Title
Middleware Approaches to Societal Scale Information Sharing

Permalink
https://escholarship.org/uc/item/4g69m7q2

Author
Zhao, Ye

Publication Date
2014

Peer reviewed|Thesis/dissertation
UNIVERSITY OF CALIFORNIA, IRVINE

Middleware Approaches to Societal Scale Information Sharing

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Networked Systems

by

Ye Zhao

Dissertation Committee:
Professor Nalini Venkatasubramanian, Chair
Professor Sharad Mehrotra
Professor Amelia C Regan
Professor Ramesh Jain
Professor Magda El Zarki

2014
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>iv</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>v</td>
</tr>
<tr>
<td>CURRICULUM VITAE</td>
<td>vi</td>
</tr>
<tr>
<td>ABSTRACT OF THE DISSERTATION</td>
<td>ix</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Thesis Problem and Scope</td>
<td>9</td>
</tr>
<tr>
<td>1.3 Thesis Contributions and Organization</td>
<td>13</td>
</tr>
<tr>
<td>2 Related Work</td>
<td>17</td>
</tr>
<tr>
<td>3 Instant Information Sharing: Information Layer</td>
<td>24</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>26</td>
</tr>
<tr>
<td>3.1.1 Related Work</td>
<td>29</td>
</tr>
<tr>
<td>3.2 An Overview of DYNATOPS</td>
<td>31</td>
</tr>
<tr>
<td>3.2.1 Architecture</td>
<td>31</td>
</tr>
<tr>
<td>3.2.2 Dynamic Mappings</td>
<td>32</td>
</tr>
<tr>
<td>3.3 DYNATOPS User Broker Mapping</td>
<td>33</td>
</tr>
<tr>
<td>3.4 DYNATOPS reconfiguration</td>
<td>36</td>
</tr>
<tr>
<td>3.4.1 Preliminaries</td>
<td>36</td>
</tr>
<tr>
<td>3.4.2 Reconfiguration Process</td>
<td>38</td>
</tr>
<tr>
<td>3.4.3 Reconfiguration Computation</td>
<td>40</td>
</tr>
<tr>
<td>3.4.4 Reconfiguration Protocol</td>
<td>43</td>
</tr>
<tr>
<td>3.4.5 Analysis</td>
<td>46</td>
</tr>
<tr>
<td>3.5 Performance Evaluation</td>
<td>50</td>
</tr>
<tr>
<td>3.5.1 Dynamic Subscriptions Modeling</td>
<td>50</td>
</tr>
<tr>
<td>3.5.2 Comparison Systems</td>
<td>52</td>
</tr>
<tr>
<td>3.5.3 Experimental Results</td>
<td>54</td>
</tr>
<tr>
<td>3.6 Conclusion</td>
<td>61</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>4</td>
<td>Instant Information Sharing: Dissemination Layer</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>4.2</td>
<td>GSFord Approach</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Social Concepts of GSFord</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Geographical Concepts of GSFord</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Overlays of GSFord</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Event Notification over GSFord</td>
</tr>
<tr>
<td>4.3</td>
<td>GSFord Overlays: RRTree-based Geo-social Aware Overlays</td>
</tr>
<tr>
<td>4.3.1</td>
<td>RRTree-based Geo-aware P2P Overlay Structure</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Delivery Overlay: Efficient Regional Multicasting</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Information Overlay: Social Information Storage</td>
</tr>
<tr>
<td>4.4</td>
<td>Event Notification</td>
</tr>
<tr>
<td>4.5</td>
<td>Evaluation</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Evaluation Settings</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Reachability of GSFord</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Performance of RRTree-based Geo-aware Overlay</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Implementation of GSFord</td>
</tr>
<tr>
<td>4.6</td>
<td>Concluding Remarks</td>
</tr>
<tr>
<td>5</td>
<td>Delay-tolerant Information Sharing: Information Layer</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>5.2</td>
<td>Middleware Design Overview</td>
</tr>
<tr>
<td>5.2.1</td>
<td>SmartSource Architecture</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Scalability Consideration</td>
</tr>
<tr>
<td>5.3</td>
<td>System Model and Problem Formulation</td>
</tr>
<tr>
<td>5.3.1</td>
<td>System Models</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Problem Formulation</td>
</tr>
<tr>
<td>5.4</td>
