routing table in a typical (e.g., wired) network has one entry per destination (or destination mask). In our algorithm, the entry for destination \(w\) at vertex \(ij\) in the inter-contact graph contains the corresponding path delay, \(d_i(ij \leadsto w)\), and variance, \(\sigma_i^2(ij \leadsto w)\). We store both values as a parametric summary of the delay distribution along the optimal route, as seen by node \(i\). The delay distribution determines the message delivery probability (by a given timeout).

A question arises as to why not store the path delivery probability directly in the routing table. The problem lies in that it is message-dependent. For example, if the network timeout is 6 hours, and a message arriving at a node has already been en route for two hours, successful delivery refers to delivering this message in the next four hours. Estimating the probability of successful delivery in four hours requires knowledge of the delay distribution, which we estimate from a mean and variance. Hence, by storing the mean and variance of the delay distribution of a path, we can individually compute the different delivery probabilities for different messages.
A more subtle question lies in the notion of path optimality. How to choose the path whose delay and variance should be stored in the routing table entry for a given destination? The typical answer is, we should store the parameters of the optimal path. However, as we just mentioned, the delivery probability depends on the message, as it depends on how long the message has already spent en route. Hence, an optimal message-parameter-independent path cannot be established. Instead, to determine the path whose parameters to store in the routing table, we come up with an alternative message-independent path cost defined as the 95th-percentile of path delay, or $\text{cost} = d_i(ij \rightsquigarrow w) + 1.65 \times \sqrt{\sigma^2_i(ij \rightsquigarrow w)}$. This metric favors paths with an appropriately low delay and delay variance, hopefully leading to a higher delivery probability (within the timeout).

Hence, our routing protocol stores the delay and variance of the minimum cost path to each known destination as defined by the above message-independent cost metric. A separate routing table is constructed at each node $i$ for each vertex $ij$ (i.e., for each vertex in the inter-contact graph that refers to an encounter of node $i$ with one of its neighbors, $j$). The table for vertex $ij$ at node $i$ has an entry per destination $w$ (known to $i$), storing $d_i(ij \rightsquigarrow w)$ and $\sigma^2_i(ij \rightsquigarrow w)$ of the optimal path cost to that destination, should the message be forwarded via the encountered node, $j$. Observe that, a vertex $ij$ in the inter-contact graph is therefore associated with two routing tables; one maintained at node $i$ and one at $j$.

The Inter-contact Delay Table

In a typical connected network, all outgoing links are available simultaneously. The situation is different in DTNs in that we must account for inter-contact delays. We use a second table, called the inter-contact delay table, to store the average inter-contact delay and delay variance between contacts of the node in question with every two of its neighbors. Hence, this table contains one entry per neighbor pair. For simplicity, we ignore the time it actually takes to forward a message to another node once the contact is made. Typically, such delay is orders of magnitude lower than the inter-contact delay, which may be of the order to tens of minutes or hours. For the same reason, we consider contacts to be points in time (rather than time intervals). If a contact lasts for a long duration, the relevant point in time is the
beginning of such interval, as that is when buffered messages are exchanged.

When node $i$ meets a neighbor $k$, it computes $x(ij \to ik)$, the delay between contacts $ij$ and $ik$, for each neighbor $j$ seen more recently than the previous encounter with $k$. Let the current time be $T_{now}$, the last encounter time with $j$ be $T_i^j$ and the last encounter time with $k$ be $T_i^k$. Node $i$ computes the individual sample $x(ij \to ik)$, as well as the mean and variance as follows:

$$
\begin{align*}
x(ij \to ik) &= T_{now} - T_i^j, \text{if } T_i^k < T_i^j \\
\delta(ij \to ik) &= \bar{x}(ij \to ik) \\
\sigma^2(ij \to ik) &= S_n^2 = \frac{\sum(x - \bar{x})^2}{n - 1}
\end{align*}
$$

where $n$ is the number of samples in the summation and $S_n$ is the standard deviation of these samples. Due to central limit theorem, provided that each inter-contact delay is computed over enough samples, we can assume that the delay is normally distributed with mean, $\delta$ and variance, $\sigma^2$.

Obviously, unlike traditional link state tables, inter-contact delay table size is in the order of the square of the number of neighbors instead of linear of neighbor count. Since these tables are locally maintained at an individual node and are not shared among nodes, they do not incorporate much overhead other than some storage need.

Routing Table Updates

Routing table updates occur in a distance-vector manner. When two nodes $i$ and $j$ meet, they update each other’s routing tables for vertex $ij$. Consider node $j$ that meets node $i$. Node $j$ recomputes its optimal paths to each of the destinations known to $j$ and shares the result with $i$ to store in $i$’s routing table for vertex $ij$ (which describes the path costs via $j$). The path delay and variance from vertex $ij$ to each destination $w$ is computed by considering the set of $j$’s neighbors, $N_j$. For every neighbor $l \in N_j$, the mean delay, variance and corresponding cost are first computed. The optimal cost is then found:
\[
\begin{align*}
\text{delay}_l &= \delta(ji \rightarrow jl) + d_j(jl \sim w) \\
\text{var}_l &= \sigma^2(ji \rightarrow jl) + \sigma_j^2(jl \sim w) \\
\text{cost}_l &= \text{delay}_l + 1.65 \times \sqrt{\text{var}_l} \\
l^* &= \operatorname*{arg\,min}_{l \in \mathcal{N}_j, l \neq i \text{ cost}_l
\end{align*}
\]

Node \(j\) then sends \(\text{delay}_{l^*}\) and \(\text{var}_{l^*}\) to \(i\) as the mean delay and variance of the optimal path to destination \(w\) via \(j\) (for each destination \(w\)). The routing table for vertex \(ij\) at node \(i\) is updated, that is, \(d(ij \sim w) = \text{delay}_{l^*}\) and \(\sigma^2(ij \sim w) = \text{var}_{l^*}\). Similarly, node \(i\) updates the routing table for node \(j\). In principle, the above computation can be done in advance of the actual encounter. We use a “lazy” approach, since we might not encounter a node for a long time (and need not precompute its table entries), and because the inter-contact delays (used in the above computation) are updated upon each encounter, making previous results stale anyway. The path delay is a sum of inter-contact delays and inter-contacts are independent. So, the path delay can also be assumed to be normally distributed with the associated mean and variance. Since paths are usually short (2-4 hops), we store intermediate nodes with each entry. This avoids loop in paths. We can also try sequencing as in destination sequenced distant vector (DSDV) to avoid loops.

The above parameters are then used at node \(i\) to compute the message-dependent delivery probabilities that decide if any of its messages are to be forwarded to \(j\), and how many copies should be forwarded, as described below.

Forwarding and Message Replication

Every message in our network starts with a certain number of initial message copies from its source and an original timeout value, which defines a time-frame for successful delivery. A remaining time-to-live (TTL) field is initialized to the original timeout. When a node \(i\) that has copies of a message meets the destination, the message is delivered. When the node meets one other than the destination, it may forward some copies of the message to the encountered node. Obviously, the same message is not sent repeatedly,
but rather sent once. The intended number of copies it represents is carried in its header. Suppose, node \( i \) has a message with \( L \) copies to deliver to \( w \), and it meets its neighbor \( j \). The node updates the TTL field of the message (by subtracting the time since the last update). It then computes the delivery probability \( p_k \) via each one of its neighbors, \( k \) (including \( k = j \)), as follows:

\[
p_k = P\{0 < \text{delay} \leq \text{TTL}|\text{delay} > 0\} = \Phi\left(\frac{\text{TTL} - \text{delay}_k}{\sqrt{\text{var}_k}}\right) - \Phi\left(-\frac{\text{delay}_k}{\sqrt{\text{var}_k}}\right)
\]

(2.1)

where \( \Phi(.) \) is the CDF of normal distribution, and:

\[
\text{delay}_k = \delta(ij \rightarrow ik) + d_i(ik \rightarrow w)
\]

(2.2)

\[
\text{var}_k = \sigma^2(ij \rightarrow ik) + \sigma_i^2(ik \rightarrow w)
\]

(2.3)

Neighbors are then logically ordered by \( p_k \) in descending order. Each neighbor \( k \), in that order, is allotted \( p_kL \) copies and these allotments are subtracted from \( L \). The process continues until \( L \) runs out or until all neighbors have been considered. If all neighbors have been considered yet some copies remain, it must be that the probabilities \( p_k \) were low. The remaining copies are sprayed, which means half of them are given to the encountered node, \( j \). Hence, there are three possibilities with regard to node \( j \):

- \( L \) runs out before we disburse \( j \)'s allotment. This means there are sufficiently better nodes \( k \) ahead of \( j \) in the ordered list by \( p_k \). No copies are forwarded to \( j \).

- \( L \) runs out when or after we disburse \( j \)'s allotment. In this case, \( j \) gets \( p_jL \) copies (or the remainder of \( L \) if less than \( p_jL \))

- \( L \) does not run out even after all neighbors are considered. In this case, \( j \) gets \( p_jL \) copies plus half the remainder of \( L \).

When the message is forwarded, its TTL is current. The receiver notes the time it arrived and uses that timestamp to update the TTL of the message.
later by subtracting local residence time. The above forwarding algorithm has some interesting properties:

- If \( p_j = 1 \): The encountered node \( j \) takes all copies of the message. The case \( p_j = 1 \) can happen only if the variance along the path is extremely low indicating a very stable path. In that case, this path alone carries all copies of the message and no other paths are tried. No physical message replication occurs.

- All \( p_k = 0 \): This is the case by default in the beginning when no information is known about any destination, or in the case when no recurrence has been observed. In this case, our protocols defaults to the Spray-and-Wait protocol, allowing node \( i \) to forward half of the message copies to each encountered node \( j \).

- \( 0 < p_k < 1 \): In this case, the protocol opportunistically exploits paths to the degree to which they are recurrent, hence reducing the number of copies needed compare to Spray-and-Wait, and achieving better resource economy commensurately with the degree of recurrence in the network.

Pruning Low-confidence Links and Bad Neighbors

It remains to show how a node decides on which neighbors and links to consider in routing decisions. When only a very small number of encounters were observed with others, the confidence in the computed mean delay and delay variance could be very low. Such low-confidence links should not be part of the inter-contact graph. They are identified and eliminated. Specifically, let the real mean of the inter-contact delay of a link be denoted by \( \mu \), where \( \mu = E[\delta(ij \rightarrow ik)] \). Under fairly broad assumptions, we can determine how close the computed \( \delta(ij \rightarrow ik) \) is to the real mean \( \mu \) by applying the central limit theorem:

\[
P \left\{ \mu \in \left[ \mu \pm Z_\alpha \frac{\sigma}{\sqrt{n}} \right] \right\} \approx 1 - 2\alpha
\]

where \( Z_\alpha \) is the critical value such that \( \Phi(Z_\alpha) = 1 - \alpha \), \( \Phi(Z) \) is the CDF of standard normal distribution, defined as \( \Phi(Z) = \int_{-\infty}^{Z} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt \). We
can now compute a confidence value \( \gamma(ij \rightarrow ik) \), representing the probability that the estimated average delay is within some bound \( \tau \) from the true mean by setting \( \tau = Z_\alpha \frac{\bar{x}}{\sqrt{n}} \) above. Hence, \( Z_\alpha = \tau \frac{\bar{x}}{S_n} \). Therefore:

\[
\gamma(ij \rightarrow ik) = P\{\mu \in [\bar{x} \pm \tau]\} = 1 - 2\alpha = 2\Phi\left(\frac{\tau \sqrt{n}}{S_n}\right) - 1 \quad (2.4)
\]

Hence, given the interval \( \tau \), we can use the above equation to find the probability that the measured sample mean is within that interval from the true process mean. Since ultimately, we are not interested in accurate delay measurements, but rather in rough assurances of bounded delivery, we allow for a considerably large interval (nearly the same width as the sample mean). We use \( \tau = 30 \) minutes and the constraint \( \gamma > 0.75 \). Otherwise the link is dropped as untrusted.

Observe that since link delay and variance are updated only upon encounters with a node, a neighbor that disappears leaves state that may become stale. Hence, if the last encounter with a neighbor becomes “too old”, confidence in inter-contact delays involving that neighbors is exponentially decreased. The corresponding links are eventually dropped per rules above. A node with no links is dropped from the tables.

A Numeric Example

Let us consider an example to illustrate the routing protocol. Figure 2.2

Figure 2.2: Node \( i \) is deciding message copies for \( j \).

Let us consider an example to illustrate the routing protocol. Figure 2.2
shows the inter-contact table of node $i$ and the routing entry for destination $w$ via its three neighbors $j$, $k$ and $m$ (entry contains $(\mu, \sigma^2)$ of the associated delay in minutes). Let node $i$ have a message with 20 copies to deliver to $w$ within the delivery timeout of 100 minutes. Node $i$ meets $j$. We are to compute delivery probabilities $p_j$, $p_k$ and $p_m$. Path delay and variance via $j$ are given in the routing table, $(d_j, \sigma^2_j) = (92, 24^2)$. Similarly, $(d_k, \sigma^2_k) = (5+65, 4^2+30^2) = (70, 30.26^2)$ and $(d_m, \sigma^2_m) = (3+85, 4^2+94^2) = (88, 94.08^2)$ are obtained by adding the inter-contact delay with the routing table entry. By using Equation 2.1, $p_j = \Phi\left(\frac{100-92}{24}\right) - \Phi\left(-\frac{92}{24}\right) = 0.63$, similarly, $p_k = 0.84$ and $p_m = 0.46$. Node $k$ gets $p_k L = 0.84 \times 20 = 17$ copies first, since it offers the highest delivery probability. Node $j$ receives $0.63 \times 20 = 12$ or $20 - 17 = 3$, whichever is smaller. Hence, 3 message copies are forwarded to $j$.

### An Energy-differentiated Service for ICR

Energy is a vital resource in DTNs. Different devices in the network may have different amounts of available energy. For example, rescue centers and vehicles may have access to comparatively abundant energy sources (e.g., generators or running engines), so they are not energy constrained as far as computation and communication goes. Handheld devices (say, with rescue workers) and routers at homes (e.g., WiFi capabilities on user cell-phones) are more energy constrained and they may not have a chance to recharge their batteries. If energy is not properly regulated, nodes may die prematurely, reducing coverage and impeding the recovery mission. In the evaluation, unless otherwise stated, we consider an expected mission lifetime of 48 hours. Beyond 48 hours, the odds of finding survivors are diminished, for example, when survivors are injured, exposed to severe weather, or have no access to water.

Although ICR saves energy by keeping the number of replicas per message small, further energy economy can be achieved by *blocking* unnecessary messages when nodes determine that they are overspending their energy budget. Not all messages are equally important. Calling for help may be more critical than using the network to communicate with loved ones. Based on this observation, we propose an energy-differentiated message delivery service that treats messages differently based on their urgency and the availability of
energy at forwarding nodes.

2.2.1 Energy Model and Protocol

We classify messages into two types: urgent and regular. The network gives priority to urgent messages over regular ones in that it always attempts to deliver urgent messages, whereas delivery of regular messages is governed by energy availability.

The differentiated service puts a cap on energy expended by regular messages in the form of a maximum energy depletion rate $\lambda$, expressed in Joule per hour. The rate $\lambda$ specifies how much energy a node can expend on delivery of regular messages per hour. No such constraint is imposed on urgent messages. Upon a contact, urgent messages in each node’s buffer are transferred first (as ordered by ICR), followed by regular messages until the allocated energy for regular messages is exhausted, all messages are sent, or the contact terminates, whichever occurs first. By setting an appropriate value for $\lambda$, the network reserves energy for urgent messages by policing regular ones, which in turn increases urgent message delivery rate at the expense of slightly decreasing delivery of regular messages.

We use a simple energy model to account for energy expenditure of message sending and receiving (for brevity, we lump read/write energy for accessing the storage device into the corresponding radio operations—reading into sending and writing into receiving). We also consider energy dissipation due to idling when nodes are not in communication with others. The model works as follows. Every node starts with an initial total energy, $E_{init}$, that is specific to that node. It paces the consumption of that energy by computing an hourly expenditure that can be sustained throughout its desired lifetime (e.g., 48 hours). The energy balance available for regular messages is updated it every hour according to the formula:

$$E_i = E_{i-1} + \lambda$$

(2.5)

where $E_i$ is the accumulated energy available for regular messages by the $i$th hour and $\lambda$ is the hourly energy budget. Whenever a node receives or relays messages, the above energy balance is reduced by the amount expended by communication. Energy also gets depleted due to idling. In our system, the
total idling energy consumption over the planned lifetime is simply subtracted from the initial budget at the start. Hence, $E_i$ is the remaining amount of energy available solely for communication of regular messages. The following table gives a snippet of energy update rules.

<table>
<thead>
<tr>
<th>At the beginning:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_0 \leftarrow 0$;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>In every hour $i \ (\geq 1)$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_i \leftarrow E_{i-1} + \lambda$;</td>
</tr>
</tbody>
</table>

Receiving/transferring message $m$:

$e_{req} = m\text{.size} \times \text{energy per byte}$;

if ($m\text{.type} = \text{‘urgent’}$)

Receive/transfer $m$;

if ($m\text{.type} = \text{‘regular’}$ and $E_i > 0$)

$E_i \leftarrow E_{i-1} - e_{req}$;

Receive/transfer $m$;

Else

Ignore $m$; /* No more allocated energy */

2.2.2 Enforcement Issues

We assume that urgent messages are considerably fewer than regular messages. If all messages are declared as urgent, they will all compete for the limited resources, and eventually, service differentiation will fail. Hence, without priority enforcement, the differentiated service may not function properly. There can be two possible solutions to this problem: designated destinations and source authentication.

In the “designated destinations” version of enforcement, urgent messages can only be issued to a particular set of designated destinations, similar to 911 calls. Anyone can send urgent messages, but only to this set of pre-specified destinations. Messages targeted to other destinations will not be treated as urgent but regular.

In contrast, the “source authentication” approach relies on cryptographic protection. In that case, only pre-authorized nodes (e.g., police) can issue
urgent messages. We can assume that police officers share a common private key and the corresponding public key is made available to all other nodes. Key distribution may occur before the disaster strikes, because in many cases the impending advent of a disaster can be predicted in advance (e.g., in the case of a hurricane). Only authenticated messages get priority.

2.3 A Post-Disaster Mobility Model

In this section, we briefly highlight the main aspects of the mobility model that we use to evaluate our protocol in a disaster-response setup. Abstract mobility models, such as the classical Random Waypoint Model (RWP) and the Map-Based Random Walk Model (MBM) (a variant of Random Walk on a map), do not capture well the recurrence inherent in mobility patterns of people, objects and activities in post-disaster scenarios. In contrast, the mobility model we use, named the post-disaster mobility model (PDM), attempts to mimic the aftermath of disasters such as hurricanes or large earthquakes. While we defer the details of PDM to Chapter 7, here we mention only the key aspects of the proposed mobility model.

In the spirit of abstraction and simplification, which has been the virtue of several widely-used mobility models in current literature, we do not attempt to model the disaster to a high degree of realism, but rather create a recognizable abstraction of the disaster, focusing on key aspects whose effects and ramifications we want to explore. In this case, the key aspect is the existence of some degree of recurrence in modeled activities. For example, consider a Hurricane site in the aftermath of the disaster. Coordination centers and relief camps are established nearby. Vehicles may carry supplies between the coordination centers and evacuation camps. A good number of rescue workers and volunteers are engaged to locate survivors and offer help. A few emergency vehicles may run between relief centers and distressed neighborhoods. Police officers may patrol neighborhoods to prevent unwanted activities.

We have implemented PDM as an extension to the ONE simulator on top of ONE's Map-Based Mobility Model (MBM). MBM uses map data of roads and streets instead of a flat Euclidean 2-D space. We added four movement classes; the InterCenterClass (recurrent motion of supplies between centers), the RescueWorkerClass (localized random motion visiting various
locations), the PolicePatrolClass (a recurrently patroled path through multiple neighborhoods) and EmergencyClass (dispatch of vehicles from a center to a random destination and back). The final class was the class of fixed nodes including centers, households, and stranded vehicles. We used a small city map (that comes with ONE) to describe an urban disaster area with centers, neighborhoods, houses, people, vehicles, rescue workers, and police patrols. Figure 2.3 depicts a post-disaster scene. We assume that the nodes above (vehicles, rescuers, centers) are equipped with wireless networking devices that can act as ‘routers’. We treat all people inside evacuation camps and other gathering areas as attached to the access points in those areas. We first ignore the effect of energy constraints. Later, an amount of energy is given to each node that corresponds to its role.

2.4 Evaluation and Simulation Results

In the following subsections, we first highlight the simulator settings for our experiments, then evaluate three aspects of ICR: i) its performance compared
Figure 2.4: Performance comparison, ICR, Spray-and-Wait, Spray-and-Focus, MaxProp and Prophet.

to other protocols in different scenarios, ii) its efficacy at energy saving, and iii) the impact of its parameter tuning on performance.

2.4.1 Simulator Settings

We simulate ICR in the ONE (Opportunistic Network Environment) [13] simulator on top of the post-disaster mobility model that we developed and outlined in Section 2.3. All experiments, unless otherwise stated, are done with 5 neighborhoods, 2–5 main centers, 10 relief and evacuation camps, 4–10 supply vehicles, 60–100 rescue workers, 5 police patrols, 5 emergency vehicles, and 100–200 distressed households (although the total number of houses can be much larger). Data traffic is generated as a Poisson process at a rate of 1 message per 5 minutes. For ICR settings, we use confidence threshold = 0.75 and message delivery timeout = 6 hours. Router’s buffer size is 10MB. The simulation duration is 48 hours.

For energy experiments, we use energy profiles based on Android plat-
forms, as measured by PowerTutor [14], an online energy measuring tool. We modified the idle current in anticipation of phone models with more efficient idle modes when communication with a cell-tower is not expected. The used power values are given below:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>200 mW</td>
</tr>
<tr>
<td>Transmit</td>
<td>950 mW</td>
</tr>
<tr>
<td>Receive</td>
<td>950 mW</td>
</tr>
<tr>
<td>Idle</td>
<td>0.6 mW</td>
</tr>
</tbody>
</table>

More specifically, Android devices [14] use 55mW idling power. We believe that this is attributed to inherent inefficiencies in cell-phone design mandated by the need to receive incoming communication from remote cell towers. When these are down, significant portions of the idle energy can be saved by resorting to low power modes such as those described for wireless sensor platforms. For example, MICAZ mote radios provide idle power as low as $20\mu$A [15]. Hence, for our experiments, we use 0.6 mW ($\approx 150\mu$A at 4.0 V) as the idling power.

2.4.2 Performance of ICR

In this section, we consider general performance and efficiency metrics that apply to many routing protocols, energy conserving or not. Evaluation of energy expenditure is delegated to the next section. Specifically, we are interested below in three performance metrics: i) message overhead, ii) delivery ratio and iii) average message delivery delay. Message overhead is computed as the total number of one-hop message transmissions divided by the total number of created messages that are transmitted at least once. We use messages created instead of messages delivered to tease apart the overhead and the delivery ratio. The delivery ratio gives the fraction of total messages delivered to total messages created. For those messages that are delivered, message delay measures the average amount of time a message spends in the network.

It should be noted that the experiments presented in this section do not show effects of resource constraints. The system is intentionally set up to have sufficient resources not to confuse message delivery failures due to resource constraints with delivery failures because of poor routing decisions that do
not adequately exploit mobility and recurrence patterns. Hence, we first
evaluate protocol overhead as well as delivery ratio (that reflects the efficacy
of protocols at the exploitation of mobility and recurrence). In a later section,
we show how energy-constraints lead to a drop in the number of delivered
messages in protocols that have more overhead.

Figure 2.4 compares our performance to that of four other DTN rout-
ing protocols, Spray-and-Wait [12] (Spray), Spray-and-Focus [16] (Focus),
Prophet [17] and MaxProp [18]. In these experiments, we assume that
nodes are not energy constrained. We see from Figure 2.4(a) that ICR gives
the smallest overhead among all protocols, whereas MaxProp, an epidemic
scheme, has the highest overhead. A similarly large overhead is also ob-
served for Prophet (another flooding-based scheme). MaxProp and Prophet,
in contrast, give the lowest message delay (Figure 2.4(c)) and the best de-
livery ratio (Figure 2.4(d)). One should conclude that, in the absence of
resource constraints, there is no need to sacrifice delivery ratio or delay in
order to reduce overhead. Indeed, in terms of performance metrics of interest
to the application, an epidemic scheme such as MaxProp or Prophet is the
best. The advantage of our reduced-overhead scheme will be demonstrated
later, when we consider energy-constrained environments, which is the target
for our design.

One can also notice from Figure 2.4(a), Figure 2.4(c), and Figure 2.4(d)
that Spray-and-Focus is consistently worse than ICR in all three performance
metrics. Hence, it can be safely dropped from further evaluation.

![Comparison with RWP and City Scenario.](image)

Figure 2.5: Comparison with RWP and City Scenario.
In Figure 2.5, we compare the performance of the above protocols (except Focus) in scenarios other than disaster. We used MapBased Random Waypoint (RWP) mobility model and a city mobility model (pertains to the ONE simulator) with cars, buses, and pedestrians. We normalized all performance metrics (overhead, delivery, delay) to those of ICR. Despite the fact that ICR is tailored for disaster response scenarios with recurrent mobility patterns, it is shown to perform well in the other scenarios as well, particularly in terms of overhead, compared to other protocols.

Figure 2.6: Performance of ICR in different mobility scenarios and traffic patterns.
Figure 2.6 explores ICR performance as the network configuration changes (such as the number of participants of different types). We consider houses, main centers, supply vehicles, rescue workers, and police patrols. Since flooding based approaches, namely Prophet and MaxProp, have clear lead in performance over quota-based protocols under no resource constraints (as we observe in Figure 2.4), here for the successive experiments, we limit our comparison with Spray only. Allowing ICR to be compared in various different settings, we compare Spray at the setting when the former performs best/worst as well as at the base setting. We observe that when we change the number of nodes keeping their movement pattern the same, the overhead of ICR doesn’t change much; all variations are reflected in delays and delivery ratios. The number of hops that the first copy of any message takes to reach to the destination remains unchanged when the number of participants changes (usually it is 3-4, not shown in figure). Hence, overhead remains unaffected as the number changes, but delays and delivery ratio are affected. When the distressed household count increases to 150, delivery ratio drops and delay becomes high. When rescue workers are increased to 100, both metrics improve, particularly the delay. When 5 main centers are opened instead of 2, delivery ratio increases sharply, but delay increases as well, because meetings with supply vehicles decrease as they report to more main centers. Police patrols have larger impact on both metrics. Since patrols connect neighborhoods, increasing the number of patrols offers better connectivity among neighborhoods. Both delay and delivery improve. The following table shows the effect on performance (delay and delivery), as the number of various agents increases. In most cases, as expected, Spray has higher overhead, but slightly high delivery ratio and less delay than ICR.

<table>
<thead>
<tr>
<th>Agents</th>
<th>Effect</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centers/camps</td>
<td>improves</td>
<td>exploit recurrence</td>
</tr>
<tr>
<td>Back-&amp;-forth vehicles</td>
<td>improves</td>
<td>shorter IC delay</td>
</tr>
<tr>
<td>Target dest. locs</td>
<td>drops</td>
<td>sparse, less meeting</td>
</tr>
<tr>
<td>Regular patrols</td>
<td>improves</td>
<td>more coverage/meeting</td>
</tr>
<tr>
<td>Moblty within cluster</td>
<td>improves</td>
<td>frequent meeting</td>
</tr>
</tbody>
</table>

Figure 2.6 (d—f) show performance trends as a function of different traffic patterns. Here also, we compare results with Spray. Earlier results are shown for random source-destination pairs. However, we show that perfor-
mance varies based on traffic mix. We compare three cases: survivors-to-
survivors (S–S, people at houses or camps send messages to other survivors),
survivors-to-rescuers (S–R, people asking for help to rescue workers, emergen-
cies, polices), and rescuers-to-rescuers (R–R, rescue workers, supplies, main
centers, polices, emergency responders, etc pass messages among themselves
for relief coordination). For each case, a node in source group randomly
chooses a node from destination group and generates message at rate 1 per
5 min. Performance follows the following order: R–R > S–R > S–S. Rescue
workers are mostly connected by the underlying movement of vehicles and
centers, hence they achieve the highest delivery and lowest delay. On the
other hand, survivors, as might be expected, are harder to connect. They
encounter the least delivery ratio and the highest delay. In all cases, the over-
head remains similar, but S–R has the lowest value, because survivors and
rescuers (who forward messages to others) experience repeated encounters.
Spray encounters similar effects due to traffic patterns and produces nearly
the same results, except that it has higher overhead.

Indeed, a consistent property of ICR is that it offers a lower overhead
compared to other routing protocols. We shall see that this property plays
an important role below, when we perform energy-limited experiments.

2.4.3 Results in Energy-constrained Environments

In this section, we consider resource-constrained environments where en-
ergy of some devices is limited. We assume that the mission life is 48 hours,
since the odds of finding stranded survivors may diminish after that. Before enforcing energy constraints, we first compare energy consumption under different protocols. In all experiments, we accounted two main energy components per protocol: communication and computation. While communication energy is computed from the power values of the device while sending and receiving bytes, computation energy is explicitly derived using a real phone (Android G1). Computation energy is mainly attributed to updating contact records and routing table at the beginning of each contact (except for Spray). To measure computation energy, we implemented each protocol for a moderate-sized (100 nodes) routing table onto the phone and ran the code (by emulating a contact) to see how much energy it spends on processing routing table updates. We took an average over many such runs. In simulations, this amount of energy is deducted from the energy balance of the contacting nodes at each contact. We also accounted energy consumption due to exchanging routing tables between nodes.

Figure 2.7(a) shows the total amount of energy consumed by different nodes under different protocols. We see that flooding-based schemes (Prophet and MaxProp) consume much more energy than quota-based ones (Spray and ICR). We also observe that ICR consumes much less energy than Spray. This allows ICR to be useful in an energy-constrained environment.