<td>SmartSource Algorithms</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Design Principles</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Worker Selection Algorithm</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Assignor Selection Algorithm</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Performance Analysis</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Theoretical Proofs</td>
</tr>
<tr>
<td>5.5</td>
<td>Implementation and Performance Evaluation</td>
</tr>
<tr>
<td>5.5.1</td>
<td>SmartSource Implementation</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Performance Evaluation</td>
</tr>
<tr>
<td>5.6</td>
<td>Conclusion</td>
</tr>
<tr>
<td>6</td>
<td>Delay-tolerant Information Sharing: Dissemination Layer</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>6.2</td>
<td>A Mobile Middleware Architecture for Social Content Delivery</td>
</tr>
<tr>
<td>6.3</td>
<td>O2SM Mobile Middleware Algorithms</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Social Media Content Ranking Component</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Ranking-driven Social Media Prefetching</td>
</tr>
<tr>
<td>6.4</td>
<td>A Broker/Proxy Framework for Social Content Delivery</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Middleware for societal scale information sharing</td>
<td>8</td>
</tr>
<tr>
<td>1.2</td>
<td>Thesis problems and scope in quadrant presentation.</td>
<td>10</td>
</tr>
<tr>
<td>1.3</td>
<td>Thesis contributions in each quadrant of the societal scale information sharing framework</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>Contributions of Chapter 3 in the societal scale information sharing framework.</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>DYNATOPS architecture.</td>
<td>31</td>
</tr>
<tr>
<td>3.3</td>
<td>Example of rendezvous-based Pub/Sub.</td>
<td>32</td>
</tr>
<tr>
<td>3.4</td>
<td>Example of the structure and subscription aware reconfiguration.</td>
<td>34</td>
</tr>
<tr>
<td>3.5</td>
<td>DYNATOPS broker data structure.</td>
<td>44</td>
</tr>
<tr>
<td>3.6</td>
<td>Dynamic subscription model from Twitter dataset.</td>
<td>52</td>
</tr>
<tr>
<td>3.7</td>
<td>Location-based subscription model results.</td>
<td>55</td>
</tr>
<tr>
<td>3.8</td>
<td>Poisson subscription model results under varying rate of user subscription changes with multimodal pattern</td>
<td>56</td>
</tr>
<tr>
<td>3.9</td>
<td>Poisson subscription model results in different patterns (subscription change rate = 0.1, $T_{free} = 10hrs$).</td>
<td>57</td>
</tr>
<tr>
<td>3.10</td>
<td>Results under varying reconfiguration free period. For location-based model, $c = 10$; For Poisson model, subscription change rate = 0.025 and $c = 5$.</td>
<td>59</td>
</tr>
<tr>
<td>3.11</td>
<td>Performance savings of DYNATOPS configurations against CHash under various size of broker network $</td>
<td>B</td>
</tr>
<tr>
<td>4.1</td>
<td>Contributions of Chapter 4 in the societal scale information sharing framework.</td>
<td>63</td>
</tr>
<tr>
<td>4.2</td>
<td>The overview of GSFord.</td>
<td>66</td>
</tr>
<tr>
<td>4.3</td>
<td>The RRTree-based overlay structure.</td>
<td>72</td>
</tr>
<tr>
<td>4.4</td>
<td>Pseudo code of the geographical regional multicasting with RHT.</td>
<td>78</td>
</tr>
<tr>
<td>4.5</td>
<td>Pseudo code of geographical regional multicasting with the RRTree.</td>
<td>79</td>
</tr>
<tr>
<td>4.6</td>
<td>The concept of conjugate region.</td>
<td>81</td>
</tr>
<tr>
<td>4.7</td>
<td>An example of the notification process of GSFord.</td>
<td>84</td>
</tr>
<tr>
<td>4.8</td>
<td>Reachability comparison of GSFord with other systems.</td>
<td>90</td>
</tr>
<tr>
<td>4.9</td>
<td>Performance comparison of GSFord with other systems.</td>
<td>90</td>
</tr>
<tr>
<td>4.10</td>
<td>Reachability comparison between GSFord and GeoPeer.</td>
<td>93</td>
</tr>
<tr>
<td>4.11</td>
<td>Scalability of the GSFord overlay.</td>
<td>93</td>
</tr>
</tbody>
</table>
4.12 Average number of Backup RR of a PNode ........................................ 96
4.13 Overview of the prototype implementation of GSFord ...................... 97
4.14 Reachability vs. latency of the implemented delivery overlay ........... 98