Next, we impose energy constraints, simulating node death when a node runs out of energy. We give different devices different amounts of energy. Cars and data centers are given infinite energy. Devices carried by humans and household routers (perhaps WiFi on cell phones) are given a low energy amount (200 – 400J). Admittedly, this initial amount is lower than that of a typical cell phone. The reduction is to match the reduced scale of our simulation compared to the size of a typical urban disaster, where many more messages are generated and may need to be forwarded within neighborhoods containing energy-constrained devices. Figure 2.7(b) shows the fraction of (energy constrained) nodes that remain alive (i.e., with remaining energy greater than zero) as time proceeds. We see that ICR allows more nodes to survive compared to other protocols. This has a direct impact on the ability of the network to deliver messages, as shown in Figure 2.8(a). The figure shows the total number of messages delivered by different protocols as the initial energy for energy constrained devices is varied. All protocols are subjected to the same application-level message generation rate. Some run out
of energy faster, hence delivering fewer messages. We show that, in contrast to results of the previous section, in energy-constrained scenarios ICR has a significantly better lifetime and total message delivery performance compared to all other protocols (20–30% more messages are delivered compared to others). In particular, flooding-based protocols perform the worst.

![Figure 2.8: (a) Delivery ratio at different initial energy values (b) Dissection of energy component of ICR.](image)

We also show that when the energy-differentiated service is enacted (shown as ICR(EN-*) in Figure 2.8(a)), ICR delivers more urgent messages and slightly less regular messages, since differentiated service rate-limits regular messages to allow urgent messages to go through.

Figure 2.8(b) further dissects the energy used by ICR into components to understand the sources of energy expenditure. The first of these is the communication cost. It represents the amount of energy expended on sending and receiving messages. The second is the control cost of the protocol. It is attributed to routing table exchange per contact. The last is the computation cost, which is incurred due to the need to process routing updates. The results are shown in Figure 2.8(b) (note the logarithmic scale on the vertical axis). It can be seen that the communication cost is approximately two orders of magnitude higher than the control cost and approximately three orders of magnitude higher than the computation cost. This explains why minimizing communication overhead is a winning policy in terms of lifetime and number of delivered messages, as reported earlier in this section.

Next, we evaluate in more depth the performance of energy-differentiated operation modes of ICR. In particular, we explore the performance impact of the initial energy available to energy-constrained nodes, as well as the
effect of the allowed energy depletion rate, \( \lambda \). Two performance metrics are of interest; namely, (i) the message delivery ratio, defined as the ratio of messages delivered to those generated, and (ii) the total number of messages generated over the mission lifetime. Note that, the total number of messages generated changes depending on the initial energy available to the nodes (since nodes stop generating messages when their energy is depleted), whereas the fraction of these messages delivered is more a function of the allowed energy-depletion rate, \( \lambda \), set by the differentiated service.

Figure 2.9 shows the effect of initial energy and the the effect of energy-depletion rate, \( \lambda \), on delivery ratio. We produce a mix where the ratio of urgent messages to regular messages is 1:5. We then run experiments with \( E_{\text{init}} = 100–1000 \) J for constrained devices and \( \lambda = 1.0–8.0 \) Joule/Hour (J/H). Observe that since \( \lambda \) polices the energy consumption attributed to regular messages, it is expected that fewer such messages will be delivered as \( \lambda \) decreases. As a result, more energy becomes available to the delivery of urgent messages. It can be seen that a significant improvement is indeed achieved in the delivery ratio of urgent messages, at the expense of a slight decline in that of regular messages. This is attributed to the fact that there are fewer urgent messages to deliver. Hence, a small change in the fraction of delivered regular messages frees up enough resources to make a big difference in the delivery of urgent messages. It is also seen that Spray and Prophet have a lower delivery ratio than both that of urgent messages and that of regular messages under ICR. This is due to larger overhead of these protocols.

Figure 2.10 shows the number of total messages generated at various \( \lambda \)'s. Note that the number of messages increases as the devices are given more
initial energy, also at various energy budget. Note also that Spray, Prophet and MaxProp generate fewer messages overall, as they lead to a faster energy depletion.

![Figure 2.10: Total number of messages generated.](image)

We conclude that ICR outperforms leading DTN protocols both in terms of the total messages generated and in terms of the fraction of them delivered. It also allows further improvements in delivery ratio for urgent messages. These advances are attributed primarily to its lower overhead, made possible by better exploitation of mobility and recurrence patterns in DTNs.

It remains to comment on the impact of certain protocol parameters on performance, which is the topic of the next section.

### 2.4.4 Impact of Protocol Parameters

ICR uses a new graphical DTN model, namely inter-contact graphs instead of traditional encounter graphs. In an encounter graph, physical devices are represented as nodes and the time gap between encounters involving the same pair of nodes is labeled on the corresponding edge. Unlike inter-contact graph, the delay on each edge in the encounter graph is solely given by the next hop, whereas the inter-contact graph the same depends on the previous hop as well. For example, both $\delta(ij \rightarrow ik)$ and $\delta(il \rightarrow ik)$ are same in the encounter graph (which is the average contact period of $ik$, i.e., the time gap between meeting of $i$ and $k$), but for the inter-contact graph they are not. Figure 2.11(a) shows the delivery ratio of ICR mounted on
an encounter graph as well as on an inter-contact graph. We see that ICR offers greater delivery ratio for the inter-contact graph than the encounter graph. This is because inter-contact presentation provides better estimation of delay distribution, thus chooses better paths.

![Figure 2.11](image1.png)

**Figure 2.11:** ICR on inter-contact and encounter graph and effect of timeout on delivery ratio.

![Figure 2.12](image2.png)

**Figure 2.12:** Bootstrapping ICR with Spray.

Next, we illustrate the advantage of bootstrapping ICR with “Spray”. Recall that ICR constructs routing tables to destinations as recurrence patterns are discovered, but uses spray whenever a path to the destination is unknown or confidence in the path is too low. In particular, messages are sprayed at the beginning, when the routing table is yet to be populated. Figure 2.12 shows how delivery ratio and overhead are affected by using Spray-based bootstrapping. In Figure 2.12, “Contact (Only)” uses only routing entries to forward messages, whereas “Contact+Spray” uses both routing entries and spray. We see that “Contact” has smaller delivery ratio at the first few
hours, but improves after a while (after 1 day). However, “Contact” incurs far fewer message transmissions (because it does not spray) than the “Contact+Spray” and Spray-and-Wait. In the case of “Spray”, the delivery ratio and overhead jump at the very first hour of operation. Hence, Spray is used during the time when ICR learns the network and routes.

In Figure 2.11(b), we show the effect of timeout on the delivery rate. ICR uses a timeout to compute the message delivery probability to destinations. At the smaller timeout value, many potential paths appear to be ‘bad’ because of their considerably long path-delays. This is why the delivery ratio is low for smaller timeouts ( < 5 hours). When the timeout is set to higher values, more paths become feasible and messages are propagated along these paths. However, there is a threshold (here it’s around 7 hours) beyond which the delivery ratio does not improve much. This is because at that timeout value, all possible paths are discovered. Increasing allowable latency does not include newer paths anymore. Figure 2.11(b) also shows the fraction of messages delivered within the timeout against the total delivery. It shows that at a smaller timeout value, there is a gap in timely delivery (i.e., some messages are delivered late); but as timeout value rises, the gap vanishes.

Figure 2.13: Performance results when moving agents deviate from their recurrent patterns for a while.

ICR exploits recurrence to infer low-variance delay paths to forward packets. Recurrence is demonstrated by repeated movement of agents in a given pattern. It would be interesting to see the impact on results if these movement patterns are violated for any reason. In Figure 2.13, we present performance of ICR when moving agents deviate from their recurrent patterns for a while (say, 10 hours). We experimented with three time intervals 10–20, 20–30, and 30–40 hours. During the chosen time interval, recurrent nodes, for example, police patrol and supply vehicles, move randomly in the map instead of following their regular patterns. After the time interval is elapsed,
nodes resume their regular patterns. We see that although performance metrics, such as delivery ratio, overhead and latency, all rise during the given time interval, at the end (after 48 hours), however, the values converge close to the results observed without any deviation (‘NO’ results). That means, ICR is responsive to changed mobility pattern of nodes and adapts its routing metrics accordingly.

2.5 Conclusion and Future Work

This chapter presented a novel approach to DTN routing tailored for disaster response networks. In contrast to flooding-based DTN routing techniques, we propose a routing table based approach that computes paths and costs to destinations using a novel routing metric called ‘inter-contact delay’. To construct these paths and costs, we exploit the underlying recurrent movement pattern that prevails in a rescue and relief operation after a disaster. The proposed protocol, ICR, has very low message overhead resulting in big energy saving, one of the most fundamental requirements at disaster time.

As a future work, we intend to propose a solution where we do not need to specify the explicit bound on initial message copies, rather the network itself obtains the “optimal” bound by learning more about the network. This leads to a new direction that requires gathering knowledge of the network in different aspects, viz., mobility, location, popularity, and so on. Then, we may combine them together to make good predictions on future encounters to determine which node should be given what fraction of message copies so that overall network performance becomes better (a similar work appears at [19]). Primarily we may focus on introducing a few more metrics that all independently capture the network behavior from different perspectives and then merge them together using some inference engine.
CHAPTER 3

DELAY BOUND FOR RECURRENT DTNS

This chapter describes an analysis technique that we develop for quantifying end-to-end delays of packets in DTNs \(^1\). In particular, we are interested in DTNs that experience recurrent mobility pattern arising due to repetitive movements of a set of moving agents. Disaster-response DTNs could be one, because in disaster-response operation individuals and vehicles perform repetitive tasks (e.g., move supplies between locations, report to bases, or patrol troubled areas). This enables information to be propagated inside the network through a series of recurrent node encounters. A question becomes interesting, how long does it take to transfer information across this network from specific sources to destinations, given some knowledge of recurrent node mobility patterns? The question is primarily hypothetical and used for planning only. During actual execution, it is impossible to measure global network state. Nevertheless, it is important to understand the relation between mobility patterns, load, and resulting delay, if one is to guess what network performance might be under given deployment conditions. We present analytic results that constitute a first step towards achieving such an understanding. The results can be used to quantify network performance under assumptions on load and mobility. In order to make the analysis, we bring techniques from distributed real-time literature, which is well-opted with theories and tools for analyzing timing properties of networked systems.

Unlike previous work on distributed real-time systems, where resources were assumed to be connected, in our scenario, messages are transferred through the network along a sequence of short-lived node contacts (i.e., encounters between nodes). Two nodes are said to be in contact when they are close enough to establish a connection and transfer packets. Usually, there

\(^1\)This work has been published as:
There is a long time gap between such contacts, compared to the duration of a contact. For example, moving vehicles might occasionally pass landmarks where wireless devices with some data storage capacity were planted to act as “mailboxes”. When packets are forwarded to a node, these packets reside on the node and are physically carried around until the node encounters another, possibly forwarding the packet onwards. The sum of all residence delays of a packet at different nodes (until delivery) constitutes its end-to-end delivery delay.

In this context, we address the following problem. Given a set of prioritized data flows of arbitrary source-destination pairs in a disruption-tolerant network, we compute an end-to-end packet delivery delay bound for each flow. One of the challenges of computing packet delivery delay is to understand the temporal and spatial behavior of the moving nodes and to represent the network in some analyzable form that allows us to derive delay expressions. As we mentioned earlier in Section 2.1.1, we introduced a DTN model where the network is modeled by an *inter-contact graph*. This graph abstracts the physical network into a collection of contacts and the associated time gaps between those contacts. In this work, we revisit the same graph and give it a completion with further annotations to allow modeling of end-to-end delay.

We use *delay composition algebra* [20] to analyze inter-contact graphs. Delay composition algebra allows reducing distributed systems to centralized systems that are equivalent to the original systems in terms of timing properties. Hence, we show how to convert disruption-tolerant network models into a form analyzable by delay composition algebra, so that it can be used to compute delay bounds. As might be expected, the resulting delay bounds are not tight. Evaluation shows, however, that the pessimism is moderate. The approach does provide a meaningful estimate of worst-case network behavior given recurrent mobility patterns.

The rest of this chapter is organized as follows. Section 3.1 describes our proposed network model for DTNs and the data flow model that we consider for delay analysis. Section 3.2 presents the techniques of computing end-to-end delay bounds of data flows. Section 3.3 presents simulation results, followed by conclusion (Section 3.4).
3.1 Modeling Recurrent DTNs

Delays in any computing system arise when tasks wait for resources to become available (e.g., to finish executing higher-priority tasks). Generally, resources may be processors, communication links, or anything that can be used by one task at a time. In a mobile environment (e.g., DTNs), we need to consider availability in space as well as in time. In this environment, a resource (say, a communication link) may be unavailable because it is busy (allocated to someone else at this time) or out of range (not available at the current location). Hence, one must quantify wait times as a function of not only workload, but also mobility patterns of nodes.

Rather than modeling the complex spatial behavior of nodes, we model the effects of such spatial behavior on the topological properties of interest (namely, connectivity and wait times). In particular, we are interested in identifying properties that can affect the end-to-end delay of packet delivery. Observe that complex spatial behavior models are not always necessary for this purpose. For example, in the architecture community, rather than understanding the exact “mobility patterns” of the program counter (in the space of memory locations), it was possible to achieve significant performance benefits simply by exploiting an overarching behavior principle such as locality of reference, which leads to caching. Similarly, in DTNs, to route packets, we exploit an overarching node behavior principle; namely, recurrence. We consider a category of networks where nodes perform recurrent jobs such as moving supplies between given locations, patrolling given neighborhoods, and shuttling between hospitals and relief centers. The network is thus formed of fixed nodes at certain key locations and mobile nodes that visit them often in a given order recurrently. Node contacts represent the main means for forwarding packets. We directly capture the delay properties of recurrent contacts, thus resulting in a very simple model for DTNs, as described below.

3.1.1 Network Model

We consider a set of static and mobile nodes extended over a geographic area, where they remain disconnected, except when contacts occur. At each contact, packets from one node can be transferred to the other. Packets,
originating at source nodes, can be forwarded via a sequence of contacts until they reach their destinations. Hence, instead of using the traditional network model consisting of network nodes and links, a DTN can best be described by contacts (encounters of pairs of nodes) and the timing relations between them.

We denote contacts by the corresponding node names. For example, contact \( ij \) denotes an encounter between node \( i \) and node \( j \). Much like with periodic task models, if a pair of nodes meet repeatedly at a fixed schedule, a time series can be used to enumerate all future meeting times. (Later, we relax the need to know future meeting times.) Let \( T_{ij}^{(l)} \) denote the \( l \)-th meeting time between node \( i \) and \( j \). The meeting time indicates the starting time of the corresponding contact, although the contact may continue for some duration. The time series for each contact enlists all meetings for a given pair of nodes. We represent the network, \( N \), as a collection of time series of all such contacts:

\[
N = \left\{ T_{ij}^{(l)} \mid 0 \leq l < \infty, \forall i, j, i \neq j \right\} \tag{3.1}
\]

We assume that for all contacts \( ij \), \( T_{ij}^{(0)} = 0 \). If two nodes never meet, we set, \( T_{ij}^{(1)} = \infty \). We define a few terms as follows:

\[
\delta^{(l)}(ij) = T_{ij}^{(l+1)} - T_{ij}^{(l)} \tag{3.2}
\]

\[
\delta^{(l)}(ij \rightarrow jk) = T_{jk}^{(l')} - T_{ij}^{(l)} \tag{3.3}
\]

\[
l' = \arg \min_h \left\{ T_{jk}^{(h)} | T_{jk}^{(h)} > T_{ij}^{(l)} \right\} \tag{3.4}
\]

where \( \delta^{(l)}(ij) \), called \textit{contact interval}, denotes the time gap between \( l \) and \( l+1 \)-th \( ij \) encounters and \( \delta^{(l)}(ij \rightarrow jk) \), called \textit{inter-contact interval}, denotes the time gap between the next immediate encounter of \( jk \) followed by \( l \)-th \( ij \) encounter. Semantically, \( \delta^{(l)}(ij) \) denotes the waiting time of node \( i \) for the next immediate transfer opportunity to node \( j \) after the \( l \)-th encounter, and \( \delta^{(l)}(ij \rightarrow jk) \) denotes the waiting time of a packet, at node \( j \), for the next immediate transfer opportunity to node \( k \) after node \( j \) has received the packet from node \( i \) at their \( l \)-th encounter.

In addition to meeting times, each contact is associated with another quantity, called \textit{contact duration}. Contact duration measures the time span of a
contact during which two nodes can remain in active radio communication by exchanging packets. We denote $\gamma^{(l)}(ij)$ as the contact duration of $l$-th $ij$ contact. The network, $N$, can now be described as a time series graph, $G = (\mathcal{C}, \Gamma, \mathcal{E})$, where $\mathcal{C}$, the set of contact intervals, $\Gamma$, the set of contact durations, and $\mathcal{E}$, the set of inter-contact intervals, are given by the following set of time series:

$$\mathcal{C} = \left\{ \delta^{(l)}(ij) \right\}_{0 \leq l < \infty, \forall i,j, i \neq j}$$

$$\Gamma = \left\{ \gamma^{(l)}(ij) \right\}_{0 \leq l < \infty, \forall i,j, i \neq j}$$

$$\mathcal{E} = \left\{ \delta^{(l)}(ij \rightarrow jk) \right\}_{0 \leq l < \infty, \forall i,j,k, i \neq j \neq k}$$

The end-to-end packet delivery delay between a pair of nodes can be expressed as a summation of contact intervals and inter-contact intervals. Suppose, a packet $P$ originates at node $i$ and takes the contact sequence $ij, jk$, to reach node $k$, i.e., node $i$ forwards the packet to $j$ (at contact $ij$) and then $j$ forwards to $k$ (at contact $jk$). For brevity, let the packet be created at $T^{(l)}(ij)$ (if not, a certain offset can be added). Assuming the actual packet transmission time on an active connection is zero, the end-to-end delay is simply the wait time between two contacts. Hence, $delay(P) = \delta^{(l)}(ij \rightarrow jk)$ (Figure 3.1(a)).

This delay is however exact only if the packet can be successfully transferred at the very first occurrence of each contact. Depending on the number of other packets waiting for the same contact, the packet may not be able
to be transferred to the next peer on the first contact; it may be delayed until the next contact or beyond. Figure 3.1(b) illustrates the case where $P$ is transferred to node $j$ on third $ij$ contact, and to node $k$ on second $jk$ contact. Hence, the end-to-end delay is given by:

$$\text{delay}(P) = \sum_{l}^{l+1} \delta^{(l)}(ij) + \delta^{(l+2)}(ij \rightarrow jk) + \delta^{(l')}(jk)$$  \hspace{1cm} (3.8)

One obvious problem with the above expression is that it requires knowledge of the entire future time series of contacts. The complete time series of contacts for a large network operating for a moderately long time would be very hard and expensive to collect. One possibility is to replace the exact time series with the probability distributions of contact intervals. Given delay distributions for all contacts and edges in the graph, one can probabilistically compute delays between contacts. Since we are interested in the worst case delay bound, however, we instead consider the worst case time separation between two successive contacts. We define the following three terms:

$$\delta(ij) = \sup \{\delta^{(l)}(ij)\}, \forall i, j$$ \hspace{1cm} (3.9)

$$\gamma(ij) = \inf \{\gamma^{(l)}(ij)\}, \forall i, j$$ \hspace{1cm} (3.10)

$$\delta(ij \rightarrow jk) = \sup \{\delta^{(l)}(ij \rightarrow jk)\}, \forall i, j, k$$ \hspace{1cm} (3.11)

In practice, the above quantities may be estimates of the maximum delays between pairs of successive contacts. For contact duration, it denotes the minimum span. If we replace all $\delta^{(l)}$’s for a certain contact, say $ij$, by a single $\delta(ij)$ value, we obtain a purely periodic network, in that $\delta(ij)$ denotes the contact period of the contact $ij$. Semantically, $\delta(ij)$ denotes the longest time gap by which node $i$ or $j$ expects to encounter the next contact $ij$, no matter when they start waiting for the contact. Similarly, $\delta(ij \rightarrow jk)$ denotes the longest possible time that node $j$ waits to encounter node $k$ after it meets node $i$. Obviously, while $\delta(ij)$ and $\delta(ji)$ are same, $\delta(ij \rightarrow jk)$ and the reverse $\delta(kj \rightarrow ji)$ are not. To appreciate the difference, consider a patrol that circles a given neighborhood on a specified path. The length of one round may be an hour, but the time between passing intersection $i$ and the subsequent intersection $k$, by patrol $j$, may by only a few minutes. Once
is passed, it takes the rest of the hour to reach $i$ again. Hence, $\delta(ij \rightarrow jk)$ is much shorter than $\delta(kj \rightarrow ji)$.

Based on the above discussion, we define our DTN model as follows. The network is described by a time-invariant directed graph, called inter-contact graph, $G = (C, E)$, where $C$ is the set of encounters and $E$ is the set of edges between encounters. Each vertex $c$ is annotated by a tuple $(\delta(c), \gamma(c))$, measuring the contact period and contact duration of $c$, respectively. To avoid ambiguity, we use the term “vertex” to denote a contact in the inter-contact graph, whereas “node” means the physical device. We use $c_1 \rightarrow c_2$ to denote an edge that expresses the occurrence of contact $c_2$ followed by contact $c_1$. Two contacts have an edge if they share a common node between them. We write the shared node in the inner side of the edge expression for better understanding of the sequence of packet transfers. For example, $ij \rightarrow jk$ implies the transfer of packets in a sequence $i \rightarrow j \rightarrow k$. The reverse edge between the same contacts is written as $kj \rightarrow ji$. Once we adopt the above notations, we can compute the end-to-end delay of a packet in terms of $\delta$’s. For example, Equation 3.8 now becomes, $\text{delay}'(P) = \delta(ij \rightarrow jk) + 2 \times \delta(ij) + \delta(jk)$. Obviously, $\text{delay}'(P) \geq \text{delay}(P)$.

For each contact $c$, we introduce another term transfer volume, $v(c)$, which indicates the minimum number of bytes that a node can transfer onto the peer node upon contact $c$. Transfer volume of a contact is determined by the contact duration and transmission bandwidth of the link associated between the node pair. For a transmission rate of $R$ (byte/sec), the transfer volume is given by $v(c) = R \times \gamma(c)$. Assuming the symmetry of the channel, we consider $v(ij) = v(ji)$.

We denote a path or route, in the inter-contact graph, as a sequence of vertices (i.e., contacts) $\{c_1, c_2, \ldots, c_k\}$ such that each successive edge $c_i \rightarrow c_{i+1}$ exists in the inter-contact graph. The first and last contact contain the source and destination node respectively. A path explicitly enumerates the sequence of contacts, and in turn nodes, that a packet follows when forwarded along the same path. The path can also be written as $c_1 \rightarrow c_2 \rightarrow \cdots \rightarrow c_k$.

The inter-contact graph, $G$, expresses the topographical properties of the network. Any timing property of the network, particularly the end-to-end delay, can be computed in terms of $\delta$’s and $v$’s of $G$. All such delays are, however, an over-estimation of the actual delays computed from $G$. Since we are interested in worst case delay bound, such over-estimation is allowed.
3.1.2 Data Flow Model

We define a data flow to be a set of packets emanating from a source node that traverse a given path to reach a certain destination. Multiple data flows may be forwarded at the same contact or a set of contacts. We consider prioritized flows in that each flow is assigned an explicit priority value. Without loss of generality, we assume that flow ID and its priority are the same and lower ID indicates higher priority, flow 1 being the highest priority flow. Each flow $k$ is assigned a path, $path_k$, that enumerates the sequence of contacts along which packets from the same flow are forwarded. Later, we use the notion of flow and path interchangeably. All packets in a flow $k$ have the same size, $p_k$. Packets can be injected periodically or sporadically. For periodic flows, $P_k$ denotes the period of flow $k$, which is the time interval between two successive packets injected by the source. For aperiodic flows, $P_k$ denotes the minimum time gap between two successive packets.

Although the inter-contact graph may have many vertices and edges, in the following, we shall only consider the subgraph that contains flows. Edges and vertices that are not part of any flow are removed. This subgraph is called the data flow graph.

We need to explain why data flows are denoted as a sequence of contacts, not nodes. It is obvious that packets will be transferred from one node to another. Recall that our objective is to compute the end-to-end delay for a set of data flows. So, we need to identify the entities where competition among flows happens, in that a flow (possibly a higher priority one) imposes delay on other flows (of lower priority). These points of competition are the contacts. Only flows that need to be forwarded at the same contact compete. We assume that memory is sufficient. Hence, packets that reside on a node, waiting for different contacts, are not in competition.

3.2 End-to-end Delay Bound for DTNs

In this section, we describe how an end-to-end delay bound is computed for fixed-route prioritized data flows in DTNs. We first analyze the various components of the end-to-end delay and then use delay composition algebra to compute the delay bound.
3.2.1 Delay Components

As modeled by our data flow model, each packet from a particular flow follows a certain sequence of contacts. The time gap between two successive contacts on the path is given by the inter-contact delay. Once a contact occurs, a connection (i.e., link) between the two corresponding nodes is established and packets are transferred from one node to another in priority order. Due to the limited transfer volume, it may not be possible to transfer all packets in the buffer to the next node. Some packets may need to wait for a future occurrence of the contact. For a particular packet, we define transfer delay to be the time between the instant of first encounter with the next-hop node and the instant when it is successfully transferred. Note that, the inter-contact delay is the time between different contacts, whereas the transfer delay is caused by waiting for returns of the same contact. The sum of inter-contact delays and transfer delays for all contacts along the path constitutes the total end-to-end delay of a particular flow.

We assume that actual transmissions over an active connection happen in zero time. This assumption is based on the fact that physical packet transmission time along a link is negligible compared to the time a node waits for contact with another in a DTN.

In Figure 3.2(a), we demonstrate the delay trajectory of a packet from flow $c_1 \rightarrow c_2 \rightarrow c_3$. We observe from the figure that the packet required 2 occurrence of $c_1$ contact, 1 $c_2$ contact and 3 $c_3$ contacts to make a successful transfer onward and for the final delivery. There are also two inter-contact delays for two edges $c_1 \rightarrow c_2$ and $c_2 \rightarrow c_3$. There is also a certain amount of waiting time involved prior to the very first contact $c_1$, after the packet was originated. This delay is smaller than $\delta(c_1)$, the contact period of $c_1$. We safely assume this waiting delay to be $\delta(c_1)$. In subsequent delay com-
putations, we implicitly add this value to the estimated end-to-end contact delay.

The end-to-end delay for a certain flow is determined as the summation of all inter-contact delays and sum of zero or some integer multiple of contact periods along the path, as given by the following general expression, for flow $k$:

\[
\text{delay}(k) = \sum_{c_j \in \text{path}_k} \delta(c_j \rightarrow c_{j+1}) + \sum_{c_j \in \text{path}_k} x_j \delta(c_j)
\] (3.12)

where $x_j \geq 0$ denotes the number of times contact $c_j$ has been missed by the flow. The transfer delay at contact $c_j$ is $x_j \delta(c_j)$. While the sum of inter-contact delays for a flow is entirely given by the path of the flow, which does not depend on the presence of other flows, the transfer delay is greatly influenced by the presence of other flows, particularly by higher priority flows that share one or more contacts with flow $k$. One subtle issue is to compute $x_j$, counting the number of contact misses that a certain flow suffers due to the presence of other higher priority flows. Given the data flow graph, the $x_j$’s are the only unknowns in computing the end-to-end delay bound for a flow. Estimating the worst-case values of $x_j$’s is the purpose of our analysis.

To ease the analysis, we order the terms in the delay summation putting the sum of all inter-contact delays first ahead of the transfer delays as suggested by Equation (3.12) and depicted in Figure 3.2(b). This split does not affect the total end-to-end delay. Let the sum of inter-contact delays, *end-to-end contact delay*, be $D_c(k)$ and the sum of transfer delays, *end-to-end transfer delay*, be $D_t(k)$ for flow $k$. The total end-to-end delay, denoted by $\text{delay}(k)$, is therefore given by, $\text{delay}(k) = D_c(k) + D_t(k)$.

Splitting the transfer delay from the contact delay separates path-specific delays (the wait for first contact with next hop) from load-specific delay (the wait for a number of returns of that contact before a packet’s turn comes to be forwarded). The latter delay is a step function. As described in the next section, this step function can be upper-bounded by a straight line. The slope of the straight line can be interpreted as link bandwidth in a virtual connected network. Hence, the original path delay of flow $k$ can be decomposed into a leading delay $D_c(k) = \sum_{c_j \in \text{path}_k} \delta(c_j \rightarrow c_{j+1})$ plus the path delay, $D'_t(k)$, through a virtual connected network with point-to-point links, that upper bounds $D_t(k) = \sum_{c_j \in \text{path}_k} x_j \delta(c_j)$. Real-time literature has
recent results to compute delay (upper) bounds on $D'_t(k)$. In particular, we use delay composition algebra [20] for that purpose.

3.2.2 Computing End-to-end Transfer Delay

Delay composition algebra is used to compute the end-to-end delay bounds for multi-stage distributed tasks, each executing on a pipeline of resources. In this case, tasks are flows. Individual resources are path links in the virtual connected network mentioned above. Packets are transferred on this links in priority order.

The delay composition algebra maintains so called load matrices, which we refer to as delay matrices. For each vertex of the virtual connected network (i.e., for each contact in the data flow graph), a two dimensional delay matrix of size $n \times n$ is defined, where $n$ is the number of flows in the network. Semantically, the delay matrix for a certain contact node designates how much delay a higher priority flow imposes on a lower priority flow. In particular, $(i,k)$-th entry of the matrix, denoted as $q^c_{i,k}$ for contact $c$, contains the amount of delay that is exerted by flow $i$ on a lower or equal priority flow $k$ (for $i \leq k$), when packets from both flows wait for the same contact $c$. Given all delay matrices for all vertices of the data flow graph, the delay composition algebra then reduces the whole network to a single vertex containing only a single matrix that quantifies the overall delay interactions among all flows in the network. The algebra uses two reduction operators, namely PIPE and SPLIT. In Figure 3.3, we show the reduction process for two flows, $c_1 \rightarrow c_2 \rightarrow c_4$ and $c_1 \rightarrow c_3 \rightarrow c_4$.

In the following, we describe how the initial delay matrix per contact is computed for our particular data flow graph. Suppose, a packet $P$ from flow $k$ is waiting for contact $c$, where $\delta(c)$ is the contact period and $v(c)$ is the
contact volume. Since the wait time for the first occurrence of the contact is accounted for separately, we consider only the delay in waiting for returns of the same contact. Hence, the packet experiences zero delay, if it can be forwarded at the very first contact, or $\delta$ delay if forwarded at the second contact, $2\delta$ for third contact, and so on. The total delay is a multiple of $\delta$, depending on how many returns of the contact it ends up waiting for. This, in turn, depends on the number and size of higher priority packets waiting for the same contact. Let $\sum_{i=1}^{k} p_i$ be the sum of bytes of all higher or equal priority packets waiting at the contact, where $p_i$ is the packet size of flow $i$. Considering the contact can transfer only $v(c)$ bytes at every occurrence, the delay exerted on $P$ is given by:

$$
delay_c(P) = \left\lfloor \frac{\sum_{i=1}^{k} p_i}{v} \right\rfloor \times \delta \\
\leq \left( \frac{\sum_{i=1}^{k} p_i}{v} \right) \times \delta \\
= \sum_{i=1}^{k} \left( \frac{p_i}{v/\delta} \right) \quad (3.13)
$$

Figure 3.4: Linearization of transfer delay.