5.1 Contributions of Chapter 5 in the societal scale information sharing framework. 100
5.2 Overview of the SmartSource system ............................................... 105
5.3 Software modules in SmartSource broker middleware ....................... 108
5.4 Geo-based nested tree structure ....................................................... 108
5.5 The representation of a sample optimization problem in minimum cost maximum flow problem ......................................................... 120
5.6 Prototype architecture ................................................................. 133
5.7 Screenshots of prototype: (a) submit a question, (b) display recommended questions, and (c) answer a question ........................................ 133
5.8 Statistics of the usage of the client app ............................................. 134
5.9 Comparing Smartsource with others on spatial-temporal Q&A, $k_{max}^r = 3$, 50% worker availability ......................................................... 137
5.10 Comparing Smartsource with others on spatial-temporal Q&A, $k_{max}^r = 6$, 33% worker availability ......................................................... 138
5.11 Performance tradeoff of Smartsource by varying $\gamma$ in spatial-temporal Q&A and varying $V$ in knowledge-based Q&A .......................... 140

6.1 Contributions of Chapter 6 in the societal scale information sharing framework. 144
6.2 The proposed system architecture and an example of social media stream metadata ................................................................. 148
6.3 Considered system architecture ....................................................... 164
6.4 Major software modules in our system ............................................ 165
6.5 Snapshots of the oFacebook app: (a) show a video content, (b) show an album content, and (c) set configuration parameters .......................... 184
6.6 Evaluations on the content ranker: (a) AUC for all users, (b) the ROC curve for three users ................................................................. 186
6.7 Evaluation on dataset ................................................................. 191
6.8 Evaluation using synthetic contents where 50% are viewed ................ 191
6.9 Evaluation using synthetic contents where 100% are viewed ............... 192
6.10 Evaluation using synthetic network connectivity .............................. 193
6.11 Performance of the algorithms by varying $V$ .................................. 195
6.12 Performance comparison of different algorithms ............................ 196
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>The computation time of the DYNATOPS-Greedy algorithm under various size of broker network $</td>
<td>B</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparison between GSFord and GeoPeer</td>
<td>91</td>
</tr>
<tr>
<td>6.1</td>
<td>Representation of user interactions</td>
<td>153</td>
</tr>
<tr>
<td>6.2</td>
<td>Energy consumption (J) at devices for offline access to Facebook in 30 hours</td>
<td>193</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

There are a number of people without whom this thesis might not have been written, and to whom I am greatly indebted.

My first debt of gratitude must go to my advisor, professor Nalini Venkatasubramanian. She patiently provided the vision, encouragement and advise necessary for me to proceed through the doctorial program and complete my dissertation. Her broad vision and direction guided me to discover novel research problems that are practical and valuable in real world. Her guidance on presentation and writing, her encouragement on attending meetings and networking benefit me greatly not just in researches but in my life as well.

I’m also very grateful to Dr. Kyungbaek Kim at Chonnam National University and Dr. Chenghsin Hsu at National Tsing Hua University for their advises and helps during the journey of research. They played critical roles in my research projects and I learned essential skills for team working, research development as well as paper writing from them. Their guidance has served me well and I owe them my heartfelt appreciation.

I appreciate supports from all of my labmates and colleagues, Zhijing Qin, Reza Rahimi, Kyle Bension, Shu-Ting Wang, and specially Nogc Do who worked closely with me in many of my projects, and being so supportive in all stages of the research. I could not complete my work without invaluable friendly assistance of him.

I wish to thank my parents Wenbo Zhao and Lina Gao for their love encouragement and supports both financially and mentally. My wife, Nan Wang, whose love and encouragement allowed me to finish this journey. My lovely daughter, Evangeline Zhao, and son, Ethan Zhao. They are sources of my laughter, joy, and my driving force.
CURRICULUM VITAE

Ye Zhao

EDUCATION

University of California, Irvine

Master of Science in Communications and Signal Processing 2004–2005
Imperial College London

Bachelor of Science in Communication Engineering 2000–2004
Beijing University of Posts and Telecommunications

WORKING EXPERIENCE

Graduate Research and Teaching Assistant 2008–2014
University of California, Irvine

Software Engineer Intern 2013–2013
Google, Inc.

Software Engineer Intern 2011–2011
Google, Inc.