Now, the question arises how much delay flow $i$ imposes on flow $k$. Obviously, it is $\frac{\delta}{v} \times p_i$. Adding all such delays imposed by all higher priority packets gives the total delay on flow $k$ at contact $c$. This effectively emulates a situation where all packets are waiting in a queue and are then released one after another, in their priority order, at a rate $r = \frac{v}{\delta}$. Based on this observation, we can replace a high speed, periodic but discretely available link by a slow but continuous one with the same effective transfer rate. This is called
“linearization of transfer delay”. By this, a contact $c$ with contact period $\delta(c)$ and contact volume $v(c)$ can be replaced by a continuous link of transfer rate $r(c) = \frac{v(c)}{\delta(c)}$ (Figure 3.4). Due to linearization, the estimated transfer delay becomes larger than the actual transfer delay. This over-estimation is however bounded by at most $\delta$. Since we are computing the worst case delay bound, such over-estimation is allowed as long as the over-estimation is bounded.

Therefore, the initial entries for the delay matrix at contact $c$ is given by:

$$ q_{i,k}^c = \begin{cases} \frac{p_i}{r(c)} & \text{if } i \leq j \text{ and } i, k \text{ pass through } c; \\ 0 & \text{otherwise.} \end{cases} $$

### 3.2.3 Computing the Delay Bound

Delay composition algebra gives a reduced system represented by a single delay matrix that quantifies the worst case delay characteristics of the original data flow graph. While computing the equivalent delay quantities, the composition algebra however considers a single packet from each flow. But, there could be multiple packets from each flow in the network (akin to multiple invocations of the same task in a uni-processor system). At this point, we need to convert the flows of the data flow graph into an equivalent task set and let the task set “execute” in a uni-processor setup. As per the composition algebra, the execution of these tasks in a single processor—provided that their execution times are appropriately chosen from the final resultant delay matrix—generates the worst case delay bound for the original distributed data flows.

As suggested by the composition algebra, in order to compute delay bound for flow $k$, we need to build a task set of size $k$, $\{T_1^*, T_2^*, \ldots, T_k^*\}$ such that execution times are chosen as follows:

- For $i < k$ (i.e., higher priority flows), $C_i^* = q_{i,k} + r_{i,k}$.
- $C_k^* = q_{k,k} + r_{k,k} + s_k$.

Now, we run the classical response time analysis [21] to compute the end-to-end delay bound (a.k.a the worst case response time) for each flow. Response time is defined as the time gap between the time when a task appears
in the system and the time when it ends, being interrupted by higher priority
tasks in the middle. In our case, this response time is actually the end-to-
end transfer delay. The response time \( R_k \) of the task \( T_k^* \) is computed by the
following recursive relation:

\[
R_k^{(0)} = C^*_k
\]

\[
R_k^{(l+1)} = C^*_k + \sum_{i=1}^{k-1} \left\lceil \frac{R_k^{(l)}}{P_i} \right\rceil C^*_i
\]

(3.14)

where \( P_k \) is the period of the data flow \( k \) for periodic flows, or the least time
gap between two successive data packets from the same flow \( k \). The recursion
terminates when \( R^{(l)} = R^{(l+1)} \) for some \( l \).

The response time, \( R_k \), gives the upper bound of end-to-end transfer delay,
\( D_t(k) \), for flow \( k \). We need to add the precomputed end-to-end contact delay,
\( D_c(k) \), (i.e., sum of inter-contact delays along the path) with it to find the
total end-to-end delay bound for flow \( k \), as follows:

\[
\text{delay}(k) = D_c(k) + D_t(k) \\
\leq D_c(k) + R_k \\
= \sum_{\text{path}_k} \delta(c_j \rightarrow c_{j+1}) + R_k
\]

(3.15)

3.2.4 Pessimism in End-to-End Delay

The pessimism in the computed end-to-end delay arises due to two main
reasons: smaller inter-contact delay and extended transfer delay.

- **Smaller inter-contact delay**: We assumed that once a successful trans-
fer is made, the next contact leading to the next transfer could only
happen after an inter-contact delay has been elapsed. Sometime, this
inter-contact delay overlaps with the transfer delay. So, the addition
simply counts double. For instance, let us consider a flow along the
path \( ij \rightarrow jk \) (Figure 3.5). Let the first contact \( ij \) happen at time 0
and the contact doesn’t lead to successful transfer of a given packet.

54
The transfer happens at the next contact at time 10. The next contact $jk$ is supposed to happen at time 15. In that case, $j$ holds the packet for $15 - 10 = 5$ units of time. But, in our computation, we added 15 as the inter-contact delay between $ij$ and $jk$. This leads to an over-estimation.

- **Extended transfer delay**: Due to linearization of transfer delay, the transfer delay is over-estimated by an expression $(x - \lfloor x \rfloor) \times \delta(c)$, for some $x$ and contact $c$ (Equation 3.13). Since $0 \leq x - \lfloor x \rfloor < 1$, on the average the over-estimation per contact is $\frac{1}{2} \delta$. So, the average over-estimation for the entire path is $\frac{1}{2} \sum_{c \in \text{path}(k)} \delta(c)$.

### 3.3 Evaluation

In this section, we evaluate quality of end-to-end delay bounds for DTNs. We simulate a disaster-response scenario based on the post-disaster mobility model (PDM) presented in Section 2.3. PDM, implemented on top of the ONE [13] simulator, has four major movement types: inter-center movement (repeated back and forth movement of supply vehicles between relief centers and main coordination centers), rescue worker movement (localized mobility of volunteers in distressed neighborhoods), police patrols (cyclic police patrol movement among neighborhoods), and emergency movement (vehicles attending to an emergency event). There are also several types of fixed nodes including centers, police stations, and hospitals, which act as meeting places of moving nodes that help relay packets among them. All moving entities and centers are equipped with ratio devices and these devices run DTN routing
protocols. All experiments are conducted for 5 neighborhoods, 2 main centers, 10 relief and evacuation camps, 20 supply vehicles, 20 rescue workers, 5 police patrols, 5 emergency vehicles, and 50 distressed households.

Figure 3.6: The data flow graph containing all flows.

We construct data flow graphs from a trace file generated for a large number of runs of the above mobility scenario with different parameters. The trace contains delay attributes (mean, min, max, std. dev) of contact intervals and durations for contacts. Contacts are depicted by vertices labeled by the node ID involved in the encounter (e.g. “64:14” denotes a contact between node 64 and 14). Inter-contact delays are labeled on edges between such vertices. In our case, the entire trace contains nearly 1,200 contacts and 30,000 edges. To simplify, we assume that data flows on only parts of that network. Hence, we construct data flow graphs that are subsets of the above trace. Towards that end, we randomly choose a number of paths of certain length (e.g., up to 8) by picking connected edges from the trace. We then pick edges that connect vertices from one path to another. Figure 3.6 shows one such data flow graph. Each experiment is conducted on 10 such graphs, each containing 100 random flows. The typical simulation parameters are shown in Table 3.1.

As per our network model, we use the worst-case values for contact intervals, contact durations and inter-contact delays to label the vertices and edges in the data flow graph. In Figure 3.7, we plot the delay distribution
Table 3.1: Simulation parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of flows per graph</td>
<td>40-100</td>
</tr>
<tr>
<td>Packet size</td>
<td>2.5-10.0KB</td>
</tr>
<tr>
<td>Flow period</td>
<td>1/2-3 hour</td>
</tr>
<tr>
<td>Deadline</td>
<td>12-24 hour</td>
</tr>
<tr>
<td>Confidence level</td>
<td>95%</td>
</tr>
<tr>
<td>Conf. interval</td>
<td>1% of mean</td>
</tr>
</tbody>
</table>

Figure 3.7: Delay metrics of vertices and edges of the data flow graph.

of contact intervals, contact durations and inter-contact delays for some 400 contacts and edges that our generated data flow graphs entail. The figure plots the average metric and 95% confidence interval for all observations collected from several instances of the aforementioned disaster mobility scenario. The graph also shows the corresponding worst case values that have been used by our data flow graphs. For the post-disaster mobility scenario, the average sum of worst case inter-contact delays for a 5-hop path is around 10 hours. That’s why we choose 12-24 hour deadlines for flows.

Figure 3.8: Delay bound ratio for varying loads.

We evaluate two main performance metrics: delay bound ratio, the ratio between the average delivery delay of packets and the estimated end-to-end delay bound for a particular flow, and utilization per contact, the fraction of time a particular contact transfers packets over its entire contact duration. Bound ratio indicates the tightness of the estimated delay bound compared to actual delivery delay, whereas utilization measures “load” of the network.

Figure 3.8 shows the delay bound ratio for different load scenarios. Fig-
Figure 3.8(a) shows the bound ratio for varying number of flows for different periods. We observe that the bound ratio is approximately 30%, which remains fairly constant as the number of flows increases and the period changes. Figure 3.8(b) demonstrates the effect of path length on bound ratio in different deadlines. Deadlines are determined by deadline factors (DF), which means deadline (in hour) per hop. In this case too, the bound ratio remain fairly close to 30%, not being affected much by deadlines and path lengths. This is because factors that affect original packet delay also affect the estimated delay bound in the same scale. That’s why the ratio remains fairly unchanged. In terms of pessimism, the results match closely with the earlier delay composition results, presented in [20].

In order to evaluate the tightness of the estimated delay bound, we plot maximum and average bound ratios for 50 flows in Figure 3.8(c). Maximum bound ratio for a given flow is determined by dividing the maximum packet delay by the estimated delay bound for the same flow. We observe that the average bound ratio remains \( \approx 30\% \), whereas the maximum bound is nearly 60\%, sometimes, the bound is even as close as 80\%. This high bound ratio is possible when a few packets actually suffer delays that are very close to their estimated bounds. In particular, this can happen to the highest priority flow among a set of competing flows that only undertake the end-to-end contact delay, but zero or very low transfer delay.

We construct an offline (capacity planning) admission controller based on the computed delay bounds. For planning purposes, the admission controller only considers those flows whose deadlines are smaller than the end-to-end delay bound estimated the controller. In a network with that traffic profile, no packets will ever miss their deadlines. The delivery guarantee is however achieved at the cost of reduced utilization of network resources. In Figure 3.9, we demonstrate the effect of the capacity planning admission controller.
controller on network performance. Figure 3.9(a) compares when packets are injected into the network in violation of capacity planning and when packets are constrained to what the planning deemed feasible. We observe that in the absence of an admission controller, as more packets are injected, more and more packets miss deadlines. When packets adhere to capacity profiles, fewer packets are used, but all those packets meet their deadlines. Figure 3.9(b) plots what fraction of total flows are found admissible for different flow periods and deadlines, compared to no capacity planning.

Figure 3.9(c) shows the average utilization per link for different flow periods (1/2 Hr and 1 Hr). Without planning, the utilization goes up as the number of flows increases, but eventually some flows miss deadlines, which makes the timely traffic lower. When traffic is restricted to computed capacity, utilization goes down, since less flows are found admissible, but all flows meet deadlines.

![Graphs showing transfer delays and max transfer delay](image)

(a) Period = 1 Hr  
(b) Varying periods

Figure 3.10: Transfer delays of flows at various loads.

The estimated end-to-end delay bound is the sum of contact delays and transfer delays. For a given flow, the contact delay is determined by the path of the flow, whereas the transfer delay is due to interactions among flows and is given by the amount of time a lower priority flow is delayed by other higher priority flows. This delay is computed by the delay composition algebra. Figure 3.10(a) shows transfer delays for some 40 flows. In most cases, transfer delays are quite small compared to the end-to-end contact delay, which is on average 380 minutes for this flow set. The maximum transfer delay (nearly 470 minutes) is however comparable. Recall that the transfer delay depends on periods of flows (due to Equation 3.14). Usually shorter periods cause longer response times, hence longer transfer delay. Figure 3.10(b)
plots the maximum transfer delays for different flow periods. As we observe, the maximum transfer delay decreases as the flow period increases.

3.4 Conclusion

In this work, we analyze end-to-end delays of prioritized flows in DTNs using delay composition algebra. Knowing the end-to-end delay bounds for data flows in DTNs can help in planning resource provisioning and even influencing the mobility of agents in practical DTN deployment cases. DTNs pose challenges in characterizing delay attributes of the network, in part, because of their disconnected nature. We model the network as a collection of distributed encounters between nodes and represent sequences of encounters by edges between them. We then systematically convert the DTN model into a “connected” network representation so that data flows are expressed as paths in the graph and the worst case end-to-end delay bounds are computed. Evaluation shows that computed delay bounds are not significantly different from measured worst-case values.
In this chapter, we describe our efforts to develop application services that can be useful in disaster context. One such application could be situation awareness. Situation awareness refers to collecting various information from within the disaster site that gives the overall status of people’s life and properties therein. Information may be about the environment (e.g., location of damages, flooded areas, infrastructure collapse) as well as the status of individuals affected by the disaster (e.g., health hazard, spread of diseases). Situational awareness is important for victims as well as for first responders and rescue workers. People inside the disaster area need this information not merely for their safety and well-living, but because they often participate in the response efforts themselves [22]. Huge proliferation of wireless devices, such as mobile phones, tablets, enables people to seamlessly generate and share information about important happenings around their livelihood in their regular life. These information can be of various formats, such as in text, images, audio and video clips. Popular dissemination platforms, such as social networks (e.g., Facebook), micro-blogging service (e.g., Twitter) and photo sharing service (e.g., Flickr), made the sharing and distribution of these information onto others very easy and ubiquitous. The same activity can be harnessed during disaster time when information becomes more important and critical (e.g., information about a missing person). In fact, there has been evidence of people using these services, such as Twitter, during recent disaster events, e.g., Hurricane Irene.

In our work, we focus on building a DTN application service, called PhotoNet, that leverages the potential of pictures for situation awareness\(^1\). Pictures are admittedly rich data objects that capture real physical events more

\(^1\)This work has been published as: Md Yusuf Uddin, Hongyan Wang, Fatemeh Saremi, Guo-Jun Qi, Tarek Abdelzaher and Thomas Huang, “PhotoNet: a Similarity-aware Picture Delivery Service for Situation Awareness”, IEEE RTSS, Viena, Austria, November, 2011.
reliably than other data formats (e.g., text), not to mention pictures can be easily generated by people at almost zero overhead. It is also popularly believed that a picture can say thousand words. During the moments of disaster when chaos prevails and human attention is diverted, enabling people to share their condition and critical physical events (e.g., a collapsed house) merely by taking pictures would be very practical and useful. After all, taking pictures is essentially less engaging than generating other information content (say, typing text reports).

PhotoNet can be regarded as a media crowd-sourcing service for disaster response. The service is geared for disaster recovery scenarios where survivors and first responders survey post-disaster damage and send pictures of it from their mobile devices (such as camera phones) to a command or rescue center. In practice, such picture delivery services exist for situation awareness during disaster recovery. One example is “geo-pictures” (http://www.geo-pictures.eu/), which relies on satellite communication for sending pictures to the command station. Ushahidi (http://www.ushahidi.com) is a web portal that allows users to upload text messages or images and geo-tags all content onto a map. This service was used to report civil violences and events due to mass disruptions during the Kenyan election in 2008.

While the above mentioned services use satellite communication or the Internet, we assume that infrastructure (such as power and cell towers) is down, making communication possible only opportunistically between nearby wireless battery-operated devices. As nodes move and meet other nodes, data spreads, leading to a disruption-tolerant network (DTN) model. We are interested in scenarios where bandwidth, contact opportunities between nodes, or node storage is limited, leading to resource bottlenecks that prevent delivery of all pictures to the destination. In our application, pictures become the first person payload inside the network, which effectively converts the DTN into a participatory camera sensor network. Before we describe our application services in more detail, we shortly iterate the promise of participatory camera sensor networks in general as well as in the context disaster response over DTNs.
4.1 Participatory Camera Sensor Networks

We define a participatory camera network as one where participants contribute pictorial data, either on their own initiative or through participation in a corresponding data collection campaign. For example, in the aftermath of a natural disaster, relief workers and other first responders might survey an area in search of damage that is then pictorially documented and reported. Another application might be to ask residents of a neighborhood to pictorially document issues that require attention in their neighborhood (e.g., graffiti on walls, trash piles, hazardous potholes, or other problems). Yet a third application might be to compile a list of most visited tourist landmarks from pictures contributed by tourists in a given location. Participatory camera sensing applications are made popular by the vast proliferation of cameras and camera phones in the possession of the average individual, not to mention the richness of information contained in pictures compared to other sensing modalities.

Working with pictures apparently has a couple of issues. Firstly, pictures are large data objects (usually 100's KB or even several MB each). So, in contrast to other data content, such as text reports, they consume more resources in terms of network bandwidth and storage. The second issue is the inherent redundancy among generated pictures. Suppose a community of volunteers are independently uploading pictures of flood damage in their neighborhood after a storm. It might happen that multiple of them take pictures of the same damaged site or object albeit a slight change in camera angle or location thus making one of them redundant in the presence of another. Redundancy may also arise when the server might already know of some of the damage (from earlier uploads) and hence may not need some of the pictures. Another issue is the existence of noise and outliers that are not representative, since the participants may not always be entirely reliable. The last issue is exacerbated by the fact that taking pictures is a very non-intrusive to user attention (unlike writing text by pressing keys and thinking on sentence formation), which causes users to take pictures almost everything they deem interesting. In many cases, the captured pictures might not be related to task at hand. To conserve resources, one would like to minimize redundancy while eliminating the outliers.

Resources may need to be conserved for several reasons. For example,
participants, who upload pictures from their mobile phones, may have to pay for their data plans. Hence, uploading less data is better. If pictures taken by participants propagate opportunistically, for example over a disruption-tolerant network (DTN), an individual participant may end up collecting too much redundant content and may need to do some data triage to fit the local storage or energy constraints. Such might be the case in disaster recovery scenarios, where infrastructure may be destroyed leaving only DTN-style communication, or in military scenarios, where groups of soldiers in the field may have only a low bandwidth channel to a remote base, making it advantageous to triage the data locally prior to transmission on that channel.

We therefore develop a picture delivery service, PhotoNet, that aims at collecting and delivery of the most representative subset of pictures from a vast pool, where a significant portion of pictures are redundant, irrelevant or noisy. A representative subset is one that offers roughly the same coverage of the environment, but with fewer pictures. Since the network runs in a lower capacity with respect to the total volume of data generated, an in-network prioritization scheme is required to identify which content is more useful (non-redundant with respect to current collection) than others so that data objects can be accordingly transferred or stored in the network. Our contribution is to design these prioritization protocols that we discuss next. In the following, we use term data objects, items, content and pictures interchangeably.

4.2 Content-Aware Prioritization (CAP)

We propose a set of content prioritization schemes, generally called CAP (Content-Aware Prioritization), that aims at minimizing redundancy of delivered content subject to different constraints. CAP runs on participants’ phones (the clients) and on a destination server (the collection point). When pictures are taken using our application, they are locally stored on the device. When two CAP-enabled devices meet, they gossip by exchanging a portion of their pictures. Similarly, when a device connects to the destination server it uploads a portion of its pictures. CAP assigns the priority order to pictures in which they should be transferred onto the other end, such that both the sender and the receiver ultimately end up having a set of pictures
with the least possible redundancy (resulting in the most representative subset). In the same way, prioritization happens when pictures are required to be dropped when the storage becomes full at a node. CAP determines the dropping order.

CAP, bears an interesting difference from traditional traffic prioritization schemes (e.g., those discussed in network QoS literature [23]). While prior schemes associate priority with each message independently (e.g., based on its content type, class, source, or destination), CAP considers the relations between different objects in assigning priority to each. Understanding relations between objects is a necessity, not a choice, whenever network utility is not additive in utility derived from delivery of individual objects from different sources. For example, delivering the first picture of a damage scene may have high utility. Delivering more pictures of the same scene from other sources has progressively lower utility, since the information becomes partially redundant. This utility saturation effect calls for content prioritization schemes in which priority is not a value defined inherently for each object in isolation, but rather is a function of relations between objects (such as whether a new picture is similar to, and from the same location as, a previously delivered one).

The above observation leads to an important concern. If priority cannot generally be determined for each object in isolation, it becomes expensive to assign priorities. For example, if pairwise combinations of \( n \) objects need to be considered, the worst-case overhead is \( O(n^2) \). Fortunately, in DTNs, a node spends most of its time between encounters (e.g., spends minutes or tens of minutes). Hence, the bulk of similarity processing can be done in between encounters, leading to efficient prioritization of forwarding when nodes meet. Much of proposed protocols design tries to minimize prioritization overhead.

To the end, we propose a set of redundancy minimization problems and the associated CAP protocols in different contexts. We further recognize that minimizing redundancy of data objects can be regarded as maximizing diversity of the same in the sense that the collection retains as much “dissimilar” or “different” (non-redundant) content as possible. In that, redundancy minimization problems are computationally analogous to diversity maximization problems. That’s we call them diversity-aware protocols. We introduce appropriate metrics for quantifying diversity of a content collection and propose a set of optimization problems where the associated metric
needs to be maximized subject to a variety of constraints (e.g., storage and bandwidth).

**Diversity-aware picture delivery services** We develop the following two services. At first, we propose PhotoNet, a picture collection service for situation awareness in a post disaster recovery operation that aims at collecting pictures from a disaster site and maximizes the geographic coverage of events that are manifested by those pictures. The service maximizes spatio-temporal diversity of collected pictures by favoring pictures from distant locations, and visually different pictures from the same location (Section 4.3). We devise associated CAP priority rules that assign appropriate priority to pictures while storing and transfer. Unfortunately, PhotoNet turns out to be susceptible to noises and outliers. This is because diversity maximization is at odd when the content collection has outliers. This is due to the fact that outliers are “different” from others hence causes their inclusion to be naturally favored over others. To fix that, we propose PhotoNet+, which tries to eliminate outliers from the collection (Chaper 5). This requires defining outlier-resilient diversity measure for content collection and devising the associated CAP priority rules that put outliers in lower priority with respect to other candidate objects. In the following, we describe these two scheme in more details.

4.3 PhotoNet: Picture Delivery for Situation Awareness

PhotoNet is a picture delivery service for situation awareness. The service aims at collecting and delivery of the most representative subset of pictures from a vast pool, where a significant portion of pictures are redundant. In particular, the service aims at maximizing awareness of locations that need attention. We call this metric event coverage. Hence, delivery of pictures from many different locations is preferred to delivery of many pictures from the same location. Similarly, delivery of dissimilar pictures from a given location (likely covering different events) is preferred to delivery of very similar pictures from that location. We show that by prioritizing image storage and
transmission depending on possible overlap with other images, one can make significantly better use of scarce resources thereby substantially improving overall event coverage at the sink. Overlap is estimated by comparing the visual content of images as well as their locations and timestamps. The service is made possible thanks to the proliferation of mobile devices with digital cameras, which makes deployment feasible.

The main contribution of PhotoNet lies in its message prioritization scheme (CAP) that aims to maximize delivered content diversity. By maximizing diversity, the network has a better chance at giving the sink the “big picture” quicker, as opposed to delivering lots of pictorial coverage of more populated locales and none on more isolated ones.

4.3.1 Design of PhotoNet

In DTNs, content is usually replicated on multiple devices creating a distributed in-network storage system. Each node carries pictures generated by the node itself or obtained from others via a replication process. Each stored picture is accompanied by meta-data that enables computing semantic redundancy. In PhotoNet, pictures are delivered against a query, issued by an authorized entity such as the command station. These queries are long-lived and stored in a query table. The default query is to obtain all pictures from all sources. In the rest of the discussion, we focus on the default query. To prioritize message transmission and storage, each node also runs CAP, the content-aware prioritization scheme. It generates the order in which pictures are replicated to other nodes and the order in which they are dropped from the storage when capacity is exceeded. The components of PhotoNet are shown in Figure 4.1.

Picture Organization and Naming  PhotoNet organizes pictures in a way that facilitates prioritization. A very general way of doing so is inspired by the content-centric networking paradigm, recently articulated by Van Jacobson [24]. In this paradigm, networks name content chunks, not machines, and queries express interest in content collections by name.

Following conventions of content-centric networking, all pictures managed by PhotoNet fall into a global naming structure that looks like a UNIX directory tree. Pictures taken by source nodes have names that place them
in one of the “directories”. For example, a rescue worker might take a picture, called ‘/rescue/pictures/volunteerA/pic1.jpg’. A fully quantified name refers to a unique item. Names can also be partially quantified to designate a collection of items that have the named prefix. Pictures have unique IDs, computed as a combination of device-dependent identifier (for example, IMEI of mobile devices, physical MAC address, or a long hash of the picture itself or a random string) and the local timestamp when the picture is generated. These IDs are used in detecting duplicate pictures in local storage, not as a part of their names.

Queries are expressed in terms of content identifiers or prefixes, such as ‘/rescue/pictures’, that define a subtree in the global naming structure. The collection of pictures that belongs in that subtree is said to match the query. This collection is denoted by \( \text{pics}(q) \) for a query \( q \). Each query is associated with a sink node, to which pictures of the query are due.

In PhotoNet, queries are long-lived. They can even be issued prior to mission launch at initialization (for example, before volunteers are deployed in the field). Queries flood the network. Source and intermediate nodes determine whether a particular picture in their buffer belongs to a particular query based on whether the queried name is a prefix of the picture’s name. Pictures matching each query are logically grouped by two types of linked lists, sorted by priority. One list is sorted by forwarding priority and the other list is by dropping priority. The objective from both priority orders is to reduce semantic redundancy based on content similarity. The priority order for forwarding and dropping are different because dropping priority is a function of local content only, whereas forwarding priority to an encountered node is a function of content on both nodes and needs to be computed on the fly when a node is at contact with another node. We describe the details of content prioritization in subsequent sections. Figure 4.1 describes, at a conceptual level, data structures used by PhotoNet.

4.3.2 Picture Representation and Semantic Distance

PhotoNet extracts features of pictures and expresses them as multi-dimensional vectors that we call *feature vectors*. We define a function, \( \text{map}(x) \), that maps a picture, \( x \), into a fixed-length vector in the feature vector space. Once
mapped, the semantic similarity between pictures is simply reflected in distance between the associated points in space. Items that have similar content and taken at nearby locations are closer in the feature vector space, whereas dissimilar objects or objects from different locations lie farther apart. Any suitable distance function, say Euclidean, can be used to measure distance or degrees of similarity among items.

The components of a feature vector associated with a picture can be attributed to a set of physical properties of the picture or some raw features extracted from the image data, such as color histograms. Since we are interested in information on geographic locales, meta-data such as location and time associated with data items also serves as features in the feature vector. Those features are very important in determining semantic redundancy. For example, different buildings may look alike in pictures. However, if their locations are different, then there is no semantic redundancy because these pictures carry information on different events. In addition, human assessment can also serve as useful input for the vector space. For example, the photographer can label his images with tags or keywords for organization purposes. In the current implementation human tags are not used.

To this end, we define \( \mathbf{x} = \{x_1, x_2, \ldots, x_k\} \) to be the feature vector of

Figure 4.1: PhotoNet architecture.
picture $x$. We use both spatio-temporal attributes and image-features. Thus, the vector can be expressed as: $\bar{x} = \{ t(x) \mid \ell(x) \mid \bar{f}(x) \}$, where $t(x)$ and $\ell(x)$ denote time and location of the picture (when and where originated) respectively and $\bar{f}(x)$ denotes the vector of visual image-features. In the current implementation, we simply use the color histogram.

We define semantic distance to be the level of dissimilarity between pictures: larger distance means highly dissimilar and shorter distance means fairly similar. In our application context, pictures are taken of physical events. Hence, we say that pictures are similar if they can be associated with the same physical event. Accordingly, pictures generated in two relatively remote locations (say, 1km apart) or at two largely different times (say, 6 hours apart) are “dissimilar” (high spatio-temporal distance). When pictures are closer in spatio-temporal space, their similarity is further decided by the distance in the image-feature space.

In our implementation of distance between pictures, we first use the time gap between two pictures as a binary decider. If it is beyond some threshold (say, 6 hours), we assume these two pictures belong to different events and assign a very large distance value. Otherwise, the distance is given by location and image-features. Semantic distance due to location is simply Euclidean distance between location coordinates. For image-features, we compute the distribution of colors in a picture, commonly known as color histogram. For a certain color (or a color bin), $i$, of picture $x$, let $f_i(x)$ be the fraction of pixels in the picture that has color $i$ or any color in that bin. Obviously, $\sum_i f_i(x) = 1$. We compute KL-divergence (Kullback-Leibler divergence) distance between the corresponding histograms for two pictures $x$ and $y$ as follows:

$$ \| \bar{f}(x) - \bar{f}(y) \| = \left[ \sum_i f_i(x) \log \frac{f_i(x)}{f_i(y)} + \sum_i f_i(y) \log \frac{f_i(y)}{f_i(x)} \right] $$

(4.1)

where $[u, v] = \max(u, v)$. By proper weighting, as shown in the evaluation (Figure 4.3), we can ensure that the numerical value of image-feature distance is very small compared to location distances. Therefore, location predominantly defines the semantic distance between pictures. When pictures originate near one another, the image-features further refine the semantic distance. Therefore, we use the following distance function:
\[ \| \mathbf{x} - \mathbf{y} \|^2 = \alpha \| \mathbf{t}(x) - \mathbf{t}(y) \|^2 + (1 - \alpha) \| \mathbf{f}(x) - \mathbf{f}(y) \|^2 \] (4.2)

where \( \alpha \) is a scaling factor and is given by: \( \alpha = \exp \left( -\frac{\tau^2}{\| \mathbf{t}(x) - \mathbf{t}(y) \|^2} \right) \). It indicates that image-features are ignored above a certain location threshold, but become increasingly more important below that threshold when the location distance itself is ignored. The parameter \( \tau \) defines this threshold and we use \( \tau = 100 \) meters. In the following, we use the same symbol to represent a picture and its feature vector, and use \( \| x - y \| \) to denote the corresponding distance.