System Engineer 2007–2008
Nortel Networks (China) Limited

Software Engineer in Test 2006–2007
Huawei Technologies Co. Ltd.
REFEREED JOURNAL PUBLICATIONS

A Reliable Geo-Social Notification System 2014
In submission

An Adaptive Topic-based Pub/Sub Architecture 2014
In submission

Offline Access to Online Social Networks 2014
In submission

A Crowdsourcing System for Mobile Internet Access 2014
In submission

REFEREED CONFERENCE PUBLICATIONS

SmartSource: A Mobile Q&A Middleware Powered by Crowdsourcing 2014
In submission

Optimizing Offline Access to Online Social Networks Apr 2014
IEEE Infocom

O2SM: Online and Offline Access to Social Media Dec 2013
ACM/IFIP/USENIX Middleware

DYNATOPS: A Dynamic Topic-based Publish/Subscribe Architecture July 2013
ACM DEBS

Modeling Path Duration Time in Hybrid Convergecast Network April 2013
IEEE WCNC

GSFord: Towards a Reliable Geo-Social Notification System Feb 2012
IEEE SRDS

Adaptive Channel Allocation for IEEE 802.11 Wireless LANs April 2006
IEEE EW
SOFTWARE

oFacebook http://www.ics.uci.edu/~dsm/oFacebook/
Android app for offline access to Facebook

Dynapubsub http://www.ics.uci.edu/~dsm/dynapubsub/
Chord based Topic Publish/Subscribe implementation
ABSTRACT OF THE DISSERTATION

Middleware Approaches to Societal Scale Information Sharing

By

Ye Zhao

Doctor of Philosophy in Networked Systems

University of California, Irvine, 2014

Professor Nalini Venkatasubramanian, Chair

Over the years, Internet users have exhibited an increasing demand for sharing information in a variety of application domains, and a large number of applications have emerged. Despite advances in Internet technologies, the following remain key challenges: (a) dynamicity of user context (e.g., locations and activities) causes rapid changes in terms of what to share and who to share with; (b) diverse performance requirements of information sharing applications (in terms of reliability, timeliness, accuracy and efficiency); (c) limitations in infrastructure (e.g., unreliable yet changing networks and resourced limited end devices). These challenges are further aggravated by large number of mobile users distributed over a wide geography.

In this thesis, we study information sharing at societal scale involving a large group of people across a large geography. We exploit the knowledge of geographical and social relationships between users to address the above challenges. We adopt a middleware approach that is resilient to the heterogeneity of the communication environments (i.e., networks and devices) and offers adaptive services to a verity of applications. We scope our work along 2 dimensions. Firstly, along the dimension of system layers our focus is on two layers: 1) the information layer - what to share: determine specificity of contents and accurately target information consumers and providers. 2) the dissemination layer - how to share: determine dissemination mechanisms to deliver information from its source to
targeted receivers to meet the performance goals. Secondly, along the dimension of timing constraints we consider two classes of applications: 1) *instant information sharing* and 2) *delay-tolerant information sharing* applications. We develop sharing techniques for multiple use cases.

Specifically, for instant sharing we design two systems. At the information layer, we propose and design DYNATOPS, a dynamic topic based publish/subscribe middleware to efficiently keep track of the large scale dynamic information interests of users, and provide efficient event notifications to subscribers. DYNATOPS organizes pub/sub brokers into a structured overlay. To adapt to the subscription dynamics and to maintain efficient event notifications, it strategically and moderately reposition pub/sub users on brokers and reposition brokers on the overlay. At the dissemination layer, we propose and design GSFord, a reliable notification middleware that aims to provide timely and reliable instant information dissemination in extreme situations (e.g., catastrophic disasters). GSFord builds robust geo-aware P2P overlays and provides reliable storage of geo-social information of users under extreme regional failures. It reliably delivers messages to unfailing recipients who are either geographically or socially correlated to the event and exploits a targeted social diffusion through diverse out-of-band channels to reach those in failed regions on a best efforts basis.

For delay-tolerant information sharing, at the information layer, we propose and implement SmartSource, a crowdsourcing based mobile Question & Answer (Q&A) middleware. SmartSource aims to provide mobile information seekers with timely, trustworthy and accurate answers while ensuring that information providers are not inappropriately burdened. It takes advantage of both static and dynamic context and semantics from mobile users (e.g., geolocation, social network, expertise/interest, device sensor profiles, battery level) to identify sources of information (i.e., providers) that are trusted by the user and accurate enough for the questions at hand. At the dissemination layer, we propose O2SM middleware that aims to enable mobile users to access to online social media contents anytime anyplace
without requiring to be online all the time. We develop the middleware to (i) rank the social
media streams by estimating probability that a given user views a given content item and (ii)
invest the limited resources (network, energy and storage) on prefetching only those social
media streams that are most likely to be watched when mobile devices have good Internet
connectivity. As a proof of concept we implement an Android app, oFacebook, to provide
mobile users with uninterrupted access to Facebook.
Chapter 1

Introduction

In this thesis, we study information sharing at societal scale involving a large group of people across a large geography. With the rapid evolution of web and mobile technologies, societal or community information sharing applications have penetrated many aspects of our day-to-day life, e.g., entertainment, education, transportation, emergency response, and etc. For example, the phenomenal popularity of social networks such as Facebook, Twitter, LinkedIn, and Google+ has changed the way people interact. Many people nowadays rely on online social networks to communicate with their friends, family, and community on a day to day basis. Studies show that over 800 million people log onto Facebook daily and every minute Facebook users create over 510000 comments, 695000 status updates to interact with their friends.

The eventual goal of information sharing is to fulfill people’ information needs in diverse applications through the delivery of right information to right people in right formats at right times. Different information sharing applications have their distinctive requirements on the performance of sharing; They include but are not limited to accurate targeting (e.g., who are relevant or interested in specific information), reliability (e.g., how many of the
intended recipients receive the information), timeliness (e.g., how fast they receive the information) and efficiency (e.g., how many resources such as bandwidth, storage and energy are consumed). However, information sharing at societal scale are subject to massive dynamics, e.g., changing information consuming and sharing demand of people as a result of mobility and varying activities, dynamic bandwidth and congestion in underlying network infrastructures, and dynamic resources at end devices. This poses significant challenges to satisfy the requirements of applications. For example, when earthquake happens, accurate evacuation information needs to be disseminated to people impacted by the disaster timely and reliably. However, it is difficult to target people who need the campus map to evacuate a damaged region versus those who need instructions on first aid. Their needs may change over time. Deliver information to a wrong target wouldn’t help much. Moreover, massive failures in the communication network caused by the disaster make it harder to reach people in the disaster region. In this dissertation, we aim to address these challenges by exploiting the geographical and social network information of application users. We design adaptive middleware techniques to bridge a broad range of sharing applications with underlying communication environments that are heterogeneous and failure prone.

In Sec. 1.1, we present motivation behind our research. We state and lay out the scope of work with specific problems in Sec. 1.2. Finally, we highlight our contributions and the organization of the thesis in Sec. 1.3.

1.1 Motivation

The recent years have witnessed the success of commercial societal scale information sharing applications and systems. One such example is the raise of mass notification providing real time sharing of mission critical information such as emergency responses and critical weather warning. Key example systems include CityWatch[7], CodeRED[8], Everbridge[10], etc. A
report in 2011 [23] indicates that over 75 percent of cities in U.S. with more than 150,000 in population have adopted a mass notification system. In the wake of the Boston bombing manhunt and the Sandy Hook massacre, more governments, schools and other organizations are turning to mass notification systems to help protect public safety. Spending on mass notification systems in North America is expected to rise to 2.1 billion dollars in 2017, up from 1.6 billion in 2013 [26]. Another class of applications is that of personal information sharing, i.e., the sharing of personal contents generated by common users to the general public or among friends and family. Key examples include multimedia (e.g., video or photo) sharing such as YouTube and Flickr, file sharing such as 4Shared, social networks such as Twitter and Facebook, question and answer such as Quora and Answer.com. These applications are hugely popular and successful; reports show that as of January 2013, the active monthly users of Facebook has reached 1 billion, that of YouTube is over 800 million, and Twitter has over 200 million active monthly users. Moreover, in every single minute over 600 new videos are uploaded to YouTube, 6600 new photos are uploaded to Flickr, 100 questions are asked on Answer.com, 510000 comments, 79000 wall posts and 695000 status updates are generated on Facebook [24]. All the above statistics indicate that Internet users have exhibited an increasing demand of societal scale information sharing.

Some of the technological factors that lead to the dramatic increases in societal scale information sharing applications include the gradually mature web and mobile technologies. With the launch of Web 2.0, Internet users no longer passively retrieve information from websites. The technology enabled a participatory platform where users can contribute to the content available to others and exercise control over the data. Sharing of user-generated contents has received great popularity over the Internet. The proliferation of mobile platforms and increasing coverage and bandwidth of cellular and wireless networks contribute to the growth of the applications as well. The wide spread pervasiveness of the mobile infrastructure enables users to share information anytime, anywhere. Furthermore, high-end smartphones today equipped with a variety of specialized sensors e.g., light sensor, digital
compass, proximity sensor, gyroscope, accelerometer, GPS, and general purpose sensors like microphone and camera. These sensors enable users to collect information and data in a variety of modalities ready to be shared across a broad range of applications such as mobile social networks, smart transportation, healthcare and commerce. In the coming years, the ability to access and share knowledge and information anytime, anywhere at societal scale will be an essential mode of communications and interactions.