The choice of color histogram as image-features requires an explanation. One reason of choosing histogram is that it is simple and efficient and works moderately well on normal conditions. More specifically because it is computationally less expensive and suitable for being implemented on handheld mobile devices (see implementation, Section 4.5.3). Obviously, color histogram can have false positives: two visibly different pictures (pertaining to different events) from the same location can be concluded as similar. It also fails to differentiate between images that are otherwise the same but the presence of a certain object in them. As an immediate fix, more advanced and sophisticated features, say SIFT (scale-invariant feature transforms) [25] or SURF (speeded up robust features) [26] can be used to measure similarity between images, although they are computationally expensive and heavy for cell phones.

Again, we use location information to measure semantic distances between images. In a very large geographic area, it may happen that images are originated from many different locations, and hence most images are uncorrelated. In our current scheme, we however consider the cases where the number of events is outnumbered by the number generated pictures by several order of magnitudes, which renders the reasoning among pictures for similarity important. The measure can be further improved if we apply some kind of proximity-based group discovery for users to coordinate among themselves. These issues would be investigated in future.
4.3.3 Diversity Measure of Picture Collection

The goal of PhotoNet is to maximize event coverage. It does so by maximizing the diversity of delivered content. This maximization requires a measure of diversity. Hence, given a set \( I \) of \( n \) pictures, its diversity, denoted as \( \Psi(I) \), is computed as the average of squares of pairwise distances between all pictures.

\[
\Psi(I) = \frac{\sum_{x,y \in I} \|x - y\|^2}{n \times (n - 1)}
\]  

(4.3)

Note that pictures that are farther from each other produce a larger \( \Psi(I) \) because they improve diversity, whereas data items that are clustered together produce a lower \( \Psi(I) \) because they cover a smaller region of the data space. To reduce semantic redundancy, the network should always make forwarding and dropping decisions that generate a higher \( \Psi(I) \) for a collection of pictures, \( I \).

It is also useful to measure the contribution of a given individual picture to the diversity measure of the collection. This quantity, denoted by \( \psi(x) \), is simply given by the average of square of distances with other pictures in the set, i.e., \( \psi(x) = \frac{1}{n-1} \sum_{y \neq x} \|x - y\|^2 \). Obviously, \( \Psi(I) = \frac{1}{n} \sum_x \psi(x) \).

4.4 CAP Rules for Dropping and Transferring Content

4.4.1 A Prioritized Dropping Policy

In cases where storage space becomes a bottleneck, a node may need to decide which picture to drop. A node never deletes pictures that the node itself produces (for which the node is the source). We assume that nodes have enough space to hold their own pictures, but the storage for pictures that are replicated from other nodes is limited. Hence, some of these pictures may be dropped due to storage constraints. The question is to which picture to drop when storage capacity is exceeded.

The notion of the diversity \( \Psi(I) \) of picture set, \( I \), offers an answer. Namely, pictures should be dropped in an order that maximizes the diversity of the
remaining set. Since different pictures have different length, it is best to normalize diversity by storage requirements. For example, removal of a long picture is preferred to removal of a short picture, if diversity of the remaining set is the same. Hence, we drop the picture that maximizes the diversity of the remaining set per byte stored. More formally, let the set of pictures stored locally at node $X$, that match query $q$, be called $\text{pics}_X(q)$. Let the total space needed to store $\text{pics}_X(q)$ be $S_{\text{total}}^q$ and let the size of picture $x$ be denoted by $s(x)$. For each query, $q$, the next picture to drop, $x$, among those locally stored pictures that match the query, $\text{pics}_X(q)$, is computed as follows:

$$x = \arg_{x \in \text{pics}_X(q)} \max \left( \frac{\Psi(\text{pics}_X(q) - \{x\})}{S_{\text{total}}^q - s(x)} \right)$$ (4.4)

In the presence of multiple queries, CAP first drops pictures that do not match any query. If no such picture exists, it chooses one query at a time and drops the picture computed from Equation (4.4). Observe that the order in which pictures will be dropped from a given set $\text{pics}_X(q)$ depends only on local information. This order can therefore be precomputed in advance. Since nodes in DTNs spend a lot of time between encounters when they apparently remain idle, there is enough time to compute the order in which pictures are to be dropped. The dropping policy pre-marks some content as deleted to create enough free buffer space to accept content from newly encountered nodes, and creates a single linked list in the order these pictures are to be removed. During an encounter, marked content is replaced if space is needed. After an encounter, the dropping policy pre-computes the order of deletion again. Locally generated content triggers periodic recomputation of the dropping order. The above algorithm implicitly assumes that the total amount of content delivered to a node during an encounter is not a significant fraction of its storage capacity. Hence, the odds that the newly received content is a better candidate for dropping than the pre-marked content is low. These odds are further reduced by the fact that the forwarding policy prioritizes content such that the most “useful” content is forwarded first. This mechanism is described next.

In case CAP needs to make dropping decisions on the fly upon a contact, an efficient implementation of computing dropping order can be made. Equation 4.4 suggests that pictures can be dropped at an ascending order of
ψ(x)/s(x), that is, the least diverse content first. When a particular picture, say y, is dropped, ψ(x)'s of remaining items are updated as follows: subtract \(\|x - y\|_2^2\) from \(ψ(x)\) (and normalize by the size of the collection; the same is added when a new picture is added to the set). Then, the remaining pictures are again sorted to find the candidate for the next drop. This can be implemented by a min heap that returns elements with minimum \(ψ(x)/s(x)\).

4.4.2 A Prioritized Replication Policy

In DTNs, upon a contact, a node decides which messages it needs to replicate to the other peer, in some particular order. The term replication is slightly different than traditional forwarding. Here, the sending node retains the copy of the message that it transfers to the peer. This allows the same message to be transferred onto another node, opportunistically increasing the chance that at least one of these messages would be eventually delivered to the destination. CAP follows a simple replication policy “most diverse content first”, that is, pictures that maximize diversity \(Ψ(I)\) of the receiver’s collection should be replicated first. In the following, we use the terms transfer, forward and replication interchangeably.

Hence, when node X meets Y, it needs to know exactly what pictures node Y currently holds for a given query so that X does not send similar content to Y. In other words, for each query, \(q\), node X should forward to Y the pictures that maximize \(Ψ(pics_Y(q))\) at Y. A naive approach could be that Y sends the feature vectors of all pictures in \(pics_Y(q)\) to X for each query \(q\). Hence, X could choose those pictures that, if forwarded, will maximize \(Ψ(pics_Y(q))\). Intuitively, these pictures are the most distant in its vector space from Y’s current picture set. Obviously, exchanging vectors for all stored pictures is costly, especially when the vector size is large and the number of pictures is many. This is also computationally expensive to compute distances from all pair of pictures. Instead, each node partitions its picture collection into clusters.

**Clustering Pictures** Every node, \(X\), clusters each of its picture collections, \(pics_X(q)\) (one per outstanding query) and computes the centroid of each cluster, called a *pivot*. Hence, only pivots need to be exchanged. The above clustering is a function of only local information on the node, and
hence can be done in free time in between encounters (i.e., before the node actually gets in contact with another). The clustering operation is done for each query in the query table. For a given number of pivots $k$, an optimal position for pivot points would be such that the sum of distances from non-pivot points to the nearest pivot is minimized over all other possible choices of pivot locations. This problem, known as $k$-mean clustering and reportedly NP-Hard, can be approximated by Lloyd’s algorithm [27]. Lloyd algorithm starts with a random $k$ pivots and then iteratively adjusts pivot locations based on assignment of pictures to their respective nearest pivot. The clustering stops when the diameter of each cluster reaches within some threshold. The query table at each node stores the list of computed pivots, called $P(q)$, for each query $q$. These pivots are computed over all pictures, $pics_X(q)$, at node $X$.

**Priority Order for Forwarding** Once pivots are computed, CAP’s forward ordering of pictures matching query $q$ is fairly simple. When two nodes meet, they exchange their pivot vectors for all queries. Let $P_X(q)$ and $P_Y(q)$ be the pivot vectors of nodes $X$ and $Y$ respectively, for query $q$. After these vectors are exchanged, node $X$ computes the possible diversity that a picture $x$ (matching query $q$) can introduce to the existing picture set at $Y$. This is computed as the average square distance from the picture to the pivot vectors:

$$\psi(x, P_Y(q)) = \frac{1}{|P_Y(q)|} \sum_{y \in P_Y(q)} \|x - y\|^2$$  \hspace{1cm} (4.5)

The picture that is farther away from the corresponding pivots of $Y$ introduces greater diversity than other pictures with smaller distances. This is however obtained at the cost of transferring the picture itself to the peer node, which costs (in terms of energy or bandwidth occupancy) in proportion to the size of the picture (the length of the message in bytes). Therefore, whichever picture produces the largest gain per byte, $\psi(x, P_Y(q))/s(x)$, is given the highest priority. When the pivot set $P_Y(q)$ is well understood from the context, we drop $P_Y(q)$ from the diversity expression (Equation 4.5), to write $\psi(x)$. 75
Algorithm 1 replicate-messages(Contact $c : X \rightarrow Y$
)

1: exchange query tables, i.e., $name(q)$’s
2: for each query $q$ in query table do
3: send $P_X(q)$ to $Y$, receive $P_Y(q)$ from $Y$
4: populate $pics_X(q)$
5: $P(q) = P_Y(q)$
6: if $P(q) = \emptyset$ then
7: $P(p) = \arg_{x,y} \max \|x - y\|$, $x, y \in pics_X(q)$
8: end if
9: end for
10: while connection $c$ persists do
11: pick the next query, $q$, in RR or WFQ manner
12: set $p = \arg_{x \in pics_X(q)} \max \psi(x)/s(x)$
13: send-picture($p$, $c$)
14: if picture $p$ is transferred then
15: $P(q) = P(q) \cup \{p\}$
16: end if
17: end while

A note on optimization and computation complexity  As an optimization, when computing pivots, we in fact compute them based on clustering the union of sets $pics_X(q) \cup pics_X^{past}(q)$, where the latter set denotes the pictures that match query $q$ that have been previously forwarded by $X$ to other nodes and erased. There is a good chance the query sink will receive these pictures. Hence, this retention of “previous memory” prevents forwarding semantically redundant objects to a destination if they arrive at the forwarder at different times. Otherwise, if the forwarder forgets such evicted content, it can end up subsequently forwarding similar content to the same destination, hence contributing to needless redundancy. To remember set $pics_X^{past}(q)$, When a message is deleted that has previously been forwarded to another node, CAP retains the feature vector of the deleted picture. These vectors can be eventually removed from $pics_X^{past}(q)$, although, in the current implementation, we do not expire such vectors. Feature vectors are considerably smaller in size than the original content. Therefore, they do not produce much storage overhead.

With these additional records, the computation complexity of prioritizing pictures in forwarding list (a sorted list) and dropping list (a min heap) becomes $O(n \log n)$ and $O(\log n)$ respectively, where $n = pics_X(q) \cup pics_X^{past}(q)$. The clustering operation, the most expensive routine and mostly done when
devices are not communicating (offline), is mainly dependent on the number of iterations required for convergence. At each iteration, the computation complexity is $O(n^2)$ (each item is assigned to the nearest cluster).

4.4.3 Additional Mechanisms and Implementation Issues

For more efficient replication, PhotoNet performs some additional work. While replicating onward, each picture contains a list of nodes that the picture has already passed through. In that case, the picture would not be replicated onto the same node again. Each node also maintains a list of picture IDs (not feature vectors) that have been delivered to the collection point for a given query. This list is exchanged when two nodes meet and is propagated into the network. When the list is updated upon a contact, the corresponding pictures are exempted from being replicated anymore since they are already delivered. They can still remain in the storage, because some future query may look for them. If not, they can be deleted from the store.

CAP realizes its own replication and dropping policy as it maintains its content. It can be implemented in one of two possible approaches. The first approach is to augment the underlying routing protocol with a callback routine that is invoked when the router needs to send or drop messages so that it chooses the most eligible one determined by CAP. Another approach is to implement CAP as an application overlay, in that nodes, when talk, talk to their CAP-instances. This requires the underlying routing layer to forward all received messages to the application layer, enabling CAP to implement its own routing and prioritization policies on top. We used the later approach.

4.4.4 Limitations and Possibilities of CAP

**Breadth vs Depth** CAP prefers breadth to depth, by passing more diverse pictures to the collection point. But the contrary preference might be more meaningful in some cases. Suppose, one region is particularly interesting and lots of pictures are generated from that location. In CAP, these pictures would be slowly given less priority, while delivering pictures from other non-interesting locations. It could make more sense if this particular region is
given more preference than others, because interesting things are happening here.

**Handling Noise and Outliers**  CAP prefers pictures to be scattered across the vector space allowing more dissimilar content to pass through. This is befitting to the objective of maximizing event coverage. However, it makes CAP vulnerable to noise and outliers because noise and outliers may be different from usual pictures, or may be taken at locales from which no other pictures are reported. CAP identifies them as dissimilar objects and assigns higher priority to them in the transmission queue. This can be remedied by penalizing transmission of messages from singleton clusters (i.e., clusters containing only one message) or considering “mass” around individual pictures in terms of the number of similar pictures nearby in the feature space. Higher mass represents higher support for similar pictures, whereas noisy ones, although diverse, may have lower mass around them.

**Heterogeneous Content**  Currently CAP works for pictures. But, the same architecture can be adopted for other content types, such as text, audio, and video, given an appropriate map function and an associated distance function. Since we adopt content-centric networking abstractions, content format should be implicit in its name. For example, ‘/rescue/audio/’ could refer to audio data, which may invoke the corresponding map and distance functions. CAP can then prioritize these content based on their distances, as like pictures.

### 4.5 Evaluation

We evaluate PhotoNet in a post-disaster situation assessment scenario. We use ONE simulator [13] to emulate a post-disaster rescue operations based on the post-disaster mobility (PDM) model [28]. Admittedly, reconstructing a realistic situation is hard in a simulator. Instead, we try to capture key aspects and elements of the scenario and compare all competing approaches on the same grounds. We compare PhotoNet with two other DTN protocols, namely Prophet [29] and SprayAndWait [12] (henceforth we refer to as Spray), and observe how our scheme improves performance in terms of
coverage-related performance metrics.

DTN routing protocols are of two main types: flooding-based and quota-based. Prophet, a flooding scheme, computes path metrics in terms of probability of delivery by using histories of encounters and directs message propagation toward the direction where probability of delivery only increases. Spray is a quota-based protocol that puts a pre-specified limit on the number of replicas per message. Both Prophet and Spray use FIFO transfer queue and drop-tail policy of dropping messages (drops the earliest message first). We chose these two protocols, one from each type, to compare with PhotoNet.

As an instance of a functional prototype, we also implemented the service on handheld Android phones (Google Nexus One). Since large scale communication as envisioned by the application scenario can hardly be emulated in such a small testbed, we better do communication experiments on simulation and show results of node level experiences from testbed. We also set a few parameters of the simulation from testbed measurements (Section 4.5.3). We leave a real large scale deployment of PhotoNet to future work.

4.5.1 Simulation Environment

There are two key elements of the simulation: i) simulating the mobility of agents, ii) simulating the generation of events.

Mobility Model PDM models movement of various agents, mainly humans and vehicles, in a post-disaster recovery situation on a city map. It first places a few neighborhoods scattered on the map and then puts houses and survivors in those neighborhoods. It then randomly places relief camps, police stations, command stations and other entities on the map (provided in a configuration file). After that, it deploys moving agents of four major types: center to center (recurrent back-and-forth motion of supplies between centers), rescue workers/volunteers (localized random motion within a neighborhood), cyclic patrols (recurrently patrolled paths through multiple neighborhoods), and emergency responses (vehicles from a center to random destinations and back). All static and mobile nodes are equipped with wireless routers capable of running DTN protocols.
PDM forms an aggregation tree for collecting pictures. Rescue workers and volunteers, instrumented with cameras, move in neighborhood areas and shoot pictures, and occasionally report to neighboring relief camps. Supply vehicles and police patrols visit neighborhoods. There are a few vehicles, called “data mules”, that visit the main command station and the relief camps at each neighborhood. These mules effectively carry pictures to the command station. Our experiments have 100 houses, 100 rescue workers in 5 neighborhoods, 5 relief centers with 5–10 supply vehicles, and 2–5 data mules. Limited number of data mules constitute the major communication bottlenecks in picture collection, whereas limited meeting times between moving devices (i.e., bandwidth) and on-device storage capacity are other bottlenecks.

*Generation of Events and Pictures*  PDM simulates movements of agents, but not events. We extend PDM to generate and report events (as pictures). To model events, we randomly choose a few locations on the map as event locations. These points are preferably at neighborhoods where events are likely to occur and agents are likely to visit. We then associate each location with a certain set of events that “occurred” at that location. Events, in this case, correspond to instances of damage, collapse, fires, or other hazards.

Each event is supposed to have a distinct pictorial appearance. Since we do not have real events happening in the simulation, we pre-take a set of real pictures of a few distinct landmarks and objects around our campus, and map each landmark to a certain event and a location in the map. We take several pictures for each landmark from different angles and zoom levels to mimic the reality that several users take the same picture differently. Figure 4.2 shows five sample pictures of event “fallen rocks”. We argue that there could be some events that are seen by many observers (popular events), whereas others may be less popular. To emulate this effect, we use Zipf distribution to determine the number of different images assigned to certain events for each neighborhood. We assign a popularity index (1 means highly popular) to each event and generate $\lceil \frac{n_{\text{max}}}{i} \rceil$ number of pictures for an event with an index $i$. We used $n_{\text{max}} = 50$.

When an agent happens to pass or stop by a certain event location, it randomly chooses one or more pictures from the pre-arranged set, as if it “took” pictures of this event and a message is created in the network. Once
taken, the picture is deleted from the set so that no other agents report exactly the same picture. Once stored, a certain preprocessing time needs to elapse (compressing, reducing the dimension and extracting features from the picture) before the picture is passed onto others upon contacts.

Figure 4.3: CDF of KL-distances among similar and dissimilar pictures.

We use color histogram and KL-divergence distances for image-features. Figure 4.3 shows the CDF of KL-distances among pictures used in the simulation for a total of 32, 64 and 128 color bins. We see that dissimilar pictures are further away from similar pictures in histogram space, which enables CAP to cluster them properly, if only image-features were used. We used 32 color bins, which makes meta-data (location, time and image-feature) overhead per picture quite small (8 + 8 + 32 × 4 = 144 bytes).

4.5.2 Performance Evaluation

Next, we evaluate PhotoNet with two DTN routing protocols, Prophet and Spray. Table 4.1 shows default parameters used in the simulation. We are interested in results in an operating condition when resources are so limited...
Table 4.1: Simulation parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage capacity</td>
<td>1–10MB</td>
<td>Picture size</td>
<td>100KB</td>
</tr>
<tr>
<td>Picture proc time</td>
<td>4s</td>
<td>Conn estab. time</td>
<td>2s</td>
</tr>
<tr>
<td>Trans radius</td>
<td>20m</td>
<td>Trans rate</td>
<td>250 KB/sec</td>
</tr>
<tr>
<td>Spray quota</td>
<td>50</td>
<td>Prophet const</td>
<td>(0.75, 80) [29]</td>
</tr>
</tbody>
</table>

that usual network performance is very poor. For example, in almost all experiments, if not otherwise shown explicitly, picture delivery ratio, the fraction of total pictures delivered over total generated, is very low, around 10%-20%. PhotoNet intends to serve as many diverse pictures as possible out of these very limited delivery. We focus on evaluating the performance of the default query (collecting all images from all sources).

We compare performance of PhotoNet to other protocols under resource constraints. To see that the underlying DTNs are constrained, consider a network of 5000 nodes (e.g., relief workers and volunteers in a disaster recovery scenario), generating 100KB pictures from head-mounted cameras at the rate of approximately 50 pictures per hour. If a data mule eventually delivers these pictures to the destination every four hours, the mule will need to have a storage capacity sufficient for $4 \times 50 \times 5000 = 10^6$ pictures, or about 100GB. This amount of storage is challenging, considering that the mule may be just a mobile handheld. To keep simulation time low, we do not generate thousands of pictures. Instead, we reduce the number of pictures as well as the assumed device storage capacity proportionally. To this end, we generate nearly 25 pictures per hour for a network of 100 nodes, reducing the above storage requirements to 10MB (to hold 100 pictures in lieu of $10^6$). On the same ground, in order to scale communication capacity, we set smaller radio transmission range (20 m) as well as low link transmission rate (sometimes 50KB/sec).

PhotoNet intends to collect as many different events (i.e., pictures) as possible. We define two performance metrics, event coverage and precision. Event coverage computes the fraction of total distinct events that have been successfully reported at the sink to the total number of distinct events generated. We say that an event is reported if at least one picture pertaining to that event has been delivered to the sink. Precision measures what fraction of delivered pictures were unique, that is, the first picture that contributed
to reporting an event. PhotoNet aims at achieving higher event coverage as well as higher precision. Note that higher precision means lower overhead.

Figure 4.4 shows the fraction of total events reported to those generated as a function of time for PhotoNet as well as for Prophet and Spray within some defined deadline, reportedly 10 hours. For this experiment, we generate all pictures at sometime around 1 hour, and see what fractions are delivered by the deadline. We observe that the PhotoNet reports more events (30–40% more) than regular Prophet and Spray in any given time. We also plot delivery ratio, the fraction of total pictures delivered to total generated, in every hour. It is observed that the delivery ratio of PhotoNet is slightly low (compared to Prophet), still its event coverage is higher than others. It is also to note that although delivery ratio is within 20%, PhotoNet delivers nearly 80% of total generated events. This is due to the prioritization scheme applied by PhotoNet.

The number of data mules connecting neighborhoods with the command station affects the connectivity of the deployed area, hence the picture collection efficiency. Figure 4.5(a, b) show event coverage and precision at varying number of data mules at storage capacity 5MB. We see that when the number of data mules is high, event coverage is moderately high for all protocols. With fewer mules, the coverage declines except for PhotoNet. Smaller number of mules causes less carrying capacity, thus more events failed to be reported to the sink. Since PhotoNet uses message prioritization (CAP), it
Figure 4.5: Event coverage and precision at (a, b) varying number of mules, (c, d) varying storage capacity.

delivers diverse content first utilizing bottleneck resources efficiently. Therefore, its event coverage does not decrease much with decreasing number of mules. But at certain point (say at 1 mule), poor connectivity dominates other constraints and PhotoNet’s event coverage also declines. As relay capacity becomes weak, PhotoNet’s precision rises whereas others remain quite the same. This is because CAP chooses diverse pictures to go first and consequently raises the ratio of the number of distinct pictures delivered to total delivered.

Figure 4.5(c, d) show the same set of results when storage capacity is varied. In case, storage becomes a bottleneck, more and more pictures would be dropped from nodes. Dropping pictures in some discriminate fashion would simply drop different events altogether possibly serving only most replicated popular ones. In contrast, PhotoNet gives priority to diverse content among stored pictures and drop most redundant (less diverse) content first. It thus holds as many different events as possible. Figure 4.5(c) depicts that at a high storage all protocols start at a good event coverage, but for others event
coverage eventually declines as storage becomes scare, but PhotoNet still offers higher event coverage as well as higher precision. At some extreme poor state though (at 1MB storage that can hold only 10 pictures), it also suffers.

Figure 4.6: Coverage/precision at varying link bandwidth and transmission range.

Figure 4.6(a) shows event coverage and precision at varying link bandwidth and transmission range. In these cases too, PhotoNet outperforms others both in coverage and precision. We observe that despite bandwidth gets low, event coverage is not affected that much. This is because in our mobility model there are a few static points where nodes stay for a while experiencing longer contacts. If not otherwise constrained by the storage capacity, this allows nodes to exchange their pictures where only prioritization does not help much.

4.5.3 Phone Implementation

We implemented PhotoNet as a small but functional prototype on a mobile testbed with a few Android phones. In our testbed implementation, phones are allowed to take pictures and communicate with other devices for exchanging pictures. To test the service, we visited various places and shot pictures using different phones. All pictures were tagged with GPS coordinates of places they were taken. We then set these phones to exchange pictures in an emulated DTN environment. This was done manually by repeatedly connecting and disconnecting pairs of devices via a special-purpose application GUI. We were interested in seeing that after a several rounds of exchange, the devices end up having pictures that are considerably diverse, when each device is allowed to store only a limited number of pictures.
Other than serving as a functional prototype of the PhotoNet service, the phone based implementation gives us node-level measurements (for example, time to extract image features). The implementation also helped determine some of the simulation parameters used in the earlier section (e.g., connection setting time). We present all computed timing values in Table 4.2. These activities are mainly offline operations that occur when nodes do not communicate with others.

Table 4.2: Various timing values on Android phones

<table>
<thead>
<tr>
<th>Component</th>
<th>Average time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taking picture, compress and store</td>
<td>2264.55 ms</td>
</tr>
<tr>
<td>Computing color histograms</td>
<td>1981.0 ms</td>
</tr>
<tr>
<td>Discovery and connection establishment</td>
<td>2656.76 ms</td>
</tr>
</tbody>
</table>

Once a picture is captured in our implementation, it is compressed into JPEG form, tagged with GPS location and then stored in the local storage. This compression is required because raw bitmap data is usually very large (average 1.21MB). Compressed images average 135KB, nearly 11% of the raw data. Once compressed, image-features (i.e., color histograms) are extracted from the captured pictures. This computation occurs only once. The resulting vector is then added to the application-level message header as meta information. We implemented feature extraction on an Android phone and computed the average time required to populate color histogram features. We show the average histogram computation times for pictures with different sizes in Table 4.3. Finally, the user is prompted to name the picture in the hierarchical content tree structure. The picture is now ready to be served against a query.

Table 4.3: Average feature extraction time from pictures

<table>
<thead>
<tr>
<th>Picture size</th>
<th>Avg. time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>360x480</td>
<td>719.7</td>
</tr>
<tr>
<td>600x800</td>
<td>1981.0</td>
</tr>
<tr>
<td>1200x1000</td>
<td>4898.2</td>
</tr>
</tbody>
</table>

In DTNs, nodes need to discover other nodes to do opportunistic communication. Once discovered, the associated devices establish connection between them. This discovery and connection establishment take some time, based on distance between devices, surrounding environmental conditions
and other factors. We measured the average time elapsed in discovery, pairing, and connection establishment time over bluetooth (the communication medium used in the current implementation) at each contact between a pair of Android phones. We manually pair each pair of phones beforehand and keep them listening on a connection socket. Once a pair of devices establishes a connection between them, the timing is recorded. We used this time as a connection establishment overhead in each contact in the simulation.

There is another important offline computation overhead in PhotoNet, which is to compute pivots. Pivots (per query) produce a summary of pictures on a node and are computed on all stored or ever replicated pictures (for a given query). We show the average clustering time for a set of pictures in Figure 4.7. It is shown that as the number of pictures increases, clustering time increases, somewhat non-linearly: more time for larger image set. This computation although costly happens only when nodes are otherwise idle.

Figure 4.7: Average clustering time at varying # of pictures on an Android phone
As we have described, PhotoNet tries to maximize diversity among stored pictures. Unfortunately, it turns out that PhotoNet is susceptible to noises and outliers. This is because diversity maximization is at odd when the content collection has noises and outliers that are not representative. The existence of outliers is quite natural in our scenario, since the participants may not always be entirely reliable. For example, not all returned pictures will be related to the task at hand. To conserve resources, one would like to minimize redundancy while eliminating the outliers. Note that, the objectives of reducing redundancy (thus maximizing diversity) and eliminating outliers are at odds. Outliers, by definition, are different from other pictures and hence, not redundant. We show that algorithms that minimize redundancy alone, i.e., PhotoNet, favor outliers as opposed to more representative content. The main contribution therefore lies in combining redundancy minimization with outlier elimination in participatory camera sensing networks. Towards that end, we propose a technique to identify outliers, propose a new metric for content diversity, and develop a new rule for content prioritization, where outliers receive lower priority, while diversity is maintained\textsuperscript{1}.

It is worth noting, at this point, that outlier elimination is not always a goal in a participatory camera network. In some applications, such as anomaly detection, outliers are in fact what carries the relevant information. For example, an in-store security camera might report the same view all night, except when an intruder breaks in. A frame with the intruder in view might be the outlier, but it is also the frame that contains the most interesting information. We, however, consider a different type of applications, where a community of users document relatively static conditions in the environment.

such as damage or points of interest. In such cases, one is not looking for anomalies in reporting, but rather for \textit{representative} depiction.

5.1 Basic Idea of PhotoNet$^+$

As like PhotoNet, PhotoNet$^+$ also tries to maximize \textit{event coverage}, that is to report maximum number of events that need attention. Now the question arises what constitutes an event. Generally speaking, an event is something whose picture is taken by a user. We argue that locations have a very vital role to play in this regard. We assume that devices are GPS-equipped and they label their captured pictures with the current location of the device when the picture is taken. Unlike other data content, e.g., text messages, this attestation of physical location to picture is very firm. For instance, if someone report “I see a burning house” via a text message from a certain location (as recorded by the device itself at that time), it does not necessarily mean that the user was physically there (in front of the burning house) to make the claim. The user can somewhere else writing this. But for pictures, this is not possible. One cannot generate a picture of some event without being there in the first place (unless one takes the same picture from someone else, or tamper location information, or produce fabricated pictures, which are all quite unlikely). Since users are all humans, we can safely rely on their intelligence and judgment in accessing the merit of the scene they are currently looking at. Therefore, the location attached with a captured picture strictly affirms the assertion that there was indeed an “event” happened at that location that caught the attention of that particular user. The event coverage then simply leads to collecting pictures from as many different locations as possible.

The main concern of the above understanding is that we do not really know exactly what happened at that location as a manifestation of an event. Is this something interesting or important? What attracted the user to take the picture and what does the picture actually contain? The hard part is that the devices cannot \textit{read} pictures to extract semantic details contained in it to infer the event associated with it (if any). As an easy trait, we rely on “corroboration of users” (also known as \textit{wisdom of crowds}), which states that if multiple users (without collaboration among themselves) capture the
same picture from the same location, then there is something interesting occurred at that location. In practice, however, multiple agents can never take the same exact picture, but pictures can be visually very similar. In that case, the event can be considered as an important one, otherwise it is treated as an outlier. This is the core technique we leverage to detect events and to maximize event coverage, while eliminating outliers. We describe the technique in more detail in the following subsections.