**Goal of this work:** Our goal is to develop solutions to support accurate, reliable, timely, and efficient societal scale information sharing. Accurate-targeting is the need of many applications to make sure only recipients that are relevant or interested in specific information receive it. Reliability and timeliness are desired for information to reach targeted recipients when it is useful and needed. Efficiency is desired to save resources (e.g., network, storage, energy) so that applications can accommodate more users or deliver more information under the same resource constraint. Depending on applications the objectives may be valued differently and may even need to trade off for one another. For example, emergency response applications place emphasis on reliability and timeliness due to the mission-critical nature of the information. On the other hand, for mobile applications energy efficiency is relatively more important because battery drains fast on smartphones when transmitting and receiving rich contents. In the thesis, we take into account various needs of applications.

**Challenges:** Supporting information sharing at societal scale is challenging for many reasons despite advances in Internet technologies. We highlight key aspects that present novel challenges and are relevant to designing societal scale information sharing systems to achieve the above performance objectives.

- **Users:** In societal scale information sharing applications, users are both consumers and producers of information. As information consumers, users may explicitly ask/subscribe for information (through questions/subscriptions) or they may implicitly need
the information (e.g., emergency alert broadcast). On the other hand, as information producers, users may produce and share information spontaneously (e.g., posts on social media), or as responses to queries or trade information for benefits (e.g., through question/answer or crowdsourcing). While users’ information consuming and sharing needs are based on context, e.g., location and activity, this context may be continuously changing. How do we design mechanisms to efficiently track users’ explicit and implicit dynamic information interests at societal scale? How do we identify users who are capable and willing to provide specific information when it is queried?

- Applications: In information sharing applications, information may be rich and multimodal (text, audio/video, images, albums, context sensing data). Information may be voluminous. Information needs may be instant (e.g., emergency alert) or delay-tolerant (e.g., Facebook updates). Utility of information can be based on its source and speed - whether it comes from a reputed source or friend and whether it comes in time for the user to take advantage. Moreover, applications have different performance requirements or metrics for information sharing and delivery: Some applications desire accurate targeting and customization, some may require high reliability and timeliness, and others may wish to trade part of performance for energy efficiency. Sharing information with satisfied performance requirements is critical. Furthermore, information sharing may be one-to-one (unicast), one-to-many (multicast), one-to-all (broadcast), many-to-one (e.g., data collection) and many-to-many (e.g., document sharing). Supporting these multiple modalities needs specialized techniques and optimization as well.

- Communication environments: Shared information is delivered through diverse communication infrastructure and environments, which consist of heterogeneous communication medium such as mobile/wireless/wired networks, social/human network etc, and end devices such as personal computers, smartphones, laptops, etc. All of them are subject to large scale dynamics. For example, communication through mobile and
wireless networks are inherently error-prone and unreliable, and subject to varying bandwidth caused by interference and multipath fading. In catastrophic disasters, network entities in all types of networks may experience large scale simultaneous failures. User mobility may cause disconnection between users and infrastructures (e.g., wireless access points or base stations) or cause change of access networks and IP addresses. It is challenging to ensure expected performance under such dynamic environments. Furthermore, resource limitation such as CPU, memory, storage capacity and battery at end devices affect decisions of designing and developing information sharing systems.

**Our approach:** We have observed that geography and social relationships can play an important role in societal scale information sharing applications.

- *Geography is correlated to consuming needs and provisioning capability of information:* People care information relevant to themselves and one commonly acknowledged measurement of relevance is geographical distance. People are always interested in information within their immediate vicinity (e.g., deals from local shops or traffic conditions on local highways). Similarly, for a regional event (e.g., a fair or a disaster such as earthquake), people located inside the event region typically have similar needs of information (e.g., finding shortest lines for rides or nearest shelters). On the other hand, when location based information is requested, people far away from the location can hardly be providers of the requested information.