5.2 Choosing Representative Picture Subset

In order to select a representative subset of pictures that maximizes coverage using the fewest pictures, we focus on increasing diversity among the selected pictures to minimize overlap. One might be tempted to also favor large panoramic pictures, since they presumably offer more coverage. We do not take that route since often information is contained in the detail (e.g., a close-up of a crack in the wall might indicate a damaged building, but the crack may not show in a wide panoramic view). Since we do not know what the participants’ regard at the important information in the picture, we take the more conservative approach of simply removing redundancy as a safer way to offer coverage with fewer pictures. To reduce unrepresentative outliers, we further refrain from selecting pictures that are not corroborated by others.

To implement the above selection mechanisms, we define a distance function that measures the level of similarity between pictures based on the degree of match in their visual features and in metadata between them. An important piece of metadata is location. For example, two buildings may appear visually similar, but if they happen to be in different locations, they must be different. Given an appropriate logical distance function to measure redundancy with, the diversity of a picture collection depends on distances between individual pictures in the collection. We attempt to maximize diversity while removing outliers. Some outliers can be detected at the source. For example, a picture that is blurry or otherwise of poor quality may not be useful, and hence can be discarded. Such quality problems can be handled easily by the user or by existing vision techniques applied in an automated fashion at the source, and are not the topic of this work.

Instead, we attempt to infer relevance based on similarity of the picture to
others, considering both geographic attributes and visual image features. The idea is that (versions of) more relevant scenes to the participatory sensing application will generally be photographed by more sources. By “scene”, here, we mean a visual observation, such as the observation of a damaged bridge, a collapsed building, a blocked road, a fire, a car accident, a traffic jam, or an interesting person. An explicit goal is to estimate relevance \textit{without having to understand the semantics} of what is in a picture, since this would be very complex, application-specific, and beyond the purview of a general service. Note that, short of truly understanding each picture, and short of understanding the application’s mission, there is no error-proof way of assessing relevance of a picture to the mission. Hence, by necessity, we have to settle for an imperfect scheme in exchange for a higher degree of application-independence. Our contribution, therefore, lies in proposing and assessing the performance of one such scheme empirically based on actual photographs and a representative application scenario. Evaluation shows that, despite its limitations, our scheme offers a significant improvement over entirely content-agnostic networks.

5.2.1 Picture Representation and Similarity Distance

A hallmark of our scheme lies in its separation between application-specific notions of “similarity” between objects, and the generic diversity-maximizing and outlier-eliminating prioritization scheme. The core of that separation lies in the definition of a distance metric, $d(x, y)$, between content objects $x$ and $y$ to denote their degree of similarity. Our scheme does not assume any specific distance metric. In other words, it is general in that it does not care how $d(x, y)$ is computed. The definition of the distance metric is, in fact, the primary way our scheme can be customized for the needs of a particular mission or application scenario. Given a distance metric, the scheme can perform better or worse, depending on how representative this metric is of the amount of information overlap between content objects. The metric should yield a lower distance when there is more overlap.

For the application at hand, we argue that location plays an important role in defining logical distance. When the geographic distance between two pictures is beyond some threshold, say 200m, they are physically far enough
apart that they are likely to be of different scenes, regardless of their visual
similarity (e.g., if they are both pictures of burning cars, they are likely to
involve different cars, even if the cars looked similar). Conversely, for pictures
taken from almost the same location, it is the visual features of the respective
images that give the best clue on whether they are of the same scene or not.
Let us define $T$ as the distance threshold beyond which we can safely assume
that the pictures taken are of different scenes. As a means of normalization,
when pictures are geographically distant, we set their logical distance to a
value greater than 1; otherwise, we make it smaller than 1, in which case the
logical distance should be dominated by visual difference.

Let $d_l(x, y)$ be the geographic distance between locations of picture $x$ and
$y$, and let $d_v(x, y)$ be their visual distance based on image features. We
normalize the visual distance so that $0 < d_v(x, y) \leq 1$. We then combine
visual similarity and location into a single uniform logical distance metric
using the following expression:

$$d(x, y) = \begin{cases} \frac{d_l(x, y)}{T} & \text{if } d_l(x, y) > T \\ d_v(x, y) & \text{otherwise} \end{cases} \quad (5.1)$$

A pair of pictures are said to be \textit{geographically collocated}, if the geographic
distance between them is less than $T$, that is, $d(x, y) \leq 1$. Obviously, $T$
depends on the application. For example, in a city, pictures taken more
than a few blocks apart will likely be different so $T$ is of the order of city
blocks. For an indoor deployment inside a building, $T$ might correspond to
the size of a single room. In an exhibition setup, say in a museum, $T$ can be
even smaller (e.g., of the order of the neighborhood of a single exhibit in a
room), because users’ interest naturally clusters around different objects of
the granularity of single exhibits. We assume that appropriate localization
techniques exist depending on the type of deployment (e.g., GPS for outdoor
deployment and localization via Wi-Fi [30, 31] for indoor setup) by which
pictures would be labeled by the location information. Note that the use of
location information is mainly to arbitrate whether two pictures are captured
in a nearby location. So as long as this inference can be made, some degree
of inaccuracies can be tolerated.
5.3 Cluster-based Diversity of Picture Collection

The distance metric $d(x, y)$ allows for objects to be represented as points in a multidimensional logical space, where the proximity of points designates the similarity between the corresponding objects. If two points lie very close to each other, they have information overlap, which makes them partially redundant. The purpose of diversity maximization is to reduce overlap among selected objects, subject to resource constraints (e.g., limited storage size). This in turn implies choosing points that are distant in logical space.

We further assume that there exists a certain logical distance threshold beyond which there is no information overlap. That means, each object has an extended scope of similarity around itself in the space. Let this distance be $\tau$. Hence, it is useful to imagine that each object logically covers a hyper-sphere with radius $\tau$ so that the spheres of two objects overlap when their distance is smaller than $2\tau$. Overlapping spheres indicates existence of shared information between objects. The volume of a sphere is called the coverage of a given object. For an $n$-dimensional feature space, this volume is proportional to $\tau^n$.

Note that, due to overlap, the total coverage of a set of objects is generally less than the sum of the coverages of the individual objects. The total coverage of all objects in a set can thus be treated as a quantitative estimation of the diversity of the set. The diversity maximization problem is then to chose a subset of objects whose total coverage is maximum, subject to some aggregate resource constraint (e.g., storage capacity) that limits the number of objects chosen. Figure 5.1 illustrates an example case for a 2-dimensional space.

In practice, pictures taken by participants would typically fall into groups (each group representing pictures of the same scene at the same place), such that logical distances between pictures within the same group (or cluster) are much smaller than those among different groups. This naturally leads to partitioning objects into a set of clusters, so that similar objects are grouped into the same cluster.

Coverage of a cluster follows two simple properties. First, the coverage is non-decreasing, in the sense that as objects are added to a cluster, coverage can only increase (or stay the same). Second, it has a declining marginal gain in that the expected additional coverage from adding another object to
the cluster declines as the size of the cluster grows (because spheres become more and more overlapped). Since the cluster is ultimately bounded in size, the infinite sum of all such increments is bounded. It is therefore useful to approximate this total cluster coverage, $CC_k$, for a cluster of $k$ objects, by a geometric series of the form:

$$CC_k = CC_1(1 + \lambda + \lambda^2 + \cdots + \lambda^{k-1}) \quad (5.2)$$

where $\lambda < 1$. To compute a suitable value for $\lambda$ in the above equation, it is useful to consider the infinite sum of the series. That is to say, it is useful to compute the coverage achieved in the limit, when the cluster size is very large.

Towards that end, consider a clustering algorithm that ensures that no two objects in the cluster are more than $2\beta$ distance apart. Since all individual object coverage spheres are of radius $\tau$ and all objects in a cluster are within distance $\beta$, the total volume covered by all objects inside a cluster can never exceed the volume of a sphere of diameter $2(\tau + \beta)$, no matter how many objects we put into the cluster. Since coverage grows with volume, which grows with sphere diameter, raised to the power of the number of dimensions, in an $n$-dimensional space, the cluster can cover a volume that is at most $(\frac{\tau + \beta}{\tau})^n$ the volume covered by a single object. In other words:

$$\frac{CC_\infty}{CC_1} = \left(\frac{\tau + \beta}{\tau}\right)^n \quad (5.3)$$
From Equation (5.2) and Equation (5.3), we get:

\[
\left(\frac{\tau + \beta}{\tau}\right)^n = 1 + \lambda + \lambda^2 + \cdots \infty \\
= \frac{1}{1 - \lambda}
\]  

(5.4)

From which:

\[
\lambda = 1 - \left(\frac{\tau}{\tau + \beta}\right)^n
\]  

(5.5)

Now, we can easily extend the notion of coverage to the entire object collection. Let collection $X$ contain $l$ clusters. Assuming clusters themselves are far enough apart from one another, the total coverage of all clusters is simply the sum of coverage of individual clusters. Let $s(c)$ be the size of cluster $c$. Therefore, the total coverage, that is, diversity, $\Psi(X)$ of collection $X$, is estimated by:

\[
\Psi(X) = \sum_{c=1}^{l} \sum_{i=0}^{s(c)-1} \lambda^i = \sum_{c=1}^{l} \frac{1 - \lambda^{s(c)}}{1 - \lambda}
\]  

(5.6)

What remains is to show how the value of $\tau$ and $\beta$ are chosen. First, since objects more than distance $2\tau$ apart are considered independent, it is useful to use the same threshold for clustering as well. In other words, we set $\beta = \tau$. Arguably, we want clusters to be formed among similar looking pictures originated from the same geographic area. Pictures from different locations, even if they look similar, should fall into different clusters. According to our definition of distance between pictures (Equation 5.1), we need to set $\tau < 1$. Now the question is what value of visual distance makes two pictures look alike. This depends on what visual attributes are used to determine similarity between pictures, which calls for experiments on our picture dataset. In evaluation (Section 5.5.1), we show that distance less than 0.25 happen to be a good threshold. We therefore choose $\tau = 0.25$. 

95
Table 5.1: Various diversity metrics used by biologists and ecologists

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shannon entropy</td>
<td>$-\sum P_i \log P_i$</td>
</tr>
<tr>
<td>Paired Shannon entropy</td>
<td>$-\sum P_i \log P_i - \sum (1 - P_i) \log (1 - P_i)$</td>
</tr>
<tr>
<td>Gini-Simpson index</td>
<td>$1 - \sum P_i^2$</td>
</tr>
<tr>
<td>Renyi $\alpha$-entropy</td>
<td>$(1 - \alpha)^{-1} \log(\sum P_i^\alpha)$</td>
</tr>
<tr>
<td>Havdara-Charavat entropy</td>
<td>$(\alpha - 1)^{-1} (1 - \sum P_i^\alpha)$</td>
</tr>
<tr>
<td>Hill numbers</td>
<td>$\left(\sum P_i^\alpha\right)^{1/\alpha}$, $q \geq 0$</td>
</tr>
<tr>
<td>Rao quadratic entropy</td>
<td>$\sum \sum P_i P_j d_{ij}$</td>
</tr>
</tbody>
</table>

5.3.1 Other Possible Diversity Metrics

At this point, it would be appropriate to revisit a few other popular diversity metrics available in literature, mainly in the domain of ecology and biological science [32, 33, 34]. Biologists use various measures of bio-diversity index to quantify diversity present in a species population. These indices are defined in terms of richness (the number of different species present in the population) and relative abundance of species (the number of individuals in a given species with respect to the total population). Let $P_i$ denote the abundance of a species, $i$ (in our context, $i$ is a cluster). Namely, $P_i = \frac{s_i}{n}$. Followings are some popular diversity indices used in estimating bio-diversity of a species collection:

Interestingly, our diversity metric, $\sum \frac{1 - \lambda s_i}{1 - \lambda}$, shares couple of important properties with some of the above. For example, as like ours, both Shannon and Gini-Simpson result in maximum diversity for a given number of clusters when all clusters have the equal number of objects. Renyi $\alpha$-entropy and Hill numbers are generalized expressions from which other diversity indices can be derived (e.g., Renyi entropy becomes Shannon entropy when $\alpha = 1$).

Unfortunately, none of these diversity metrics deem appropriate for our application. The problem lies in the fact that maximizing these diversity metrics does not necessarily give us the best object collection we expect. We postulate that $\Psi(X)$, as a measure of diversity, should have a set of quantitative properties in our application context. These are namely, *monotone-increasing* and *non-increasing return* property.

Monotone-increasing property refers to the fact that diversity can never decrease as more objects are added to an existing collection. In our application context, this should hold, because collecting more pictures, as long as storage capacity permits, can never be discouraged. More formally, for two
collection set $S$ and $T$:

$$\Psi(T) \geq \Psi(S), \quad \text{if } S \subset T \tag{5.7}$$

Although addition of each object to the collection increases diversity, the marginal gain (return) is however declining. As we can see from our PhotoNet application, adding a new picture to a already crowded cluster has declining gain unless it introduces a new cluster. The second property refers to this. Formally, adding a new object to larger set has smaller or equal return than adding the same to a smaller subset of the earlier. Mathematically, let $S \subset T$ and $S + d$ denote $S \cup \{d\}$. Then, the following holds:

$$\Psi(T + d) - \Psi(T) \leq \Psi(s + d) - \Psi(S), \quad \forall d \in T \tag{5.8}$$

The equality holds if the new item $d$ introduces a new cluster, otherwise the function is strictly declining. This property is known as submodular property in Metroid theory, which helps to design greedy solution to certain set of optimization problems. We can easily show by an easy induction that our conceptual notion of diversity in the form of space coverage as well as the diversity metric, defined by Equation 5.6, holds the above two properties.

None of the diversity metrics, however, shown in Table 5.1 have the above two properties. Let us check for Entropy, the most popular diversity metric. Suppose there are 10 objects in two equal groups, hence, $E = 2 \times \frac{5}{10} \ln \left(\frac{5}{10}\right) = \ln 2 = 0.6930$. Let add a new object to the collection, which joins to one of the clusters. Now, $E$ becomes $\frac{5}{11} \ln \left(\frac{11}{5}\right) + \frac{6}{11} \ln \left(\frac{11}{6}\right) = 0.6890$, which is a decline from the earlier, even though a new item is added. The same holds for other metrics. There has also been evidence that maximizing Quadratic Entropy (Rao’s Entropy) sometimes does not favor including new objects, which is referred to as lower richness problem [35].

We, therefore, use the diversity metric, $\Psi(X)$, expressed by Equation 5.6, to quantify diversity of a picture collection. Each node tries to maximize $\Psi(X)$ as it maintains its picture collection in order to hold as many diverse pictures as possible subject storage constraints. But, not all pictures are equally relevant to the end collection. Some are less representative, hence
outliers, which need to be eliminated. Below, we describe how outliers are identified and handled.

### 5.4 Outlier Resilient Diversity Maximization

It turns out that clustering offers an elegant way of separating the concern of outlier detection from the concern of diversity maximization. Intuitively, by assigning appropriate *relevance weights* to clusters, we can first get rid of low-ranked clusters (the outliers) to address relevance, then collect objects from the remaining clusters, thereby maximizing diversity, as per Equation (5.6), for only non-outlier clusters. In that sense, relevance weights are binary; a cluster is either an outlier or not. In the following, we explore the notion of outliers and relevance weights more closely.

**Outliers versus Rare Items** It is good to remind the reader at this point that an explicit design decision we make (for the sake of efficiency) is to refrain from techniques that rely on understanding picture semantics in order to determine relevance. Short of having such an understanding, we can only approximately estimate relevance, which we do from the behavior of data collection agents themselves. Presumably, they are motivated to collect relevant information. Hence, if more sources report an observation, it is more likely that the observation is relevant. With that in mind, outlier detection may seem very easy. For example, singleton clusters (i.e., those that have only one member) can be treated as outliers. This approach, however, is not always appropriate. Sometimes items may be isolated not because they are irrelevant and do not generate interest, but rather because they are in the vicinity of only very few observers. If there were more people in their vicinity, more pictures may have been taken of them. Hence, some consideration to the level of isolation of the *location* of pictures needs to be made in outlier determination. Intuitively, a scene should be considered an outlier not only because it is different but because others who are present at the scene are not taking pictures of it.

To define outliers, we borrow a terminology from the data mining community, called *spatial outliers*. Due to Shenkhar *et al.* [36], a spatial outlier is a *spatially referenced object whose non-spatial attribute values are significantly*
different from those of other spatially referenced objects in its spatial neighborhood. Correspondingly in our context, a picture is treated as an outlier, if it is geographically collocated with a popular picture set, but is visually significantly different from the group. For example, many users took a picture of a damaged house in a certain area, but one of them took a picture of something else which is different than the damaged building, while remaining geographically nearby. This picture would then be treated as an outlier, since it somehow did not trigger the curiosity of the other individuals in the same area. In contrast, if an isolated picture is reported from a location and no other pictures are taken at the same location, then it is not treated as an outlier because we do not have enough evidence to say it is irrelevant. Instead, we regard it as a rare item that simply has not been found by many observers. With that in mind, we introduce our relevance score that measures the relevance of an item consistently with the above definition.

**Relevance Weights of Clusters**  Relevance weight of a cluster is computed as the fraction of pictures that the cluster represents compared to the total number of pictures that are originated in the same geographic area.

A cluster represents all similar pictures (known to the node) from the same location. The number of all these pictures is called the estimated size of the cluster. If the size of a cluster is significantly smaller than the same sizes of other clusters in the same geographic area, then the cluster is likely to be an outlier. It indicates that not many sources were interested in recording that observation compared to others happening in the same location. Consideration of all objects known to the node, rather than only locally stored objects is important, because it allows different nodes, particularly between two communicating nodes, to agree on what they treat as outliers. This information is easy to collect via gossip among nodes. We revisit this issue when we describe our object transfer protocol in the subsequent section.

We use the standard z-statistic to determine outliers. We compute z-score of a cluster, denoted as \( z(c) \), as follows:

\[
    z(c) = \frac{es(c) - \bar{es}}{S/\sqrt{m}} 
\]

where \( es(c) \) is the estimated size of cluster \( c \), \( \bar{es} \) is the average estimated size of \( m \) geo-collocated clusters around \( c \) and \( S \) is the standard deviation of
those sizes. A cluster is treated as an outlier if its estimated size, $es(c)$, is very small and $z(c) < \epsilon$, for some threshold $\epsilon < 0$. The value of $\epsilon$ affects the accuracy of detecting outliers. Smaller $\epsilon$ values can lead to false positives (outliers are not detected) and larger $\epsilon$ leads to false negatives (others are detected as outliers), and both are detrimental to the end collection. In evaluation, we show the sensitivity of $\epsilon$ on outlier detection.

5.4.1 Outlier Resilient Drop and Transfer Rules

The main contribution of our scheme lies in implementing diversity maximization and outlier elimination as a content prioritization scheme that decides (i) the order in which objects need to be dropped on a node when storage is exceeded, and (ii) the order in which two nodes exchange content, when a connection between them is established. Objects need to be clustered as they arrive at a node. We describe the clustering process followed by the two prioritization schemes.

Online Clustering of Pictures

We use an online agglomerative clustering technique, proposed in [37], which incrementally adds new objects to existing clusters (as well as creates new clusters and splits earlier ones). We know that within a cluster all objects are within distance $\tau$ from one another. In that, the distance from the new object to all earlier objects need to be computed, which is somewhat costly to perform per arriving object. Instead, each cluster designates a representative object, called centroid object, and the distance to the centroid object is computed. If this distance is smaller than $\tau/2$, the newly object is put to the corresponding cluster. If there are multiple such clusters, it is assigned to the nearest one. If there is none, the object itself becomes a new cluster.

For a cluster $c$, the centroid object is denoted by $\mu(c)$. For each object $x$ in the cluster, we define $\delta(x)$ as its average distance from other objects in the cluster. The object with the smallest $\delta$ is chosen as the centroid object. We also sort objects inside a cluster in the ascending order of their $\delta$’s so that the centroid object becomes the highest ranked one followed by others (ties are broken arbitrarily). Let $r(x)$ denote the rank of object $x$ in its cluster.
Obviously, \( r(\mu(c)) = 0 \) and \( 0 \leq r(x) < s(c) \). Ranking is used when objects from a cluster are chosen one after another.

As objects are added and deleted from clusters, some objects may violate the clustering rule (distance becomes greater than \( \tau \)). This requires some reshuffling among clusters once in a while. This re-clustering operation is, however, costly in terms of computations. In PhotoNet\(^+\), this operation is executed offline when nodes are not in communication with another node.

Prioritized Dropping of Pictures

When the storage of a node becomes full, some earlier stored objects need to be dropped. While dropping objects, the dropping policy tries to preserve the diversity of the collection as much as possible, while also being resilient to outliers.

All clusters are divided into outlier and non-outlier clusters. As argued earlier, clusters with smaller \( z(c) \) are most likely to be outliers and hence are treated as such. The order for picture dropping is computed as follows. First, the lowest ranked outlier cluster is found and the lowest ranked object is dropped from it. This continues until no outlier clusters remain. After outliers are eliminated, the algorithm switches to improving diversity, which requires maximizing \( \psi(X) \) for non-outlier objects. From Equation (5.6), we have:

\[
\Psi(X) = \sum_{c=1}^{l} \sum_{i=0}^{s(c)-1} \lambda^i \\
= \sum_{c=1}^{l} \sum_{x \in c} \lambda^{r(x)} \\
= \sum_{x \in X} \lambda^{r(x)}
\]

(5.10)

That means, the object with the least \( \lambda^{r(x)} \) value, i.e, the largest \( r(x) \), should be dropped first, because it causes the least amount of decrement to \( \Psi(X) \). In other words, the lowest ranked object from the largest cluster is dropped. Algorithm 2 gives the pseudo-code of computing the drop order. The worst case running time is in the order of the number of clusters (finding
the largest cluster).

More appropriately, a sorted list of objects can be maintained in the ascending order of their $r(x)$’s, and objects are dropped from the tail of this list whenever required. When nodes drop objects, they drop their object content only (data payload), but store their metadata (feature vectors) for further use, such as outlier detection. Assuming feature vectors are very small compared to actual content, it does not add significant storage overload at each node.

**Algorithm 2** get-next-picture-to-drop()

Let $C$ be the set of clusters at the node
Let $x$ be the object that would be dropped next
Find cluster $c^+$ with the smallest $z(c)$
if $z(c^+) < 0$ then
  /* $c^+$ is an outlier */
  $x = \arg \min_{x \in c^+} \lambda^r(x)$
else
  Find the largest non-outlier cluster, $c^+$, in $C$
  $x = \arg \min_{x \in c^+} \lambda^r(x)$
end if
return $x$

Prioritized Transfer of Pictures

When two nodes establish a connection, they “sync” their content. In a flooding protocol, this would be achieved by exchanging all pictures that one node has but the other does not, such that both end up with the same set of pictures after the exchange. This, however, would be wasteful in resource consumption. Instead, we aim to exchange only representative content, suppressing both redundancy and outliers.

Each node maintains meta information of all pictures it ever encountered or stored. At the beginning of a connection, nodes exchange this list and update the estimated sizes and $z$-scores of their respective clusters, as described earlier. Based on $z$-scores, only non-outlier objects are considered first for transferring onto the peer node. Each one of the two nodes then determines the order at which it should be transferring objects so that the diversity at the other end is maximized.

Let node $A$ meet $B$ and $A$ be the one who is taking the transfer decision.
The same happens at $B$. At first, the centroid objects from those clusters of $A$ are sent that would create new clusters at $B$. This is because this would cause the highest increment in $B$’s diversity. After that, $B$ now has all clusters that $A$ has. Next, for each cluster $c$ at $A$, a corresponding cluster at $B$, denoted by $g(c)$, is identified to which objects from $c$ will be joining (i.e., distance < $\tau$). If an object from $c$ is sent to $B$, it would increase $B$’s diversity by $\Delta(c) = \lambda^{s(g(c))}$. But the next successive objects from the same cluster would have declining gain, each time multiplied by $\lambda$ with the earlier sent object (ordered by their ranks). So, the diversity increment at $B$ due to the transfer of an individual object, $x$, from $A$ is: $\Delta(x) = \lambda^{s(g(c))} + r(x')$. Once $\Delta$’s for all objects are computed, $A$ transfers objects in the descending order of their $\Delta(x)$’s. If pictures have large variation is their sizes, for best utilization of transfer opportunity, the value can be normalized by the size of the picture. Once all non-outlier clusters are considered, outlier clusters are considered, if transfer opportunity still allows sending more.

Algorithm 3 shows the transfer routine. Finding $g(c)$ for each cluster requires $O(l_B)$ computations ($B$ has $l_B$ clusters), so a total of $O(l_A l_B)$ computations. Computing $\Delta(x)$ per object then takes an iteration over the entire collection. So, the total running time of Algorithm 3 is $O(n_A l_B) \approx O(n + l^2)$ for a collection of $n$ pictures with $l$ clusters.

One last concern is the storage of metadata. Recall that each node attempts to store metadata of all pictures that it comes to know from other nodes, even though it may not store them all (it stores only a representative subset). When the number of pictures generated in the network becomes high, the volume of these metadata also rises. This may lead to an extra overhead of exchanging them. In order to reduce the metadata volume, we partition all these metadata into smaller clusters based on their distances, just like stored objects. While exchanging metadata, nodes then send only one representative item per cluster, called pivot, with the associated object IDs in that cluster. Pivots efficiently summarize the metadata of all objects known to a node. When pivots are exchanged between two nodes, both of the nodes check whether they have the same set of pivots (by measuring distances between them). If not, they update their current metadata clusters and pivots accordingly. Recent results, such as [38] that used bloom filters, can also be investigated in this regard.
Algorithm 3 compute-transfer-order(Contact c)

Let $c$ be a meeting between node $A$ and $B$

Let $A$ and $B$ be the set of clusters at $A$ and $B$

for all $c \in A$ and $c$ is not an outlier do
  Compute $g(c) = \arg \min_{b \in B} d(c, b)$, where $d(c, b) < \tau$
  Set $s(g(c)) = 0$, if $g(c)$ does not exist
  for all $x \in c$ do
    $\Delta(x) = \lambda^s(g(c)) + r(x)$
  end for
end for

Transfer objects in the descending order of $\Delta(x)$

5.5 Evaluation

We simulate PhotoNet+ in the ONE [13] simulator for a post-disaster rescue mission. In this setting, the underlying network is DTN, where the simulated nodes (mainly rescue workers and volunteers) carry cameras, visit places, shoot pictures of interest, and exchange these pictures, when they meet, ultimately to pass them to a central command station. Most of the simulation setting is the same from PhotoNet. We compare our results with PhotoNet to demonstrate that the elimination of outliers is important for any diversity-aware content delivery service. We surveyed related literature and could not find any other suitable protocol other than PhotoNet that address this problem. We also implemented the service on Android phones. Since we cannot produce a large scale physical network envisioned by the rescue mission, we limit our experiments with phones only to show various timing results performed on devices. We defer building a fully deployed service and conducting experiments involving real humans as future work.

5.5.1 Simulation Environment

We use the Post-Disaster Mobility (PDM) model (Chapter 7) to simulate a participatory sensing mission in a hypothetical town. PDM uses a map file to generate streets in the simulated area, such that mobile agents use streets for moving between destination points. PDM randomly locates a couple of neighborhoods with houses and puts service stations, such as rescue centers, relief camps, and police stations (specified in a configuration file) in
the map. It also places a central command station located far away from the neighborhoods. Four types of mobile agents are deployed: (i) vehicles that move back and forth between service stations, (ii) rescue workers and volunteers (mainly responsible for taking pictures) who roam around inside a given neighborhood and occasionally report to the nearby service stations, (iii) regular police patrols that visit neighborhoods, and (iv) a few data mules that commute between the command station and the different service stations in distant neighborhoods. We create 5 neighborhoods, 10–15 service stations, nearly 100 volunteers and 5 data mules.

**Generating Scenes and Pictures** We generate 25–50 scenes or events and associate each scene with a pool of pre-taken similar-looking real pictures, a total of nearly 1000 pictures. These pictures are actually of different landmarks/scenes in our campus taken at different angles and zoom levels. Different scenes have different observation popularity resulting in varying number of similar pictures per event (following a Zipf law distribution). In simulation, nodes visit event locations and take one of the of pictures at random from the pre-assigned pool. The picture is then tagged with the location of the node. Each node is equipped with a limited storage of 5-10 MB. This storage is obviously smaller than what a device could really have these days. As argued in PhotoNet chapter, this is to match the scaled-down size of our network, compared to the real size of tens of thousands of nodes (participants in a large city) and hundred of neighborhoods. Furthermore, we consider a network with a very poor delivery ratio (only 20–30% pictures are delivered) so that the diversity of picture collection really matters. We use a popular DTN routing protocol, Prophet [29], as the base packet forwarding protocol and override Prophet’s default dropping and transfer ordering as suggested in our scheme to implement PhotoNet on top of Prophet.

**Injecting Outliers** In our simulation, outliers are pictures of random scenes other than those mentioned above. They are geographically collocated with other pictures, but visually different from the rest of the pool. In each event pool, we artificially inject some non-relevant pictures. Figure 5.2 demonstrates the scene “shrubs” with an outlier. The total number of outliers is controlled by a parameter, called outlier ratio, which specifies what fraction of pictures could be outliers. Unless otherwise stated, we use
15–20% outliers.

Figure 5.2: Pictures of a scene “shrubs”; the rightmost one is an outlier

**Similarity Distance between Pictures** For computing distances between pictures based on visual similarity, we used existing CBIR (Content Based Image Retrieval) techniques. We used an open source lightweight library LIRe (http://www.semanticmetadata.net/) with four visual features, namely CEDD [39] (Color and Edge Directivity Descriptor), FCTH [40] (Fuzzy Color and Texture Histogram), Auto Color Correlogram [41], JCD [42] (Joint Composite Descriptor). In all cases, the feature is represented as global image descriptor vectors, which are mainly histograms of one or more particular interest in a very compact representation (CEDD and FCTH vectors are of 54 and 72 bytes per image respectively). Given two vectors, the distance is computed as Tanimoto coefficient [43] defined as \[ \frac{x^T y}{x^T x + y^T y - x^T y} \]. Figure 5.3 shows the probability density of similarity distance values computed between pairs of similar-looking and pairs of dissimilar pictures. We see that the distribution is multi-modal that enables us to separate similar pictures from dissimilar ones. We use JCD features for our experiments and choose the clustering threshold, \( \tau = 0.25 \). Due to the Java based implementation, we easily ported the library to Android phones.

**5.5.2 Simulation Results**

As a performance metric, we are interested in measuring what fraction of relevant scenes got delivered to the command station. We refer to this metric as *scene coverage* or simply coverage in this section. We treat a scene as delivered or *covered*, if at least one non-outlier picture of that scene is
reported to the command station. Delivery of outliers does not contribute to valid coverage. PhotoNet$^+$ aims at preventing outliers from being propagated through the network and attempts to drop them before they reach the command station.