- *Geography is correlated to communication environments:* Considering a group of people sitting in the same conference room, they all access Internet through the same WiFi access point on the wall. They are likely to subject to similar network congestion because of the interference at the access point. Moreover, colocated mobile users are likely to experience the same interference and bandwidth to the cell tower. In
catastrophic disasters (e.g., earthquake or tornado), networking entities in the disaster region tend to fail together.

- **Social relationship is correlated to consuming needs and provisioning capability of information**: Individuals are often interested in receiving information about others who are socially related to them (e.g., family members and friends). For example, Facebook users are interested in status updates from their Facebook friends than from strangers. During disasters, people interested in receiving the information which may affect their loved ones even if themselves are not physically located in the disaster region. On the other hand, people trust information from family and friends than from strangers.

The geo-social knowledge can be exploited to assist decision makings in societal scale information sharing applications to improve their performance goals. Recent technology advances in positioning and online social networks have made the collection of the geo-social information more tangible today. Firstly, several positioning techniques have become widely available to help Location Based Services (LBS) collect users’ location information. Key examples include GPS, WiFi fingerprinting, cell tower triangulation, and IP based positioning. Nowadays GPS is commonly deployed in a variety of portable devices such as smartphones and tablets. It provides accurate fine grained location information. WiFi and cell based positioning are considered as alternative solutions when GPS fails short in urban areas. They are less accurate but have good coverage. IP based positioning is least accurate and subject to IP spoofing. However, it can be used in combination with other techniques to improve reliability or used when all other techniques are not available.

Secondly, with the popularity of online social networks and social media such as Facebook and Twitter, getting one’s social network information have become tangible. The boundary between online social network and offline social network has blurred over the years. Many studies [145, 147] have confirmed that emerging adults use social networking sites to connect with people from their offline lives, such as their friends and families, confirming that there
was an overlap between people’s offline and online social networks. In terms of data accessibility, most major online social networks provide public APIs to enable authorized entities to obtain users’ social network graph and online social activities. Moreover, several works have studied how to effectively calculate the social closeness between users based on the social network graph and their interactions which can be easily extracted from their online social activities [107, 105].

In this dissertation, we design middleware based solutions to utilize geo-social information to tackle the challenges shown at various aspects of societal scale information sharing. Rather than working at the communication/network layer or the application layer as in most existing approaches, we design middleware that is positioned between the underlying network infrastructures and the information sharing applications to accommodate the dynamically changing network environments as well as the varying application needs. We illustrate the middleware in Figure 1.1.

![Middleware for societal scale information sharing](image)

**Figure 1.1:** Middleware for societal scale information sharing

The benefit of middleware based solutions are two-folds. First, middleware is loosely coupled with the downward network environments and the upward applications, and agnostic to the heterogeneity of hardware specifics and operating systems. Although we differentiate
solutions for wired and wireless environments due to their drastically different characteristics, the components of middleware for each environment are developed as general solutions that do not rely on any specific networking protocols and do not touch upon the lower layers of the network stack. Our middleware solutions can transform to software packages or libraries in easily portable languages, e.g., Java and Python, that are independent of operating systems. They can be installed and deployed easily and allows a variety of applications to invoke the provided services. Second, middleware is able to provide adaptive services and easily extensible to accommodate new requirements from applications. We developed our solutions with a variety of information sharing applications in mind. We classified applications based on their requirements and developed tailored services for each category of them. For new types of applications and requirements that have not been considered, middleware can always be extended by adding new components to meet their needs and reuse existing components when possible.

1.2 Thesis Problem and Scope

We classify societal scale information sharing and scope our work along 2 dimensions. Figure 1.2 presents the classification and the thesis problems.

Along the dimension of system layers, we focus on two major problems at two layers: 1) the information layer - what to share: determine specificity of contents and accurately target information consumers and providers. 2) the dissemination layer - how to share: determine dissemination mechanisms to deliver information from its source to targeted receivers to meet the performance goals (e.g., efficiency, reliability and timeliness).

the Information Layer Problem: at the information layer, we focus on the information flow between users in various societal scale information sharing applications. Specifically we
are interested in mechanisms to efficiently track users’ information consuming and sharing demand, and accurately target information consumers and providers of given information. When a piece of information becomes available (e.g., shared or published by an information provider to the application) how can we target potential consumers that are interested in receiving it? On the other hand, when there are needs of information from users (e.g., queried or asked for by an information consumer) how can we target the potential producers that are capable and willing to provide the information? We characterize a generic setting as follows.

Players who participate include:

- A group of consumers \((C)\) who wish to receive information. Their information needs are dynamically changing over time and space.