The deployment setup for our considered disaster rescue mission is obviously outdoor. Therefore we can assume that devices (e.g., mobile phones) equipped with GPS receivers are used. Hence, pictures can be accurately labeled with the coordinates of the locations where they were taken/captured from. One concern may be the accuracy of these locations estimates. Note that we used location information only to arbitrate the case whether two pictures are coming from the nearby location (if so, pictures are visually compared; if not, pictures belong to different clusters and no visual matching is required). As per our distance function (Equation 5.1), the distance threshold, $T$, designates this difference. Due to this, our scheme is naturally robust to some degree of location inaccuracies. Roughly if the location estimates are not deviated more than $T/2$ from their true positions, the clustering of pictures is hardly affected. In case of indoor deployment, suit-
able techniques may exist, such as localization via Wi-Fi access points, for inferring such nearby locations. Having said that we do not evaluate our simulation experiments under location inaccuracies, rather defer it until we deploy a real system in future.

![Graph showing delivery ratio and coverage of Prophet and PhotoNet+](image1)

![Graph showing delivery of pictures across neighborhoods](image2)

**Figure 5.4:** (a) Delivery ratio and coverage of Prophet and PhotoNet+, (b) Delivery across neighborhoods.

To appreciate the significance of diversity in picture collection, we begin with showing results for content-agnostic forwarding scheme (Prophet) versus our diversity-aware forwarding scheme (PhotoNet+). Figure 5.4(a) shows the delivery ratio of pictures (fraction of total pictures delivered irrespective of scenes/outliers) as well as coverage of scenes in both protocols. We see that both schemes produce nearly the same delivery ratio (below 40%), but the coverage is very poor for Prophet. The reason is obvious. Prophet being unaware of similarity among pictures stores different pictures merely by chance, whereas diversity-aware PhotoNet+ suppresses similar stuffs and results in higher coverage even at very scarce resources. Figure 5.4(b) shows the delivery of pictures (each point is a picture) from different neighborhoods against time. We observe that PhotoNet+ does a better job of distributing pictures across neighborhoods than Prophet. The standard deviation of the number of pictures per neighborhood is around 30 and 65 for PhotoNet+ and Prophet respectively.

Figure 5.5 shows the coverage of PhotoNet+ and PhotoNet when transfer bandwidth and storage capacity at each node are varied. We use separate coverage for valid scenes and outliers. Outlier coverage means the fraction of delivered outliers out of what generated. We see that PhotoNet+ covers almost all scenes, while delivering only a smaller fraction (around 20%) of
outliers. On the contrary, PhotoNet results in almost the opposite: it delivers almost all outliers leaving legitimate scenes behind (valid coverage falls below 70%). This is because PhotoNet favors different pictures which weighs outliers highly. This deprives a whole bunch of legitimate valid scenes from being reported to the base. Figure 5.6 shows the delivery of outliers as outlier ratio increases. The delivery of a few outliers in PhotoNet+ is mainly due to the error in identifying outlier clusters and the inclusion of outliers in valid clusters.

![Graphs showing coverage and storage](image)

Figure 5.5: Scene coverage (a), (b) at varying storage size and transfer rate. (c) Retrieval of outliers at varying storage.

![Graph showing retrieval of outliers](image)

Figure 5.6: Retrieval of Outliers

It would be interesting to see how robust PhotoNet+ is in detecting outliers. We trace all transfers of pictures across the nodes in the simulation and find the fraction of transfers attributed to outliers. We term this as the ratio of outlier traffic. PhotoNet+ intends to suppress outliers while transferring
pictures across nodes. Figure 5.7(a) shows the ratio of outlier traffic for PhotoNet+ and PhotoNet. In PhotoNet, outlier traffic is very high, even dominates valid traffic at some point (beyond 20% outlier ratio). On the contrary, PhotoNet+ is robust to outliers and keeps outlier traffic low. This robustness, however, depends on the accuracy of detecting outliers. Recall that outliers are detected based on estimated sizes and z-scores of clusters, namely \( z(c) < \epsilon \) implies an outlier. We run experiments on several values of \( \epsilon \) and observe that outlier detection is sensitive to \( \epsilon \). We also show results for an “oracle” that allow nodes to know truly which objects are outliers. We see that \( \epsilon = 0 \) produces the lowest outlier traffic. Figure 5.7(b) shows outlier traffic at varying values of \( \epsilon \). It depicts that as \( \epsilon \) deviates around 0, outlier detection becomes weak and both outlier traffic and outlier coverage rise. This is because smaller \( \epsilon \) leads to false positives and larger \( \epsilon \) leads to false negatives. Again, when outliers are misclassified, they are given higher priority than others at the time of exchanging pictures (generally, because of their smaller cluster size). That’s why false positives lead to higher outlier coverage than false negatives. In all of our experiments, we use \( \epsilon = 0 \). For any specific application, the best way to determine \( \epsilon \) could be to make a few test runs with a smaller set of labeled picture set (where pictures are explicitly marked as outliers and non-outliers by human assessors) and then determine the value of \( \epsilon \) that results in the most accurate classification.

As we presented before, our diversity-maximization prioritization (CAP) protocols decouple of notion of similarity between objects and the notion of diversity, that is, diversity metric, of a content collection. First part is
an application-specific, but the second one is not. Given a proper distance between objects based on specific requirements of the application under consideration, the steps executed by our CAP protocols, particularly the decision regarding to which content to drop and which content to transfer are general. Below, in the context of PhotoNet+ application, we consider several possible distance functions based on availability of location information as well as based on various visual features.

We first show results the effect of location information in distance function. As per Equation 5.1, PhotoNet+ uses location as an important metadata in defining distance between pictures. The basic understanding is that if two pictures originate in two distant locations, they can’t be similar, irrespective of their visual similarity. When more than one picture originate in a nearby location, visual similarity among pictures is checked to determine whether the pictures belong to the same group (i.e., to the same scene) or not. Figure 5.8 presents the scene coverage (i.e., fraction of scene reported to the base) for three cases, i) both location and visual feature are used, ii) only location information is used, and iii) only visual feature is used. Since PhotoNet+ eventually delivers almost all scenes to the case if sufficient time given, we produce results for only within a limited time window. We present scene coverage as the fraction of scenes reported to the based within first 10 hours of operation. It appears that location plays a key role in accurately clustering pictures into different scene groups and the system produce highest result when location information is available. When location information is not available performance declines. We also observe that using only location (no visual feature) delivers a significant portion of scenes.

Then, we present the scene coverage results for distance function with the following seven visual features we considered:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>Color Histogram</td>
<td>[44]</td>
</tr>
<tr>
<td>AUTO</td>
<td>Color Auto Correlogram</td>
<td>[41]</td>
</tr>
<tr>
<td>GMM</td>
<td>Hierarchical Gaussian Mixture Model</td>
<td>[45]</td>
</tr>
<tr>
<td>CEDD</td>
<td>Color and Edge Directionality Description</td>
<td>[39, 46]</td>
</tr>
<tr>
<td>FCTH</td>
<td>Fuzzy Color and Texture Histogram</td>
<td>[40, 46]</td>
</tr>
<tr>
<td>JCD</td>
<td>Joint Composite Descriptor</td>
<td>[42, 46]</td>
</tr>
<tr>
<td>WGHT</td>
<td>weighted sum of JCD and AUTO</td>
<td></td>
</tr>
</tbody>
</table>

The effect of choosing the right visual feature used in defining distance
function is ultimately manifested in achieving good clustering accuracy. Clustering accuracy measures whether the distance function can indeed partition pictures into groups so that similar pictures coming from the same scene or event could belong to the same group. Clustering accuracy is defined as the fraction of pictures in a cluster that indeed belong to the same scene (compared to the ground truth). Figure 5.9(a) shows the clustering accuracy in partitioning a set of randomly chosen pictures from our dataset, when different visual features. We see that visual features, namely AUTO, JCD and GMM perform better than others. Color histogram quite expectedly produce the worst results. We show scene coverage results in Figure 5.9(b) for different visual schemes.

Figure 5.9: Clustering quality and scene coverage for visual features.
5.5.3 Phone-based Implementation

We implemented PhotoNet+ on Android phones (Google Nexus S) and evaluated results that are crucial for running the service on phones. These are mostly timing values for various computations invoked by PhotoNet+. We also compare the results with PhotoNet.

We generate timing results for several computational cases. The first timing results are due to extracting visual feature from each picture. Recall that for each generated picture, visual feature (usually a multi-dimensional vector) is extracted by the source only once. Then, the feature vector becomes a part of the picture itself and is passed along with the picture onto other nodes. We evaluate these timing results for LIRE implementation of these features (available at http://www.semanticmetadata.net/lire/) except for GMM. GMM is a computation heavy image processing task (usually available in Mathlab) that we were not able to port for the Android platform (rests are available in LIRE, which is written in Java, hence easily ported to Android). It appears that while CEDD, FCTH and COLOR are light weight processes, AUTO and JCD are costly. JCD is in fact a combination of CEDD and FCTH, and in our implementation they are computed in series so the extraction time simple adds up. Extraction times depends on dimension of pictures. We recorded the timings for two possible picture dimension.

The second timing result is the clustering time; the time consumed for inserting a new picture into the current collection. Recall that PhotoNet+ uses online clustering, in that the insertion is executed immediately upon the arrival of the picture. Clusters are reorganized once in a while that requires another set of computations. Moreover, recall that, in our scheme, when an object is added to an existing cluster, the distance is computed only to the centroid object of the cluster. We refer to this as “centroid scheme”. In ideal approach, distances to all objects in a cluster need to be measured (“all-pair scheme”). We produce the timing results for these two cases. In all cases, we produce the median values over 100 runs.

Figure 5.10(b) shows delays for clustering and re-clustering. We see that for inserting an item, centroid scheme results in smaller delay (less than 2ms per 100 pictures), whereas all-pair scheme produces larger delay (nearly 20ms). On the contrary, re-clustering time is longer for centroid scheme compared to the all-pair scheme. This is because centroid distance introduces
more distortions into clusters, which in turn requires more shuffles during re-clustering. In all-pair scheme, however, clusters are more accurate and they need less shuffle afterward, but the cost for each insertion is high to begin with. Since PhotoNet+ applies online clustering, it uses centroid scheme and defers the costly reshuffle operation to offline, when nodes are not in communication with others (DTN-style communication allows that).

Next, we observe delays for computing transfer order and dropping order of pictures. Unlike PhotoNet+, which effectively ranks clusters ($O(n + l^2)$ computations), PhotoNet does the same for each picture by measuring pairwise distances ($O(n^2)$), where $l$ and $n$ is the number of pictures and clusters respectively. It turns out that checking all pictures one by one is very expensive. Figure 5.10(c) shows delays of computing transfer order of pictures. We see that the time for computing transfer order in PhotoNet+ does not change much as the number of pictures grows, whereas for PhotoNet it grows constantly. This is because the number of clusters changes far slowly than
the number of objects. Figure 5.10(d) presents the same results for dropping pictures. PhotoNet$^+$ takes magnitude order of smaller time to determine the next picture to drop compared to PhotoNet, again due to clustering.

5.6 Conclusion and Future Work

We developed a scheme for delivering a representative subset of pictures from a larger pool in a participatory camera sensor network, where many pictures may be redundant and some may not be relevant. Heuristics were developed that balance outlier elimination and diversity maximization to achieve better coverage with the lowest number of pictures. The service was shown to offer a much higher coverage compared to previous work that focused on diversity maximization alone without outlier elimination. This is because outliers, by their very nature, are diverse, and hence (incorrectly) favored by diversity-maximizing algorithms. Future work on this topic will consider integration of our mechanisms with network caching in applications where content is requested by multiple sinks. Another direction is to extend the scheme with an estimation of source reliability, such that content prioritization is affected by reliability estimates. For example, pictures sent from unique locations by unreliable sources need not be considered. An experimental evaluation of the service deployed on Android phones is another direction.
In this chapter, we investigate the design space of caching policies that are needed when cached content items are not independent. Prior work on caching typically assumed that the value attained from caching any given content item is independent of the value attained from caching any other items. For cases where the above assumption is not true, algorithms were developed that break up large composite content objects (such as web pages) into smaller independent items that can be individually cached. Hence, a cache could simply store items of highest value and replace those of lowest value, where value can be computed independently for each item. Different replacement policies were developed that differ in how value is estimated (e.g., whether it is based on frequency of access or recency of access), as well as in how value is aged, to account for passage of time, and weighted to account for cost. The implicit optimization objective has always been to maximize a sum of values of individual items retained in the cache.

In many scenarios, however, content items are not independent. For example, two pictures of the same scene taken from slightly different vantage points carry significant mutual information. Receiving either image alone may have the same value, but receiving the second one does not double the value. In other words, value is subadditive. From the perspective of caching, a question becomes: how does information overlap between content objects affect cache replacement policies?

We start our investigation of the above question by postulating a higher-level network design objective. Namely, we consider that the main function of the network (including the cache) is one of maximizing information transfer per unit cost. In the context of caching, the main question becomes one of choosing the set of objects to cache such that the aforementioned information objective is maximized.

Information theory rigorously defines mutual information between two ran-
dom variables and hence allows us to quantify information in a content collection. Unfortunately, information-theoretic techniques are hard to apply to complex content types where not all information is equally important. For example, it is hard to express that two almost identical pictures of a bridge (one before an earthquake and one after) carry different information (namely, the more recent picture indicates that the bridge survived the earthquake). It is also hard to indicate that the most important piece of information in the pictures is the bridge and not, say, the identities of different cars on the bridge at different times.

Since the importance of information is application-specific, the content provider should have a general way of describing the relative importance of different features of content. Hence, it is useful to consider an abstract logical space, whose dimensions are (appropriately-weighted) content features or content metadata (such as time and location). Different content objects are described as points in that space, with distances between them denoting their degree of similarity; a smaller distance indicates a higher degree of similarity and vice versa. We call the aforementioned logical space, the similarity space. Further, each object has a scope of validity that extends for some distance around it in similarity space. We call this scope object coverage and logically represent it as a hyper-sphere with the object at the center.

Today’s caches can be thought of as diversity-maximizing devices in a one-dimensional similarity space, whose single dimension is time. When an object is cached, it has a time-to-live parameter. It means that the cached copy remains a valid representation of the original within some coverage interval, equal to its time-to-live. Our scheme explores extensions of this concept to a richer notion of coverage and similarity spaces.

Users send queries that may match one or more objects. Returning multiple objects in response to a query has subadditive utility, if their coverage overlaps in similarity space. Our design objective is to retain in the cache those objects that maximize the total utility (received by the users in response to their queries) per unit cost incurred by the network. In this work, versions of the above problem are investigated in increasing complexity and generality.

We expect that many content objects will remain independent. For these, no changes are needed in content representation. A default content similarity space (defined by a time-to-live) will apply. For other content types,
where advantageous, we add application-specific metadata to allow caches to compute richer similarity metrics. For example, an HTML object can carry metadata in its header in the form of a set of values that determine its attributes and a weight for each attribute. Using these attributes and weights, a cache can compute object location in similarity space, from which a distance (e.g., the Cartesian distance) can be computed to estimate information overlap between any two content objects. From there on, content-aware caching policies can be applied.

6.1 Information-maximizing Content Collection

According to the classical information theory, a content collection is meant to contain maximum about of information when it possesses the least amount of redundancy. We presume that redundancy originates due to information overlap between objects, which is arguably manifested by some degree of similarity between them. If two objects are more similar in some perceived sense, they share more information in common. Measuring the degree of similarity between objects thus becomes a vital issue in defining information maximizing content collection.

But the problem lies in the fact that similarity between objects is very application-specific. For example, in an application, say, in locating a missing person from a set of human face photos, two pictures can be deemed similar if they look alike (i.e., possibly contain the same person). Another application, such as situational awareness after a disaster whose goal is to locate and report about important damages/scenes, may consider two pictures/scenes different if they come from distant geographic locations, even thought they might be visually similar (e.g., a collapsed house). Hence, defining similarity depends on specific application context and need to account for various meta information and attributes of the objects under consideration.

An important aspect of our scheme is that we decouple the notion of application-specific similarity between objects from the generic information-maximizing content collection. The separation lies in the definition of a suitable distance function between a pair of content objects in a hypothetical content space. This distance is meant to denote the degree of similarity between objects. Given a certain distance function, our scheme is essentially
general in estimating the degree of redundancy in the collection. The definition of distance metric is indeed a primary way of customizing our scheme for different application scenarios. Only one characteristic we expect from the metric is that it should yield a lower value when objects are meant to be more similar.

Given a distance function, our next step is to quantify the amount of redundancy in a content collection. The least amount of redundancy is originated from a collection where no two objects have any content overlap (no similarity). This is also the one that results in the maximum amount of diversity for the same collection. Information-maximizing content collection is therefore needs to be the one that maximizes the diversity among objects.

6.1.1 Logical Representation of Content Objects

We assume that objects are represented as points in a multidimensional logical space where the proximity between two objects designates the similarity between them: the closer means more similar. We call this similarity space. We further assume that similarity extends upto a certain a distance around each object in the sense that any second object lying within that distance has some of degree of similarity with the first object. Let this constant be \( \tau \). Hence, it is useful to imagine that each object logically covers a hypersphere with radius \( \tau \) so that the spheres of two objects overlap when their distance is smaller than \( \tau \). Overlapping spheres indicates existence of shared information between the corresponding objects. The volume of this sphere is called object coverage. For an \( n \)-dimensional space—where \( n \) represents the number of attributes considered to measure similarity between objects—this volume is proportional to \( \tau^n \).

6.1.2 Measuring Diversity of a Content Collection

Due to overlap, the total coverage of a set of objects is generally less than the sum of the coverage of all individual objects. When no two objects overlap, coverage simply adds up producing the largest possible diversity for the collection, which is equal to the total number of objects in the set (for brevity, we express diversity in terms of the number of objects rather than
in exact physical volume unit). As objects do overlap, the coverage declines so does diversity (See Figure 5.1). At another extreme, when all objects are onto one another with an exact full overlap due to zero distance between them, the whole coverage degenerates to a single object’s coverage (which corresponds to the least diversity). Therefore, the total space coverage of all objects in a set can thus be treated as a quantitative estimation of diversity of the same.

The notion of coverage observes two simple properties. First, the coverage is monotone, in the sense that as objects are added to a collection, coverage can only increase (or stay the same, it can never decrease). Second, it has declining marginal gain, in that the expected additional coverage from adding a new object to a collection gradually declines (actually, cannot increase) because spheres become more and more overlapped.

Therefore, information maximizing content collection is a collection where the total coverage of objects is maximum. In our application context of caching, this content collection is actually referred to the resultset of a user query, and our objective is to generate the largest coverage per query. Whereas objects within a resultset of a particular query have the subadditive total coverage due to overlap, the same does not hold for objects returned in different queries. This is because all queries are treated independently, hence objects returned in separate queries do not overlap. Our information-maximizing content replacement for a cache tries to hold a set of objects in the cache that maximizes the sum of object coverage over all queries. This is what we describe next.

6.2 Basic Information-maximizing Content Replacement

Given a content similarity space, in this section, we consider replacement policies in the simple case where all objects have the same average cost of a cache miss. In this case, there is no need to consider the cost of cache misses (e.g., cost of fetching a replaced object from the source). Instead, optimization can focus on the coverage of objects that are present in the cache. The only cost considered is that of object storage in the cache itself.

Note that, this model is also suitable for cases where the exact cost of
a cache miss is not known. For example, in a mobile ad hoc network, if a requested object is not in the cache, it is hard to tell what it would cost to fetch it again from the source, since it might depend on the exact location of potentially mobile nodes at a future time, which is hard or impossible to predict. The model is also suitable in cases where the cache acts as a content store, and the network simply does not serve content not in the store.

We can formulate information-maximizing caching problem as of maximizing object coverage across all user query replies. We assume, in our caching context, that queries are not requests for certain data objects by specifying their unique identities, rather they are defined on metadata space. Since data objects are represented as points in a logical space, queries can also be specified by a bounded volume, say, by a sphere, asking for objects that are within that that sphere. So, each query is described by a point in the object space and a radius. Objects that lie inside the query volume are said to satisfy the query. Let \( Q \) be the set of all user queries, \( R(q) \) be the set of objects that satisfy query \( q \) (i.e., the resultset of \( q \)) and \( C(R(q)) \) be the coverage of objects contained in the resultset, \( R(q) \). While objects within \( R(q) \) can have subadditive coverage due to overlap, the coverage across queries add up straight, because queries are independent. So, for a given set of objects, \( X \), diversity caching asks to find an optimal subset \( S \subseteq X \), such that we have:

\[
\max \sum_{q \in Q} C(R(q)) \quad (6.1)
\]

subject to:

\[
R(q) \subseteq S, \forall q \in Q \quad (6.2)
\]

\[
|S| \leq L \quad (6.3)
\]

where \( L \) is the storage capacity of the cache in terms of the number of objects the cache can hold at most.

This problem is computationally hard because of the two holds. The first one is due to computing the volume coverage of a set of overlapping objects in exact geometric sense. The second one is due to combinatorial nature of choosing objects. In terms of complexity, the problem resembles to the classical 0-1 Knapsack problem. But, this instance is indeed harder. Unlike in traditional knapsack problem where the profit gain for including an object is independent and is given as a real number per object, here, in our
case, objects have relative gain in coverage (due to pair-wise overlap), which depends on what other objects are present in the set.

Two solution approaches are provided for this model. One offers great simplifications by assuming that objects can be clustered by similarity. The other considers exact overlap between individual objects and is more appropriate when no natural cluster boundaries can be defined.

6.2.1 An Approximate Clustering-based Solution

A concern with the problem formulation as described above is that the coverage of an object depends on the identities of all previous objects served and on the overlaps between objects in the similarity space. In order to arrive at scalable caching policies it is good to start with an approximation. Towards that end, we assume that objects in the similarity space are naturally partitioned into clusters. Objects that belong to different clusters are independent, whereas objects within a cluster have a large overlap (Figure 6.1(a)).

When objects are grouped in clusters, the computation of coverage becomes traceable. In a cluster all objects remain within a certain distance threshold. Let this threshold be the radius of coverage, $\tau$. That means, all objects in a cluster overlap each other, which in turn causes the total coverage volume of the cluster to remain bounded, no matter how many objects the cluster holds. This bounded total volume is $\pi(2\tau)^2$ for a 2-D space, whereas an individual object covers $\pi\tau^2$. It is also easily observable that as more objects are added to a cluster, the marginal increment to coverage gradually declines, because successive objects experience more overlaps. By the virtue of simplification, the total coverage volume can then be approximated by a geometric series of coverage volumes in terms of the number of objects, where the coverage increment by each additional object declines by a multiplicative factor from its earlier term. That means, the first object has coverage 1, the next one has $\lambda$, the next next one $\lambda^2$, and so on for a constant, $\lambda < 1$.

The following identity is observed for 2-D coverage of a cluster:

$$ (1 + \lambda + \lambda^2 + \cdots) \times \pi\tau^2 < \pi(2\tau)^2 $$

(6.4)
which leads to: \( \frac{1}{1-\lambda} < 4 \), that is: \( \lambda < \frac{3}{4} \). We choose, \( \lambda = \frac{1}{2} \). Based on this, we can formulate the cache replacement policy as follows.

We assume each query is satisfied by one or more clusters, that is, objects within a cluster are so close that they all satisfy the query. When returning objects from the same cluster in response to a query, the coverage due to \( k \)-th object is \( \lambda^{k-1} \). Let \( y_{c,q} \) be a binary variable indicating whether objects from cluster \( c \) satisfy query \( q \) or not, \( s_c \) be the number of objects in cluster \( c \) and \( C^* \) be the set of all clusters in the cache. Obviously, the coverage of cluster \( c \), is given by:

\[
C(c) = 1 + \lambda + \lambda^2 + \cdots + \lambda^{s_c-1} = \frac{1 - \lambda^{s_c}}{1 - \lambda}
\]

Diversity caching tries to maximize the following expression (i.e., the total of coverage returned over all queries, \( Q \)) for stored objects, \( X \):

\[
C(X, Q) = \sum_{q \in Q} \sum_{c \in C^*} y_{c,q} C(c) = \sum_{c \in C^*} \left( \sum_{q \in Q} y_{c,q} \right) C(c) = \sum_{c \in C^*} f_c \times C(c) = \sum_{c \in C^*} \sum_{k=1}^{s_c} \lambda^{k-1} f_c
\]  

where \( f_c = \sum_{c \in C^*} y_{c,q} \) is the frequency or popularity of cluster \( c \), that is, the number of queries that return objects from \( c \).

Based on the above expression, and also based on the assumption that objects inside a cluster are ordered, the first object from cluster \( c \) has a total coverage \( f_c \), second object has \( \lambda f_c \), third object \( \lambda^2 f_c \), and so on. Since cache intends to keep a high total coverage, that is, \( C(X, Q) \), when the storage becomes full, the object with the lowest of the above product is dropped first (assuming equal size). That means, the cluster with the smallest \( f_c \lambda^{s_c-1} \) value is chosen and the last object from that cluster is dropped. If no explicit ordering exists among objects in a cluster, any object within that cluster
can be dropped. In practice, such ordering is simple to establish, based on other attributes, for example, arrival time. In that earliest arrived object is dropped first (giving weights to freshness). If objects in a cluster have vastly different sizes, it is also good to normalize the above product by object size, evicting the object with the smallest normalized marginal coverage first.

In the above, we assumed that all objects within a cluster are returned against a query because of very small distances among objects inside a cluster. For a non-negligible inter-object distance, not all objects from a cluster would satisfy a query, rather a subset of them would do. The same can happen when some clusters are too big to fit in a query response or when queries explicitly set limits on the size of resultsets, allowing only a fraction of objects to be returned from all eligible clusters. In that, the total coverage per cluster would depend on the number of objects that satisfy the query (instead of the size of the cluster itself).

Since all objects from a cluster are not returned altogether against a query, some objects would have higher access, i.e., more popular than others in the same cluster. Recall that objects inside a cluster are ordered. Since the cache intends to retain highly popular objects, it would be logical to order objects based on their popularity. Once objects are ordered in the descending order of their popularity, each object is assigned with a rank. The first object with rank zero is the one with the highest popularity and other objects has successively lower popularity. Ties can be broken based on other attributes, such as object’s expiration time or arrival time. Let \( r_i \) be the rank of object \( i \) in its cluster. Obviously, \( 0 \leq r_i < s_{c_i} \), where \( c_i \) is the cluster to which \( i \) belongs. Let \( y_{i,q} \) denote whether or not object \( i \) satisfies query \( q \).

So, the total coverage expressed by Equation 6.5 can be written as:

\[
C(X, Q) = \sum_{q \in Q} \sum_{i \in X} y_{i,q} \lambda^{r_i}
\]

\[
= \sum_{i \in X} \left( \sum_{q \in Q} y_{i,q} \right) \lambda^{r_i}
\]

\[
= \sum_{i \in X} f_i \lambda^{r_i}
\]

(6.7)

where \( f_i = \sum_{q \in Q} y_{i,q} \) is the access frequency (popularity) of object \( i \). Hence,
the object with the smallest $f_i \lambda^{r_i}$ value is evicted when the storage becomes full.

![Fig 6.1](image)

(a) Clusters  
(b) No obvious clusters

Figure 6.1: Each object is represented as a sphere. An overlap indicates that the corresponding objects have similarity.

### 6.2.2 Formulation without Clusters

In some cases, data objects may not follow exact clustering patterns; instead, an arbitrary order of similarity across objects may exist (Figure 6.1(b)). In that case, approximating coverage by a geometric series may not be accurate. Instead, we need to quantify the coverage in terms of the number of objects in the set and the degree of overlaps that exists in them. Whereas computing the exact geometric coverage of overlapping spheres in an arbitrary space dimension (even for 2-D circles) is hard, we show that there exists a simpler way of approximating the coverage volume (possibly for 2-D space) in terms of the number of objects and the number of overlaps among objects. Let us take a small detour to establish this in the following.

#### Coverage Approximation from Overlaps

Conceptually, the coverage obtained from a set of objects is equivalent to the number of non-overlapping objects that cover almost the same amount of space as covered by all objects in the set. Obviously, this number declines as overlaps among objects grow. Let an object set with $n$ objects contain $o$ overlaps. It would be interesting to find relationship among $c$, $n$ and $o$,
where the coverage, \( c \), is the number of non-overlapping spheres whose total coverage volume is equal to the volume covered by \( n \) objects having a total of \( o \) pair-wise overlaps.

We observe that if there is no overlaps, i.e., \( o = 0 \), \( c \) becomes \( n \). Again, when all objects overlap each other resulting in a total of \( \frac{1}{2}n(n-1) \) overlaps, \( c \) becomes 1. With an arbitrary number of overlaps, coverage remain within 1 and \( n \). For the ease exposition, let us assume \( n \) objects are partitioned into \( c \) non-overlapping equal-sized groups so that objects inside a group all overlap each other, but groups themselves do not overlap. Each group contains an equal number of \( \frac{n}{c} \) overlapping objects. Each group can contain, at most, \( \frac{n}{2c} \left( \frac{n}{c} - 1 \right) \) overlaps. So, the total number of overlaps can be, at most:

\[
c \times \frac{n}{2c} \left( \frac{n}{c} - 1 \right) = \frac{n}{2} \left( \frac{n}{c} - 1 \right)
\]

Since we already know the number of overlaps, which is \( o \), we have:

\[
o = \frac{n}{2} \left( \frac{n}{c} - 1 \right)
\]

which gives:

\[
c = \frac{n^2}{n + 2o}
\]

The above equation gives us our intended relationship. In reality, however, each group is supposed to contribute more than a single sphere. So, the actual coverage would be slightly greater than that this. In order to validate the relation empirically, we conducted a Monte Carlo experiment in a 2-D
space. We computed the exact geometric coverage produced by objects and compare the results with our estimation. The process splits the whole space into a large number of small cells and then counts the cells covered by a certain number of overlapping circles, thus approximating the total geometric area covered by them. The expected level of overlapping among circles is controlled by the size of the bounded box that contains all circles. Figure 6.2 shows the coverage results for a number of overlaps at varying number of objects. The solid line under each series of curves is due to Equation 6.9 for the associated $s$ and $o$. We observe that our estimation by Equation 6.9 closely matches to the actual (geometric) coverage (with a mean absolute error 7.87699). We also compute the Pearson’s correlation coefficient as 0.99474 (Figure 6.2(b)), which indicates that there exists a strong linear correlation between them.

Content Replacement without Clusters

Now we devise methods to apply this coverage estimation in content replacement. Let query $q$ return $n_q$ objects (for brevity, in successive discussion, we use the same symbol $q$ to denote both the query and the resultset of the query). Let $o_{ij}$ indicate whether or not two object $i$ and $j$ overlap and $o_q = \sum_{i,j \in q} o_{ij}$ be the number of total overlaps in the resultset of $q$ (assuming $o_{ii} = 0$). Note that, $o_q$ actually double counts the number of overlaps that we need to take care of (omit 2 from the expression in Equation 6.9). So, the coverage obtained by query $q$ is given by:

$$c_q = \frac{n_q^2}{n_q + o_q}$$

(6.10)

Therefore, the total coverage obtained for all queries can be obtained as follows:

$$C(X, Q) = \sum_{q \in Q} c_q = \sum_{q \in Q} \frac{n_q^2}{n_q + o_q}$$

(6.11)

We can further simplify the above expression by observing:
\[c_q = \frac{n_q^2}{n_q + o_q}\]
\[= n_q - \frac{n_q}{n_q + o_q} \times o_q\]
\[= n_q - \mu_q \times o_q\]

where \(\mu_q = \frac{n_q}{n_q + o_q}\) can be treated as discount to coverage due to overlapping. Therefore, the total coverage over all queries is given by:

\[C(X, Q) = \sum_{q \in Q} (n_q - \mu_q o_q)\]
\[= \sum_{q \in Q} \left(n_q - \mu_q \sum_{i, j \in X} o_{ij}\right)\]

Revisiting the notation \(y_{i,q}\) to denote object \(i\) satisfying query \(q\), we have \(n_q = \sum_{i \in X} y_{i,q}\). Then, Equation 6.12 can be written as:

\[C(X, Q) = \sum_{q \in Q} \left(\sum_{i \in X} y_{i,q} - \mu_q \sum_{i, j \in X} y_{i,q} y_{j,q} o_{ij}\right)\] (6.12)
\[= \sum_{i \in X} \left(\sum_{q \in Q} y_{i,q}\right) - \sum_{q \in Q} \mu_q \sum_{i, j \in X} y_{i,q} y_{j,q} o_{ij}\]
\[= \sum_{i \in X} f_i - \sum_{i \in X} \sum_{j \in X} f_{ij} o_{ij}\]
\[= \sum_{i \in X} \left(f_i - \sum_{j \in X} f_{ij} o_{ij}\right)\] (6.13)

where \(f_i = \sum_{q \in Q} y_{i,q}\) is the popularity of object \(i\) and \(f_{ij} = \sum_{q \in Q} \mu_q y_{i,q} y_{j,q}\) is the joint popularity of object \(i\) and \(j\), that is, the number of the times both \(i\) and \(j\) were returned against a query normalized by \(\mu_q\).

As per Equation 6.13, we can define coverage per object, \(c_Q(i)\), over all queries as follows:
\[ c_Q(i) = f_i - \sum_{j \in X} f_{ij} o_{ij} \]  

(6.14)

The object with the least coverage would be dropped from the cache. The values of \( f_i \) and \( f_{ij} \) for all objects can be computed incrementally for a query set \( Q_k = \{q_1, q_2, \cdots, q_k\} \) as follows:

\[
f_i(Q_k) = f_i(Q_{k-1}) + 1, \text{ if } i \text{ satisfies query } q_k \]  

(6.15)

\[
f_{ij}(Q_k) = f_{ij}(Q_{k-1}) + \mu_{q_k}, \text{ if } i,j \text{ satisfy query } q_k \]  

(6.16)

### 6.3 Protocol Design and Implementation

We design and implement our proposed diversity caching service in a system of networked caches. The service consists of a set of (mobile) nodes and a set of caches deployed at different network locations. Nodes, also called sources, generate data objects and forward their queries to the caches. Caches hold a small subset of generated objects and respond to queries. To give context, we use the following use case, called situational awareness, which is geared in the context of a disaster response scenario. Suppose a set of human neighborhoods have been stroke by a large disaster. Volunteers and rescue workers are deployed in the distressed neighborhoods in order to document damages and send those reports to a command base for an immediate attention. Volunteers equipped with camera-phones visit places and take pictures of damage scenes and send them to the local command station. Each neighborhood has a local command station where local events are reported (by pictures of events). We assume that all pictures are metadata rich; for example, geo-referenced, thanks to GPS sensor available in almost all smart phones, which enables users to make queries about events/scenes originated from a certain location. Users residing in one neighborhood could be interested in events happened at another neighborhoods. In this context, the local command bases can function as caches, and users can direct their queries to those caches. It is now becomes vital for those caches to decide which data objects (i.e., pictures) they need to hold, if the caches had a limited storage. Due to the catastrophe, we assume that regular communication styles, such as the
Internet, cell towers, are damaged or unavailable due to power outage, leaving device-to-device DTN (Disruption-tolerant network)-like communication to be the only viable option.

The caching system has the following aspects.

### 6.3.1 Data and Query Model

Objects are generated by nodes and are tagged with meta information that allows to compute similarity distance between them. Queries are also metadata based range queries containing certain metadata range (e.g., pictures from certain location) to request specific pieces of data. Rather than requesting a specific known item by object identifier, the service is intended for requesting a known event or a generic query requesting information across a general area, both of which may be satisfied by multiple items. Because multiple results may be returned, typical assumptions with caching cannot be made. We cannot assume we know the source address of a data object because multiple nodes can stamp the same meta information to a packet at different times. For this reason we choose to have sources push their content to their local caches. We describe more on this later.

### 6.3.2 Computing Distance Function

Computing distance between objects require metadata information about objects. One possible way is to use the Named-data Networking [47, 24] paradigm. In that, objects should have URL-like names that encode the metadata needed for computing the logical distance between them. A distance function can therefore take two object names and tell whether they overlap in our similarity space.

### 6.3.3 Communication Model

Data sources pro-actively push their generated data objects onto their local caches. Queries are also sent to the local caches. We assume nodes residing in a neighborhood know the address of their local cache so that they can direct their data and query traffic to the local cache. Certain discovery
protocol can exist that allows nodes to know the local cache. Once a query is received at the cache, the cache returns with the objects (if any) that satisfy the query. Queries generated from the local sources are also multicast to all other caches in other neighborhood to receive more recent objects that could satisfy the query. We show a schematic diagram of the network in Figure 6.3.

![Networked cache scenario](image)

Figure 6.3: Networked cache scenario.

### 6.3.4 Responding to Queries

Each cache stores all queries it receives from local sources as well as from remote caches (via multicast). These are called local and remote queries respectively. Due to scalability issue, queries up to a certain past time, say last couple of hours, can also be stored, instead of storing all. The following issues are considered when a cache replies against queries:

- If the local cache is able to respond immediately to a query, it does so, but still multicasts the query to other remote caches. This is to encourage moving new data through the network.

- If the local cache is able to respond immediately to a query, the query is considered satisfied. However, upon a miss, the query will be stored until new data arrives at the cache, at which point that data will be forwarded to the requester and then the query will be marked satisfied.

- Before the local cache pushes the query to other caches, it notes in the
query the packets it was able to respond with. Caches receiving this query will not respond with any packet on this lists.

- When a remote cache receives a multicast query from another cache, it responds immediately with any data it has available. If it is unable to respond, it will hold onto the query until locally generated traffic arrives that satisfies the query. Traffic from other caches (i.e., remote objects) will not satisfy a remote query because we presume the other cache would have responded to the query already.

So queries are responded with data objects in two different ways. The first one is immediate response: a query arrives at a cache node and the cache replies with the objects that satisfy the query (from both local and remote objects). Since caches hold queries for some time, objects arriving at a later time can be replied against those queries. These replies are called delayed response. Since the underlying network is DTN, objects and queries experience arbitrary delay in their propagation so delayed responses deem essential.

6.3.5 Computing Popularity of Objects

As objects are returned from the cache against queries their access frequency, i.e., popularity, changes. For an immediate response, the popularity values of the satisfying objects are updated based on the caching scheme used. In case of delayed response, the new object’s popularity value is computed based on the record of earlier query history. In an usual caching scheme, all objects stored in a cache always arrive at the cache as a part of a response to a user query. Over time, some objects build better access records than others by satisfying more queries and therefore remain in the cache while others get dropped. In our case, however, object’s arrival at the caches is quite unsolicited (due to pushing of objects by sources). That means when a new object arrives, it may not have any prior access records, although a similar object might have been requested earlier. We identify earlier queries from query history to determine objects that could have been satisfied by this new data object and compute popularity accordingly.

Popularity is aged over time. If an object is not queried for long, it’s popularity is decremented (multiplied by a constant less than 1). In the
current implementation, popularity is only factored using locally sourced queries. When a cache receives a multicast query from another cache, the query is not used to factor popularity. The purpose of this is to optimize cache contents for the local area. An optimization we hope to add in the future is to factor in these multicast queries in the popularity algorithms by giving the highest weight to queries sourced locally and gradually decreasing weight for multicast queries based on how far away in the network the query was originally sourced.

6.3.6 Purging Objects from Caches

Each cache runs a replacement policy whenever it receives a new object and its storage becomes full. The replacement policy essentially finds a ranked order of objects currently stored at the cache and identifies the object with the least utility to be evicted immediately. When a burst of objects arrive, these computations need to be computed back to back for each incoming object. In order to reduce running these expensive ordering computations, we considered a high and low watermark scheme for purging items from the cache. Once the cache reaches a size limit specified by the high watermark, it runs the replacement algorithm for items in its cache, and then purges items until the size specified by the low watermark is reached. New items are not introduced during a purge cycle.

6.4 Evaluation

Our experiments are intended to evaluate the performance of various caching policies by Simulation for a collection of non-independent data objects. We presume an application context where data objects in a network, such as pictures, may contain redundant information, so receiving multiple of these objects does not provide additional utility to a user. Diversity caching techniques attempt to maximize utility of objects in the cache by first ejecting objects deemed redundant. In the following, we describe our simulation environment, competing protocols and performance results.
6.4.1 Simulation Environment

We evaluate our caching protocols for hypothetical situational awareness application we described in Section 6.3. We simulated the scenario with 30 nodes in an area is of $4 \times 4$ km$^2$ city map split into 5 geographically isolated regions (neighborhoods). Humans (i.e., nodes) at each neighborhood visit places randomly at a speed 2-5km/hour, while a few vehicles (DTN mules) run between local centers to the main center at a speed 10-20km/hour. On each trip, vehicles choose their next neighborhood randomly. If not otherwise stated, there are six caches, one in each of the neighborhood (usually at some static location) plus a special node that moves between locations, acting as a DTN mule. All non-cache nodes generate traffic at various times and immediately push traffic to their local caches. A few nodes make queries directed to their local caches, which are later forwarded to other locations via a multicast. Pictures are tagged with x/y coordinates and each query is also expressed by a x/y location and a radius, requesting of picture taken within that specified region. Application data payload per object is 1KB.

We use x/y coordinates as a simplification for providing meta tags to object contents, in that, we envision that, two objects coming from the same geographic location are mostly similar. And, similarity decreases as the distance between them grows. In a real world situation, however, other set of meta information must to considered to infer such similarity, such as time, for pictures, color histogram and advanced visual features can be useful. This, however, simply resorts to defining an appropriate distance function with more attributes. There are indeed documented techniques to define similarity between picture objects [39, 40, 42]. As we claimed earlier, our caching scheme separates similarity measures from caching decisions in the sense that the system works generally for any appropriately defined distance function. So, the implementation with more features is a straightforward extension to the current system.

6.4.2 NS3 Wireless Emulation

We used an NS3 augmented DTN environment for our experiments. Our caching service sits on top of a DTN node which is “connected” to other DTN nodes via NS3. We primarily use NS3 to control the physical layer,
precisely because NS3 provides a high fidelity environment where network overhead, delays, and loss are realistic. We create one virtual machine (VM) for each node in our experiments. NS3 promiscuously listens to the output traffic from each virtual machine and acts as a router and pushes traffic to one-hop neighbors, which are determined based on mobility trace, distance between nodes, and pathloss algorithms in use. We produce mobility trace of mobile nodes by ONE Simulator [13], which allows node to moves in a city map with streets. We build our scenario with neighborhoods, local centers, people and vehicles moving in a map. Figure 6.4 provides a visualization of how VMs send traffic through NS3. We used the scalable Spindle3 DTN [48] that provides all capabilities specific in the DTN specification (RFC 5050). In addition, S3 DTN implemented the DTN application interface and has routing components for Hazy Sighted Link State (HSLS) [49] routing and Epidemic routing. We simulated each experiment for 10,000 real seconds, which is nearly 3 hour of operation.

6.4.3 Performance Evaluation

We evaluate our caching scheme for a couple of performance metrics. The first one is *query hit ratio*—the fraction of queries that have been immediately responded from the local cache (by immediate responses). Note that, the term “hit” is not used as an exact matching to an item as used in traditional caching literature. We still happen to use this term, as defined above, to be consistent with the popular term used to show caching performance. The second metric is *coverage per query*—the coverage per query. Coverage
for each query is at least 1 if the query is serviced by at least one satisfying object, otherwise it’s zero. As we have argued before, coverage is the quantitative measure of diversity of the objects that are responded from the caches against a certain query. Coverage is regarded as the perceived utilities for user queries. If users happen to receive diverse results (omitting similar objects, for example), it results in higher coverage and consequently achieves higher utility. In fact, in the following, we use the term utility and coverage interchangeably. The third performance metric is coverage per object replied that takes into account the number of objects replied from the caches to achieve a certain coverage. The last performance metric is network load— the ratio of the number of total objects returned from the caches to the total number of packets sent.

We compare the results of diversity-aware replacement policy, diversity caching (DC), with two other cache replacement policies. These are:

- LRU (Least Recently Used)—the object that has been least recently access is dropped when storage needs to make room for new objects.

- IC (intentional caching) [50]. IC computes popularity of a data object in terms of probability that the object would be accessed at least once from now on to its lifetime, formally given by \( p = 1 - \exp\left(\frac{-k(T-t)}{t_k-t_t}\right) \), where \( k \) is the number of times the object is queried so far, \( t_k \) denotes the time of last access, \( T \) is the expiration time, and \( t \) is the current time. IC schemes drops the object with the least \( p \) value. For longer packet expiration time \( (T) \), IC, however, becomes close to LFU (Least Frequently Used first) replacement scheme.

Figure 6.5 shows the locations of generated objects in different neighborhoods and the heatmap of queries. Heatmap, in color contrast (darker means higher), shows objects located in which geographic areas are requested more frequently than others. We assume that queries would follow a skewed distribution in that objects from a set of locations would be requested more than from the other locations. There could be also locations from where objects are never queried. Both data and query generation are Poisson processes with a certain rate. Ideally, no null-query is generated, that is, all queries are serviceable in the sense that there is at least one object generated that could satisfy the query. But in the actual operation arbitrary ordering of
packet’s arrival and the dropping of objects from caches do not necessarily lead to 100% service. Some queries may end up not having served at all.

Figure 6.5: Scatterplot of data objects and heatmap of queries (darker shade means more frequently queried).

Figure 6.6: Recall and precision at varying cache sizes.

We generate experimental results for a multi-site networked caching system. Our goal is to show that diversity caching serves queries better compared to other replacement schemes in terms of coverage (i.e., diversity). Before we use our coverage metric as a measure of diversity, let us first see results where coverage is rather understood in an usual sense, namely as recall. Recall is popularly used in information retrieval literature, which usually refers to the fraction of “relevant” information retrieved for queries out of total available objects. Another associated metric is precision: the proportion of relevant objects replied against a query. In our context, for instance, if a query is responded with 10 objects from 3 object groups (each
group contains similar objects, hence not all are assumed to be “relevant” except one), and if there were originally 4 groups that could have satisfied the query, then recall is 3/4 and precision is 3/10 for this particular query. Figure 6.6 shows the recall results for varying cache sizes. We observe that diversity caching results in better recall and precision compared to LRU and IC.

Figure 6.7: Query hit ratio, coverage per query, coverage per object replied for varying sizes of caches.

Figure 6.7 shows query hit ratio, coverage per query and total data packets responded against all queries at varying cache sizes. We present cache size as the fraction of total objects created in the network. We see that as the cache size increases, all metrics improve in all protocols, which is quite expected. In all cases, diversity caching has superior performance over LRU and IC. Figure 6.7(c) shows the number of data packets received against queries by the requester node. Caches running diversity-aware content collection keep those objects that lead to a wider coverage. Since the total storage is limited, this means, caches store less number of non-independent objects. In that, the
number of items returned against a query is lower and the query response contains a few non-overlapping objects, while other schemes respond with much redundant stuffs. Thus the total traffic in diversity caching is quite lower than other schemes. We also compute coverage per object replied as shown in Figure 6.7(d). We see that diversity caching has better coverage per object compared to LRU. This is because diversity caching replied fewer objects to achieve higher coverage than LRU. It is also observed that IC has slightly better coverage per object than DC. This is due to the fact that coverage and the number of objects replied both are small for IC, but the ratio, however, appears to be slightly higher than that of DC’s.

It is to understand that coverage for a query may depend on the number of objects returned for that query. If the number of returned objects is lower, coverage might also be low. Coverage, however, is not absolutely given by the number of objects alone, rather by the number of groups and the distribution of objects among those groups. To be more precise, it depends of the evenness of objects among different groups. A query served with objects evenly from different groups has higher coverage than that of another query which is served with the same number of objects but all from the same group. In this perspective, coverage is the most useful performance metric for evaluating caching performance with non-independent objects.

In Figure 6.7(b), we show an optimal result presenting the best possible coverage obtainable by the diversity caching replacement scheme. The optimal results are obtained in a setting when the cache replacement runs as an oracle that allows all the caches to know all future queries and that asks caches to hold objects maximizing the coverage over all queries in view of those future queries. The oracle solves the following optimization (as formulated by Equation 6.6) and finds the optimal number of objects to hold from each cluster:

\[
\begin{align*}
\max \quad & \sum_{c \in \text{clusters}} f_c \times \frac{1 - \lambda^{s_c}}{1 - \lambda} \\
\text{s.t.} \quad & \sum_{c \in \text{clusters}} s_c \leq L, \quad L = \text{cache size}
\end{align*}
\]

Effectively, the oracle may not drop an object if there is a chance that the object may be queued later. This optimal result is, however, network delay
invariant in the sense that it does not consider propagation delays of moving
data objects and queries inside the network, rather consider them instantane-
ously available or arrived at the cache as soon as they are generated.

Figure 6.8: Network load (the number of packets returned per query)

Next, we evaluate the impact of query holding time on query responses.
Recall that each cache, upon receiving a query, holds the query for a certain
time, called query hold time. New objects arriving within that time are
returned as delayed responses. We observe that as query hold time increases,
both coverage and total responses against queries increase.

Diversity caching clusters objects based on the distance between them.
We assume that similar objects are generated at the same location (with a
small jitter in location x/y coordinates). Sometimes, location sensor can be
faulty or are not properly calibrated that may introduce larger jitter than
expected. Obviously, our clustering algorithm might fail when such noises exist. We emulate noises by artificially adding some degree of deviation to the original reference coordinates. Experiments are conducted to observe the responsiveness of our caching scheme against these noises, while the underlying clustering mechanism and the associated parameters remain the same. Diversity caching, when clustering objects, assumes highest expected deviation to be 20 meters, whereas the experiments allow noises up to 50 meters. Figure 6.10(a) shows that there is indeed a decline in performance when noises become large.

![Figure 6.10(a) Effect of Jitter](image1.png) ![Figure 6.10(b) Impact of DTN Mules](image2.png)

**Figure 6.10:** Effect of metadata noise and multiple mules.

Next, we observed performance for varying degree of connectivity via data mules. In earlier experiments, a single mule is connecting caches at distant locations. Figure 6.10(b) shows the delay results for varying number of mules. As the number of mules increases, connectivity scenario improves so delays in query responses decline.

**Impact of Various Parameters of Diversity Caching**

Next, we analyze the variation of various parameters used in diversity caching, namely the variation of data and query generation pattern and the impact of utility factor, $\lambda$. Recall that $\lambda$ is the factor by which the utility gain from a similar object declines when similar objects are returned against a query. We experiment with different values of $\lambda$’s.

First, we consider one important aspect of user data generation. This is *convergence to events*. We described earlier that in our experiment, users
roam around in a distressed neighborhood and occasionally attend various events as the events are triggered. While attending an event occurred at a certain location, the attending user generates data objects (for example, take a photo of certain damage). All these content have some degree of similarity among themselves (since ideally they all correspond to the same event). In usual experiments, we generate traffic so that objects from nearby geographic locations (i.e., a set of similar objects) are pushed to the cache at some regular interval. This is based on the observation that the same event could be reported by multiple visitors but in different hours of the day, so generated objects spread over time. This corresponds to zero or small burstiness. There can be, however, occasions when a certain event might catch drastic user attention, which may cause many users to rush to the event location immediately and push their content to caches. In that, all generated similar objects are pushed at around the same time. This corresponds to high data burstiness.

Figure 6.11: Effect of degree of burstiness of data traffic.
Figure 6.11 shows the results for different degree of data burstiness. We observe that both performance metrics, namely query hit ratio and coverage are affected due to burstiness. The decline in cache hit is expected. This is because when similar objects are pushed at a very close time window, depending on the current content state caches may drop them all, which ultimately causes further queries asking for these objects to miss.

Second, we analyze performance results at varying degree of data load and query patterns. Queries are power law distributed. In that, objects originated from a few locations queried by a large number of users compared to other locations. In generate these queries in the following way. For a set of indexed locations—around which queries are actually made by specifying a radius along with—objects located in $i$-th location are queried with probability proportional to $\frac{1}{i^\alpha}$, where $\alpha$ determines the skewness of the distribution. So, $i$-th location is queried with probability $\frac{1}{i^\alpha}/\sum \frac{1}{i^\alpha}$. Different values of $\alpha$ specifies different query pattern. For example, $\alpha = 0$, means all locations are equally likely (queries are uniform). For higher $\alpha$, queries shift to more and more to a set of few popular events. Figure 6.12 shows the query heatmap for various values of $\alpha$.

Performance results for different $\alpha$ are presented in Figure 6.13. It reveals that diversity caching results in consistent results at varying degree of skewness (utility per replied object remains fairly constant), where LRU’s performance declines at higher skewness index. This is because LRU tends to hold objects only from popular queries which made a significant portion of queries to remain as unsatisfied. We also present results for values of query radius in Figure 6.14(a—c).

Third, we observe results at various values of $\lambda$, a system parameter that defines the decreasing weight toward utility for receiving a successive similar object against a query response. Usually, the effect is system-wide, i.e., the factor corresponds to all user queries in the same way. This is also possible that individual user can set its own $\lambda$ independently. Given that a query can be satisfied by objects from multiple groups, the utility of the query is computed as follows: first object from each group contributes 1, next object from the same group gives $\lambda$, next one $\lambda^2$ and so on. Therefore, utility per query is numerically slightly greater than the number of different groups the query covers. We show results in Figure 6.15. There are two extremes: $\lambda = 0$ means only the first object from each group matters; rests are devoid of any
value. On the other hand, $\lambda = 1$ means all objects are equally valuable. We observe that setting $\lambda = 0$ generates the lowest utility per query whereas for $\lambda = 1$, we have the largest utility. In terms of fraction of queries served, we observe the reverse, namely $\lambda = 0$ results in highest query hit ratio. When users set their own $\lambda$ ($\lambda= \text{per user}$), the system eventually experiences the least utility per object replied.

Finally, we generate the same set of results in comparison with LRU replacement scheme to show competitive advantages of diversity caching over LRU (Figure 6.16. It shows that diversity caching outperforms LRU in all metrics.

Figure 6.12: Query heatmap at various degree of query pattern generation.
Figure 6.13: Performance at various traffic load and query distribution.

Figure 6.14: Performance results at various value of query radius.
Figure 6.15: Performance results at various $\lambda$’s.
Figure 6.16: Performance results at various $\lambda$'s compared to LRU.
In this chapter, we describe in detail the simulation environment we created for evaluating our protocols described in this thesis. The main component of the simulation environment is the mobility model of nodes during a disaster aftermath. As a result of a disaster one sees population shifts into shelters, and movement that is dominated by the response force (e.g., rescue, utility, fire, police, national guard). We observed that DTN literature lacks a mobility model attuned to distinct characteristics of disaster scenarios. This is particularly problematic as the service characteristics of a DTN depend greatly on the physical mobility of devices whose movement ultimately connects communication islands. The most commonly used models in DTN simulations are Random Waypoint or the more accurate Map-Based Movement Model. But these models do not capture the most salient features of movement at a disaster—population movements are clustered, directed, and transient; during recovery a pattern of movement of among emergency responders dominates. These types of movement are neither completely random nor completely periodic—the defining characteristics of the existing random movement models.

To this end, we develop the Post-Disaster Mobility Model (PDM) to model the movement of people and rescue activities in the affected area. The model has been implemented as an extension to the widely used DTN simulator ONE (Opportunistic Network Environment) [13].

---

1PDM has been published as: Md Yusuf S Uddin, David Nicol, Tarek Abdelzaher, Robin Kravets. “A Post-disaster Mobility Model for Delay-tolerant Networking,” Winter Simulation (WinterSim), Austin, Texas, December 2009 (invited)
7.1 Post-Disaster Mobility Model (PDM)

We develop a novel mobility model, the Post-Disaster Mobility (PDM) model, that attempts to mimic the situation after a natural disaster. PDM describes different role-based movements, based on a given city map. It models two main groups after a disaster: survivors, and rescue workers that aid survivors. PDM describes movement models for both groups. Clearly a post-disaster scenario is entirely dependent on the type of disaster. Hurricanes, tornadoes, earthquakes, and fire all induce different movements. One distinguishing characteristic is whether a population is forewarned, such as in a hurricane or tornado, as the population movement may significantly precede the event; with a fire or earthquake the population shifts occur after the event, and may interfere with the emergency response. Recognizing the difference and the fact that we can model the population shift one way or the other, we concentrate on scenarios where a fore-warned population moves in advance of the disaster to evacuation centers. They stay at these centers for a substantial amount of time and then return to their homes.

Following the disaster a relief operation is launched to help survivors (e.g., supply food and water). Our model assumes there are a small number of main coordination centers, and a larger number of evaluation centers. Vehicles move between the main coordination centers and the evacuation centers to supply relief goods. These are some of the vehicles assumed to have DTN devices that buffer and carry communication between communication-isolated coordination and evaluation centers. In addition, a number of rescue workers and volunteers are deployed at each evacuation center to dispense relief goods and services to the affected people (these also serve as DTN “carriers”); other DTN carriers include police officers, who patrol a larger area, e.g. to prevent looting.

Although we try to model mobility in a post-disaster situation, it is hard to directly validate the model against a real disaster instance. No documents/reports happen to describe the disaster operation in as detail as required to reproduce the scene by a simulator. We consulted few formal documents prepared by FEMA (http://www.fema.org) in our construct, at least to the level of identifying various participants and their functionality in the disaster response. National Response Framework [51] provides guidelines how to mobilize resources and personnel to carry out certain response
functions. NRF applies functional approach that groups capacities into 15 Emergency Support Functions (ESFs) to provide resource, personnel, program implementation and emergency services that are likely to be needed during the incident [52]. A few ESFs that are being simulated to some extend in our model are transportation (ESF #1), fire-fighting (ESF #4), public health and medical services (ESF #8), urban search and rescue (ESF #9), and public safety and security (ESF #13). Our selection of centers and agents is in compliance with this framework.

Incident Command System [53] is a standardized on-scene incident management concept. It describes how command hierarchy is maintained among various working teams and units. It proposes establishment of ICS facilities, namely incident command post, base or staging area to coordinate rescue operation in the incident area. Base is the location from which the primary logistics functions are coordinated, whereas the staging area hold personnel and equipments waiting for tactical assignments. ICS suggests establishing one or two Emergency Operations Centers (EOCs) which are responsible for community-wide resource management. National Planning Scenarios (Scenario 10: Natural Disaster – Major Hurricane) [54] that describes the components of a response operation after a Category 5 hurricane. It entails the mission objectives such as emergency management/response, hazard mitigation, evacuation/shelters, victim care, recovery/remediation, and so on. In our model, we try to mimic a couple of these response services with the associated agents and their mobility. Note that, instead of realizing the exact rescue operation in much detail—which is admittedly beyond the scope of our work—what we wanted to do is to identify key mobile agents and to extract their mobility patterns as suggested in documents in order to build our simulator accordingly.

In our mobility model, we address a number of characteristics, described below.

7.1.1 Disaster Area and Neighborhoods

We begin with a map of the disaster area (Figure 7.1(a): the map used in the figure is a part of Helsinki, the capital of Finland. The map is available as a part of the ONE simulator). The map contains connected road segments
onto a 2-D plane where possible movements of people and vehicle can occur. We assume that human lives in clustered neighborhoods and a few such neighborhoods are affected by the disaster. To construct neighborhoods, we randomly choose points on the map as neighborhood centers that are far away from one another by certain distance (say, 500m). Then, we place houses randomly around every center within a certain radius (say, 200m) from the center. Houses are located at the intersection of streets. Once the houses are built, people are created and are randomly assigned to particular houses. The house to which a person is assigned to is treated as his/her home. Figure 7.1(b) shows the city’s neighborhood centers and houses. The various values—the number of total neighborhoods, houses per neighborhoods, total number of people, minimum distance between neighborhoods, the radius of neighborhood—are parameters to the model.

![City Map](image1.png) ![Neighborhood in the city](image2.png)

(a) City Map  (b) Neighborhood in the city

Figure 7.1: Various components of the mobility model.

### 7.1.2 Post-Disaster Relief Operation

The relief operation begins after the disaster. A set of centers are declared to participate in the recovery operation. A few centers are pre-established (e.g., fire-stations), and a few are prepared in some premises for rescue and relief operation (e.g., medical centers, relief camps). There are three key components in modeling the disaster operation, placements of centers, mobile agents, and their interactions.

**Centers** Our model includes a number of different center types, such as relief centers, evaluation camps, medical centers and hospitals, and police
stations. These are at static locations in the map and are commonly visited repeatedly by moving agents.

**Mobile Agents and Mobility Patterns** Rescue workers are the main moving agents in the disaster response, as well as vehicles running between centers, camps, and stations to carry supplies, services, and aid workers. We constrain all movement to take place along the streets on the map, whereas centers can be located in any intersection of streets. We identify four basic types of mobility patterns undertaken by the agents (Figure 7.2(a)).

![Figure 7.2: Different mobility patterns](image)

(a) Four different mobility patterns

- **Center-to-Center**: This mobility is observed by vehicles which travel back and forth between a set of designated centers, camps or stations. In each trip starting from its home center, the agent picks a destination center from a set, finds a route, and moves toward the destination. After reaching the destination, it waits for a random duration as service time. After the service at the destination, it returns back to the home center. The oscillation repeats during the operation with some pause time in between.

- **Event-driven**: This type of movement is made when a specific event is notified to a designated center and the associated agent (i.e., vehicle) visits the incident area. After the service, the agent returns to the base. This movement does not however oscillate, but occurs once as event triggers.

- **Cyclic route**: Some agents take cyclic route from a particular center, visits a few locations of interest and returns to the home center. This is
mainly observed by police as a part of their patrol, or any public transport system (bus/tram routes). After visiting each location, agents can optionally take a random walk around the location for a while before heading to the next location.

- **Convergence-Move:** A set of agents with a particular role/duty get back to their reporting center around a certain fixed time, e.g., when all rescue workers are called back to relief camp for some special instruction. The opposite pattern is the divergence-Move.

Our model includes the following moving agents:

- **Supply vehicles** to carry relief goods between main centers and evacuation centers. They follow center-to-center mobility pattern.

- **Rescue workers** at each neighborhood to help people to evacuate, and later to assist people with relocation. They move from the relief centers to houses and the reverse.

- **Ambulances and fire trucks** that respond to emergencies (i.e., event-driven mobility). Emergency events are generated at random locations at random times. An ambulance or fire truck starts from its respective home center to visit the target place. After the on-site service (with random duration) the vehicle returns back to the home center.

- **Police patrol cars** originate from police stations and regularly visit neighborhoods. Obviously, they take the ‘cyclic route’ pattern. At the beginning of a patrol, a set of neighborhoods are chosen randomly. The patrol car starts from the police station and moves to the first neighborhood center. After reaching the neighborhood, it randomly visits a couple of locations (blocks) in that neighborhood with a random wait time at every location. Then, it chooses the next neighborhood and visits a few places there. Finally it returns to the police station. After a while, it picks another patrol.

- **Volunteers:** People within a neighborhood join with rescue workers and move randomly in the neighborhood. The difference between volunteers and the rescue workers is that volunteers do not report to the relief camps, rather they return to their homes.
7.2 Implementation

We extended the ONE with our new mobility model. ONE is a highly cus-
tomizable communication network simulator for delay tolerant networking
that has several movement models implemented that import map data and
constrain entity movement to the streets and roads of the imported data.
ONE can also visualize the imported map and entity movement using a GUI
which helps on validating the model in an intuitive way [55]. All map-based
movement models obtain their configuration data using files formatted with
a subset of the Well Known Text format. These files can be edited and gen-
erated from real world map data using Geographic Information System soft-
ware such as ArcView, or OpenJUMP (www.esri.com, www.openjump.org).
With map-based movement models, the entities move using roads and walk-
ways from the map data. In addition, different entity groups can be assigned
to different maps, constraining their movement in certain parts of the entire
map.

ONE comes with three types of map-based movement models. The Map-
Based Movement model is a derivative of the Random Walk model, where
entities move to randomly determined directions on the map following the
roads. Random pauses occur at the waypoints. The Shortest Path Map-
Based Movement model is a derivative of the Random Waypoint model,
where at decision points entities choose a random destination and then follow
the map-based shortest path to that destination. It is also possible in ONE
to specify in the configuration file deterministic routes for entities to follow.

7.2.1 Extension to ONE

We added new map based movement models to ONE, extending the Shortest
Path Map-Based Movement Model. These are listed in Table 7.1.

These movement models are a cross between completely deterministic spec-
ification, and completely random movement. In BwtnCenterMovement a ve-
icle cycles between two sets of centers. The sets themselves are deterministi-
cally specified by the configuration file, but for each travel segment a member
of the destination set is chosen randomly. A vehicle may pause for a random
duration at a cycle as well. The RescueWorkersMovement model is similar,
with a rescue worker cycling between a set of centers and a set of houses

154
Table 7.1: Newly added Movement classes to ONE.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>StaticMovement</td>
<td>places immobile centers, camps, houses, police stations, on the map</td>
</tr>
<tr>
<td>BwtnCenterMovement</td>
<td>movement of vehicles between centers;</td>
</tr>
<tr>
<td>HumanMovement</td>
<td>movement of humans; move to evacuation centers and return;</td>
</tr>
<tr>
<td>RescueWorkersMovement</td>
<td>rescue workers; travel between evacuation center and houses</td>
</tr>
<tr>
<td>PolicePatrolMovement</td>
<td>movement of police officers patrolling around the area</td>
</tr>
</tbody>
</table>

in a neighborhood. The PolicePatrolMovement models a patrol car that moves from its home base and visits a random sequence of neighbors, before returning to its home base.

ONE is written in Java. Each movement model is implemented as a class which implements the generic MovementModel interface. The interface MovementModel has two abstract methods that any extended class must override.

```java
interface MovementModel {
    protected MapNode getInitialLocation();
    protected Path getPath();
}
```

The method getInitialLocation returns the initial location of the entity, while getPath returns the next path (an array of road segments) to the new destination. A particular movement model has to override these two methods depending on what specific computation that model needs to make.

7.2.2 Configuration Parameters

We use the following random distributions for the various quantitative parameters used in the model.
<table>
<thead>
<tr>
<th>Type of variate</th>
<th>Distribution</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergency events</td>
<td>Exponential</td>
<td>rate ($\lambda$)</td>
</tr>
<tr>
<td>Vehicle speed</td>
<td>Normal</td>
<td>Mean $\mu$, variance $\sigma^2$</td>
</tr>
<tr>
<td>Emergency service time</td>
<td>Pareto</td>
<td>min ($x_{min}$)</td>
</tr>
<tr>
<td>People at home</td>
<td>Bernoulli</td>
<td>Prob ($p$)</td>
</tr>
<tr>
<td>People stay-time at camp</td>
<td>Pareto</td>
<td>min ($x_{min}$)</td>
</tr>
</tbody>
</table>

ONE configures itself based on input. To illustrate we show the settings for neighborhoods below. We place 10 neighborhoods (group 1) randomly located on map 1 (several map files can be configured), where neighborhoods would be at least 500m away from each other, allowing houses to be located within 200m of neighborhood center. Neighborhoods are immobile, so they use StaticMovement model (which is already part of ONE). We also show settings for supply vehicles (Group 5). They move between main coordination centers and evacuation centers. A vehicle’s speed is sampled from a positive normal with mean 50 km/hr and standard deviation 5 km/hr, while wait time is distributed uniformly between 20 mins and 60 mins.

```plaintext
G5.centerType = supply
G5.nrofNodes = 10
G5.homeCenter = maincoordcenter
G5.targetCenters = relief, e-camp
G5.okMaps = 1
G5.movementModel = BwtnCenterMovement
G5.speedDist = normal
G5.speed = 50, 5
G5.waitTimeDist = uniform
G5.waitTime = 1200s, 3600s
```

Here we see use of one of our new mobility patterns `BwtnCenterMovement` that describes movement between centers. We also use `StaticMovement` for designated centers.
CHAPTER 8
RELATED WORK

We explored variety of recent literature relevant to our proposal. Table 8.1 lists a set of notable projects undertaken by various institutions. We briefly discuss existing works on i) DTN networking protocols, iii) redundancy minimization protocols and content-aware networking, and iii) DTN applications for disaster communication.

8.1 DTN Networking Protocols

DTNs have a wide range of applications, which vary from acoustic underwater networks [56] to networks of vehicles [57]. While there has been no explicit DTN model for disaster scenarios, there exist several general architectures and models for DTNs in past literature [58, 59]. Most noticeably, there has been a significant amount of work on routing in disruption tolerant networks [60, 18, 61, 58, 62, 63, 64].

An important category of DTN routing schemes relies on devoting specific nodes (data mules), whose mobility is controlled, to deliver messages among others. Message Ferrying [65, 66] is an example of such protocols, designed for systems with predictable mobility patterns. Other approaches also take advantage of mobile nodes as intermediate ferries, where these nodes can change their trajectories based on requests they receive [63].

Some (e.g., [62, 7, 67] concentrate on a non-deterministic DTN with cyclic mobility patterns and model the network as a space-time graph. Some consider (e.g., [68]) deterministic mobility and solves the routing problem as utility optimization problems. Others [69] consider a deterministic and centralized DTN where contacts are enumerated by edges and indexed by time in a very large graph. Then they apply graph algorithms to derive results subject to network constraints.
<table>
<thead>
<tr>
<th>Project Name</th>
<th>Host Institution</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTNRG</td>
<td>IRTF’s DTN Research Group</td>
<td><a href="http://www.dtnrg.org/">http://www.dtnrg.org/</a></td>
</tr>
<tr>
<td>TIER</td>
<td>UC Berkeley</td>
<td><a href="http://tier.cs.berkeley.edu/drupal/">http://tier.cs.berkeley.edu/drupal/</a></td>
</tr>
<tr>
<td>Bytewalla</td>
<td>Royal Institute of Technology, KTH</td>
<td><a href="http://www.tslab.ssvl.kth.se/csd/projects/092106/">http://www.tslab.ssvl.kth.se/csd/projects/092106/</a></td>
</tr>
<tr>
<td>DakNet</td>
<td>MIT</td>
<td>Commercialized through <a href="http://www.firstmilesolutions.com/">http://www.firstmilesolutions.com/</a></td>
</tr>
<tr>
<td>ION</td>
<td>University of Ohio and NASA</td>
<td><a href="https://ion.ocp.ohiou.edu/">https://ion.ocp.ohiou.edu/</a></td>
</tr>
<tr>
<td>DieselNet</td>
<td>UMASS</td>
<td><a href="http://prisms.cs.umass.edu/dome/umassdieselnet">http://prisms.cs.umass.edu/dome/umassdieselnet</a></td>
</tr>
<tr>
<td>ResiliNets</td>
<td>University of Kansas and Lancaster University</td>
<td><a href="http://wiki.ittc.ku.edu/resilinets/">http://wiki.ittc.ku.edu/resilinets/</a></td>
</tr>
<tr>
<td>Haggle</td>
<td>EU Consortium</td>
<td><a href="http://haggleproject.org/">http://haggleproject.org/</a></td>
</tr>
<tr>
<td>DTN networking</td>
<td>Helsinki University of Technology</td>
<td><a href="http://www.netlab.tkk.fi/u/fo/dtn/index.html">http://www.netlab.tkk.fi/u/fo/dtn/index.html</a></td>
</tr>
<tr>
<td>EDIFY</td>
<td>Lehigh University</td>
<td><a href="http://edify.cse.lehigh.edu/">http://edify.cse.lehigh.edu/</a></td>
</tr>
<tr>
<td>DoDWAN</td>
<td>University of South Brittany</td>
<td><a href="http://www-irisa.univ-ubs.fr/CASA/DoDWAN/">http://www-irisa.univ-ubs.fr/CASA/DoDWAN/</a></td>
</tr>
<tr>
<td>Tetherless Computing</td>
<td>Waterloo</td>
<td><a href="http://blizzard.cs.uwaterloo.ca/tetherless/">http://blizzard.cs.uwaterloo.ca/tetherless/</a></td>
</tr>
<tr>
<td>PodNet</td>
<td>SKTH Stockholm and ETH Zurich</td>
<td><a href="http://podnet.ee.ethz.ch/">http://podnet.ee.ethz.ch/</a></td>
</tr>
<tr>
<td>N4C</td>
<td>EU/FP7</td>
<td><a href="http://www.n4c.eu/">http://www.n4c.eu/</a></td>
</tr>
<tr>
<td>Saratoga</td>
<td>University of Surrey</td>
<td><a href="http://personal.ee.surrey.ac.uk/Personal/LWood/dtn/saratoga/?/">http://personal.ee.surrey.ac.uk/Personal/LWood/dtn/saratoga/?/</a></td>
</tr>
<tr>
<td>SPINDLE</td>
<td>BBN</td>
<td><a href="http://www.ir.bbn.com/pbasu/projects.html">http://www.ir.bbn.com/pbasu/projects.html</a></td>
</tr>
</tbody>
</table>

Table 8.1: Notable DTN projects (compiled from http://www.dtnrg.org).
A different approach to DTN routing is one where node mobility is not under the control of the routing algorithm. Epidemic routing [60] is the most basic of these methods that replicates messages whenever a node meets another node which has not received the same message so far. Epidemic dissemination gives the best delivery ratio and delay if storage and transfer bandwidth are not limited. SWIM (shared wireless infostation model) [70] introduces info-stations as ‘healing’ centers where a ‘diseased’ node dispatches its messages and attains ‘immunity’ from future ‘infection’. A more recent work [71] presents the resource and performance trade-offs of deployment of these infostations. Some research uses network coding [72, 73] on top of replication to achieve better delivery in the case of high failure rates. There are a few variants of epidemic algorithms that gather information from the network, with particular emphasis on encounter patterns. PROPHET [29] estimates delivery predictability to destinations using the history of encounters and the transitive property of meetings with nodes. MaxProp [18] computes delivery probabilities from meeting frequencies and sorts messages in the transmission buffer accordingly. Despite the good delivery ratio, epidemic routing and its variants incur very high communication (and, hence, energy) overhead and are thus inappropriate for disaster-response networks.

There exist schemes where the number of times a message can be replicated is pre-specified. Example includes Spray and Wait [12] that limits the total number of copies created initially (spray phase). Replicas then wait for a direct delivery opportunity to the destination (wait phase). A variant of this algorithm, called Spray and Focus [16], introduces utility-based forwarding of the last copy instead of just waiting to meet the destination. Although Spray reduces overhead, it does not assume regularity in mobility patterns and contacts. Hence, it achieves generality at the expense of losing opportunities for further optimization.

Yet another approach to DTN routing is to use prior information about network connectivity. This information enables the protocol to achieve very high delivery rates at low overhead. A framework is described in past literature [58] that formulates the DTN routing problem given different amounts of prior information about the network. MEED (minimum estimated expected delay) [61] estimates expected delay for each specific contact during network operation. Using this delay as a link metric, routing is done in a link-state fashion by propagating each individual link delay. In [62], scalable
routing is investigated in deterministic DTNs where mobility follows strictly periodic patterns. They propose a DTN Hierarchical Routing (DHR) scheme that yields the optimal time-space algorithm in terms of delay and hop-count. RAPID [64] models DTN routing as a utility-driven resource allocation problem and computes delay-based utility values of messages assuming exponential distribution of inter-node meeting times. There also DTN oracles, such as [74, 75]. [76, 77] propose data replication policies in DTNs.

There are forwarding schemes [78], [79], [80] that rely on comparisons between per-node metrics to make forwarding decisions. They intend to reduce communication overhead, while achieving a level of performance close to that of epidemic algorithms. Spyropoulos et al. [81] present an analytical framework for the forwarding-only case (allowing only one message copy) on a grid-based mobility model. FRESH [80] relies on a node’s last meeting time with the destination to make a forwarding decision. Greedy [78] relies on contact rate with the destination and greedy-total uses the total contact rate of a node. Delegation forwarding [82] proposes a metric-based general forwarding scheme; it assumes that each node has an associated ‘quality’ metric and a node forwards a message if it encounters another node whose quality metric is greater than any seen by the message so far. Analysis tried commonly-cited metrics, such as frequency and last-contact time. Our approach incorporates \textit{inter-contact delay} as a new metric in making decisions.

Our analysis of end-to-end delay bound for DTNs is relevant to works that compute timing properties of distributed systems. Accurate analysis methods, such as [83, 84], construct a precise schedule of length equal to the hyper-period. Offline schedulability tests [85, 86, 87] analyze distributed systems by dividing end-to-end deadlines of tasks into per-stage deadlines. Holistic schedulability analysis [88, 89] considers the worst-case delay of each stage of computation to derive the jitter of next stage, adding up delay across paths. Network Calculus [90, 91] analyzes the network one node at a time. Delay Composition Algebra was recently introduced [92, 20] as a reduction-based approach, which reduces the entire distributed system to a hypothetical single node. It takes into account execution pipelining effects between subtasks running in a distributed system and provides a good upper bound for end-to-end delay of tasks.
8.2 Content-aware Networking and Redundancy Reduction Techniques

Our in-network redundancy reduction techniques belong to content-aware networking. In [93], a content service model was proposed with content brokers, so that content can be stored and disseminated efficiently based on users’ requests. CNF (cache-and-forward) [94] proposed a similar idea where a few gateway nodes at the edge of the network store data items so that mobile agents can obtain their data once they are online. Other work addressed content caching [95] and content aware routing [96, 97]. CCN [24] introduced networking of named content, in place of IP networking by host naming. The above schemes, despite their efficient retrieval and dissemination of content, are not content-aware in that they do not explore semantics of content.

Data fusion literature described semantic-aware content fusion methods. Semantic fusion has usually two phases: knowledge base construction and pattern matching [98]. At first phase, a suitable abstraction for representing semantic information is chosen, which is then used in second phase for matching and fusing relevant attributes. This fusion runs in-network inference processes so that nodes only exchange semantic interpretations. Another work [99] integrates sensor data into formal languages, and then matches data with some stored knowledge base based on the hypothesis that data represented by similar languages are semantically similar. Semantic streaming [100] allows users to formulate queries over semantic values without specifying data or operations. SONGS architecture [101] uses declarative queries and converts queries into service composition graph.

Camera sensor networks have received great attention over recent years [102]. Camera based sensor platforms focused image recognition and activity recognition ([103]) that have applications to surveillance [104], habitat monitoring [105], security systems [106], and assisted living [107]. As cameras are becoming more ubiquitous in recent years, a set of participatory applications, in the form of “urban image sensing”, are also emerged, such as microblogging [108] and telemedicine (e.g. documenting diets [109]).

There have been works on mobile phone-based data collection and retrieval systems. Works proposed in [110, 111] use 3G networks on mobile phones, vehicle based DTNs, and available nearby WiFi access points to transfer HTML pages against user queries. Cartel project [112] develops a mobile
sensing system, in the form of a Web portal service. A distributed image retrieval service on a sensor platform is proposed in [113]. Works [114, 115] propose cooperative caching in DTNs.

Information retrieval (IR) community has worked at length on information retrieval system that considers redundancy and novelty of retrieved information ([116, 117, 118, 119]). In most cases, these works are for text documents. Content-based image retrieval systems ([120, 121]) work for querying images from a set of given pool of images. These services mainly index an initial collection of pictures at the server and then return images that are visually very close to the queried image.

8.3 Caching Techniques in DTNs

Caching is an effective way of improving data availability in data networks and web services [122]. Lately it has been extensively used in mobile environments as well [123, 124, 125, 126]. In mobile environments, cooperative caching is usually used that allows sharing and coordination of cached data among multiple nodes.

Cooperative caching has been studied in web environments [122] and in wireless mobile networks [127, 128, 129, 130, 131, 132, 133]. Hara [134] proposed replica allocation methods to increase data accessibility and to handle network partitions in MANETs. Although replication can improve data accessibility, the overhead for relocating replicas is significantly high. Sailhan and Issarny [129] proposed a cooperative caching scheme that improves data accessibility by P2P communication among mobile hosts, when they cannot avail the fixed infrastructure. Lau et al. [127] proposed an architecture for supporting continuous media proxy caching by transparently performing data allocation and session migration among all proxy caches. Nuggehalli et al. [135] addressed the problem of optimal cache placement and proposed a greedy algorithm, called POACH, which minimizes the weighted sum of energy and latency. A broadcast based cooperative caching scheme for hybrid networks is suggested in [136] where a client shares its caches with clients lying in its proximity. Yin and Cao [137, 138] design and evaluate several caching algorithms to efficiently support data access in ad hoc networks. Chand et al. [139] proposes zone cooperative caching that allows nodes to
form zones based on their proximity with other nodes. Zang et al. [140] discuss about security concerns for cooperative caching in ad hoc networks.

There has been work for efficient data dissemination in intermittently connected networks and DTNs. Several approaches are considered: interest profiles [114], publish/subscribe [141, 142], subscription of channels [143, 144]. In [145], the authors studied caching where nodes cache pass-by data based on data popularity, so that queries in the future can be responded with less delay. Some research efforts [146, 147] improve data accessibility from infrastructure network such as WiFi Access Points (APs) [147] or Internet [146]. While distributed caching techniques (e.g., [148, 149]) generally assume that all nodes can perform caching functions, recent works, such as [50], proposed a set of key locations as the position for caching nodes. These locations are usually highly reachable from others.

Replacement policy is one of the key issue in caching techniques. Replacement policy refers to the rule by which a cached object is selected for eviction in order to accommodate a new object in the cache when the cache becomes full. A few notable replacement policies are LRU (least recently used), LFU (least frequently used), LRU-\(k\) (LRU with last \(k\) access records), greedy-dual-size (considers data popularity as well as data size) [122], VALUE based (considers various attributes, such as last access time, frequency of access, time-to-live and object size) [139]. One fundamental contrast with our approach is that these techniques treat all stored objects independent of one another so eviction rule considers each object in isolation. Whereas in our diversity caching technique, we consider the case when objects are non-independent and they have relative values or importance in the presence of others.

8.4 DTN Mobility Models

Mobility models for mobile wireless networking is an active research area. Random Walk and Random Waypoint (RWP) [150] are the simplest and most widely used mobility models. Levy walks [151] are similar to random walks, except that the flight lengths and pause times are drawn from a power law distribution. It is believed that unconstrained human movements follow levy distribution [151]. Some model group mobility, such as Exponential Correlated Random Mobility [152] and Reference Point Group Mobility ([153]).
Works, such as [154, 155] propose mobility in group with higher social attractiveness (friendship with others). Working Day Movement (WDM) model [55] presents the everyday life of people who go to work in the morning, spend their day at work, and commute back to their homes at evenings. Nelson et al [156] describe a role-based and event-driven mobility model for disaster recovery networks, where agents’ speeds are accelerated (modeled by laws of gravity) when they approach to disaster scene (e.g., police) and retarded when flee away (e.g., people).

8.5 Applications for Disaster Communication

DTNs have a wide range of applications, which vary from acoustic underwater networks [56] to networks of vehicles [57]. While there has been no explicit DTN model for disaster scenarios, there exist several general architectures and models for DTNs in past literature [58, 59]. A set of web applications and social networking during disaster are listed in [157, 158].
CHAPTER 9

CONCLUSION AND FUTURE WORK

This thesis promises to investigate problems that arise in disaster communications. Although various aspects of disaster time communication elegantly provided context for our proposed protocols and services, we believe the protocols and services have more general application in other contexts as well. Below, we reiterate our main contribution first, and then present a few future research challenges and problems related to the topics discussed in this dissertation.

9.1 Summary of Contribution

We consider DTNs to be the primary mode of communication during disaster. In the consideration of building DTN protocols, we identify two fundamental aspects of DTNs: i) moving nodes and ii) moving data. Accordingly, we develop techniques that leverage certain physical aspects of moving nodes and propose methodologies to address certain logical properties of generated data content. These two aspects are recurrence and redundancy respectively. We propose a novel routing protocol, named inter-contact routing [159, 160], that exploits recurrence observed in mobility patterns of nodes. We then consider the problem of redundancy in human generated data content and propose a set of redundancy reduction techniques. We introduce methods for measuring diversity of a content collection and then devise a set of content-aware prioritization (CAP) protocols that enable diversity-maximizing content storage and dissemination in the network. We design and implement application services, such as PhotoNet [161, 162], PhotoNet+ [163] and diversity caching. For experimentation, we develop the Post-Disaster Mobility Model [164], and instrumented the model into the ONE simulator.
9.2 Future Research Directions

In the following, we describe a couple of future research problems and directions relevant to the topics presented in this thesis.

9.2.1 Multi-Modal Communication for Disaster Response

In recent years, portable human-carried devices are equipped with many communication options, such as WiFi, GPRS (2G), 3G, Bluetooth, WiMax, and near field communication. Leveraging all available communication paradigms and their interoperability opportunities could be a great potential for disaster time communication. Bluetooth is a lightweight protocol for discovering other bluetooth-enabled devices and is good for short range communication (within a couple of meters) at lower data rate. WiFi is good for bulk transfer and usually works with infrastructure support (i.e., with access points). WiFi has provisions for creating ad hoc (mesh) networks. A recent standard, WiFi Direct [165], enables discovering devices over WiFi and allows peer-to-peer data transfer between two WiFi enabled devices. 3G and WiMax are technologies for long range communication and cannot work without infrastructure support (very unlikely to remain available during or after a disaster). Software driven radios and cognitive radios also have opportunities to be useful in disaster context. Due to ubiquity of WiFi, we believe WiFi could be the most viable option for wireless communication in disaster mode. Popular off-the-shelf tethering services, such as USB, Bluetooth and WiFi tethering, available in today’s mobile phones allow data connection available in one device to be shared with others over multiple communication links. Taking advantages from different connectivity options and combining them in an effective synergy is an important research direction.

9.2.2 Energy Efficiency in Networked Devices

Energy is a critical resource for portable devices and is possibly one of the most challenging aspects of device manufacturing to date. While usual capabilities, such as processing, memory, storage and communication, all scale at a rate close to Moore’s law (double is every 18 months), the same is not true at any least for power sources. The advancement in battery capacity is rather
slow. So energy efficient networking protocols are essential. One particular concern for DTN is the huge amount of energy consumption due to device idling. For instance, in Android platform, the most popular open platform for mobile computing, phones reportedly consume nearly 45% of their total energy for idling, followed by 28% for cell standby (listening over wireless carrier for accepting possible voice call). In disaster mode, the later task can be disabled, because cell towers may not be functioning. Currently available off-the-shelf phones perhaps do not allow that. To reduce power consumption due to idling, low power idling mode, for example, the ones introduced by sensor network community, such as Micaz [15], can be harnessed. Duty cycling could be another possibility. One potential reason for which devices need to remain ON is to discover other nearby devices. Discovering devices while in low power mode or on duty cycling is a daunting research problem. Energy harvesting in protocol design is another important research direction.

9.2.3 Rich Sensors for Wide Area Sensing

Current mobile phones are equipped with a rich set of sensors, such as accelerometer, magnetometer, proximity, and GPS sensors. Even microphone and camera can be regarded as sensors capable of capturing and recording rich media data, namely audio and image. People can record various events and conditions of their surrounding environment either automatically without paying much attention to tasks (e.g., recording GPS traces) or with deliberate attention (e.g., taking a shot of a scene by the camera). Extracting useful information from these readings to detect important events and thus enabling a wide area sensing is an important research problem. Sensor readings can also be incorporated into improving communication capabilities of these devices; for example, user movement indicated by accelerometer can help in deciding better data transfer rates over WiFi links. A few recent studies [166, 167] demonstrated a few early results in this direction.

9.2.4 QoS, QoE, now QoI?

Human carried portable devices, such as mobile phones, can no longer treated as devices that merely consume bits produced by some services at the other
end usually hosted at a server or in a cloud. In recent days, humans are also generating information and are feeding information to the services. During disaster and any crisis times, this sort of data generation becomes very dominant. Wide spread proliferation of social networking (e.g., facebook) and web media streaming sites (e.g., Twitter) makes this kind of dissemination quite easy. While the first type of information flow, i.e., from service points to end users, usually comprises of finished-products (well prepared information by their respective producers), the later type of information flow, generated by mass people, is not filtered and distilled to be readily consumable. Extracting useful information and knowledge from these crowd generated data content is a very daunting research problem these days. Along side the popular “Q” terms such as quality of service (QoS) and quality of user experience (QoE), this new direction coins yet another new term QoI (quality of information).

9.2.5 Information-centric Networking

Our diversity-maximizing protocols, such as PhotoNet and diversity caching, serve as tools for answering a higher-level network design objective. Namely, we consider that the main function of the network is one of maximizing information transfer per unit cost. In contrast to assuming data objects as an innocent stream of consecutive bits, we rather attempt to reason about higher level aspects of data objects as a chunk of some meaningful information. Incorporating such reasoning as a network-level functionality leads to information-centric networking design, which caught much attention from research community in recent years. Our coverage-maximizing communication, caching and prioritization protocols contribute to this broader domain of information-centric networking. We envisage to investigate more aspects of these techniques and to add more extensions to current work.

9.2.6 Content-aware Prioritization and Named-Data Networking

We worked on in-network prioritization of content, which is based on the premise that not all bits in a networked system is equally important to be transferred and stored, especially on occasions when limited capabili-
ties (transmission bandwidth or storage) do not allow to transport and store them all. In our diversity-maximizing protocols, data content has marginal diversity gain that contributes to a relative ordering of content in transfer queue and dropping order from storage.

Traditional networking protocols do not have adequate provisions for supporting this kind of prioritization of data packets (IP has support by explicitly labeling packets with special flag bits). A recent proposal in the name of Future Internet Architecture, called Named-Data Networking (NDN) [24, 47], has greater promise in this respect. Unlike IP, NDN names data content instead of naming host machines. NDN proposes a hierarchal naming structure for data content and an IP-like prefix based routing mechanism, implemented in routers, in order to lookup and deliver data objects described by a specific name. NDN is suitable for application-specific prioritization. If routers had the ability to group content by prefix and apply different prioritization schemes to content of different prefixes, then an application can simply associate itself with a prefix in the name space and have all content that matches that prefix get prioritized in an application-specific way.

The application-specific prioritization mechanism can provide the ability to prioritize packets inside the network in a manner that is content-aware. In other words, it allows content-aware network resource allocation, where resources such as bandwidth, energy, and storage are appropriately allocated (i.e., prioritized) across competing content in the same application. Moreover, this content-aware resource allocation can be implemented without much violation to layering because the application-specific prioritization can be done for only content in the application’s own name subtree. It also does not violate network neutrality which says that networks should be fair to all content. This is because we alter prioritization only among content items that belong to the same application, and hence do not give one application an edge over another (which would violate network neutrality). The ability to do content-aware prioritization can significantly improve application-perceived network performance as exemplified by our PhotoNet services.

While traditional network layer protocols, such as IP, have provisions for QoS (Quality of Service) via differentiated services, the problem we discussed falls into the broader domain of QoI (Quality of Information). We believe content-centric QoI (Quality of Information) would enhance the popular IP-based QoS services and would ultimately become one of the core function-
alities of NDN, which would ultimately help to standardize much of our proposed practices for more general applications.
REFERENCES


[70] T. Small and Z. Haas, “The shared wireless infostation model: A new ad hoc networking paradigm (or where there is a whale, there is a way),” in Proc. of MobiHoc, 2003.