- A group of providers \((P)\) who wish to provide information. This could be voluntary or benefit driven. Their information provisioning capabilities are dynamically changing over time and space.

---

**Figure 1.2: Thesis problems and scope in quadrant presentation.**

<table>
<thead>
<tr>
<th>Information Layer</th>
<th>Societal scale instant information sharing</th>
<th>Societal scale delay-tolerant information sharing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficient and accurate targeting of consumers of information for instant information notification</td>
<td>Efficient and accurate targeting of providers of information for query-driven information sharing</td>
<td></td>
</tr>
<tr>
<td>Dissemination Layer</td>
<td>Reliable and timely information dissemination in failure prone network environments</td>
<td>Efficient dissemination of delay-tolerant information in dynamic wireless and mobile environments</td>
</tr>
</tbody>
</table>
We study the problem how to target and match $C$ and $P$ at real time considering the following performance metrics:

- **Efficiency**: the efficiency are two folds: 1) the matching process itself is efficient (e.g., it involves few control and computation overhead); 2) the resulting information flow from the matched providers to consumers is efficient (e.g., the information dissemination involves few message overhead or energy cost)

- **Quality**: the quality of matching is reflected as that the consumers meet their needs of information and the providers meet their needs from sharing.

- **Scalability**: the sets of $C$ and $P$ could be large in number and distributed over a large geographical space.

**the Dissemination Layer Problem**: at the dissemination layer, we focus on the delivery of information from information source to targeted receivers through diverse communication environments that are characterized by different networks and device types. The network types we explicitly considered in the thesis include wired, mobile and WiFi networks, and the accessing devices could be personal computers and smartphones. We characterize a generic setting as follows.

Players who participate include:

- A source or sources ($S$) which have contents to deliver.
- One or a group of destinations ($D$) who wish to receive contents from source $S$.

We study the problem how to deliver contents from $S$ to $D$ considering the following performance metrics:
• **Efficiency**: depending on the concerned communication environment, the efficiency may include message overhead efficiency and energy efficiency.

• **Reliability**: users in $D$ can reliably receive contents.

• **Timeliness**: users in $D$ can receive contents as quickly as possible.

• **Scalability**: contents from $S$ are delivered to $D$ which could be large in number and distributed over a large geographical space.

On the other hand, along the dimension of timing constraints we consider two classes of applications: 1) instant sharing and 2) delay-tolerant sharing.

**Instant information sharing**: In the class of instant information sharing applications, once information is shared or published, it needs to be immediately disseminated to users who are interested in it. Key example applications include emergency responses, weather alerting, etc. Typically the information to share is small. Due to its mission critical nature, reliable and timely delivery is most desired. We observe the following problems that need to be solved but haven’t been sufficiently addressed by existing work:

• at the information layer, how to efficiently and accurately target consumers of information on the fly for instant information notification applications?

• at the dissemination layer, how to provide fast and reliable information dissemination in failure prone network environments?

**Delay-tolerant information sharing**: In the class of delay-tolerant information sharing applications, information needs may be tolerant to delays, represented by a number of user-generated content sharing applications. Key examples include information sharing on online social media, e.g., Facebook, and question answer applications, e.g., Answer.com. The in-
formation or contents to share are tend to be rich. Therefore, sharing efficiency are usually concerned. Again, we observe the problems that need to be solved including:

• at the information layer, how to efficiently and accurately target potential providers who can give trustworthy and timely information in query-driven information sharing applications?

• at the dissemination layer, how to provide efficient dissemination of information in dynamic wireless and mobile environments where mobile users experience intermittent network availability and varying communication costs?

Besides the objectives we mentioned earlier, there are many other issues related to societal scale information sharing that need to be properly addressed in real service deployments. For example, data encryption and security issues are non-negligible is all types of applications. However, they are out of the scope of this thesis.

1.3 Thesis Contributions and Organization

In this dissertation, we systematically study the issue of societal scale information sharing. We first identify challenges, pinpoint the limitations of existing work, and propose a general approach to address the challenges through middleware that takes into account the knowledge of geographical and social network information from application users.

We developed a logical framework for categorizing information sharing along system layers and time constraint dimensions. Figure 1.3 presents our contributions in each quadrant of the framework.

• For the class of instant information sharing applications, at the information layer, we study mechanisms to efficiently track users’ dynamic information interests and notify
relevant users when information is available. We conjecture that a topic-based publish/subscribe system can form the basis of an efficient architecture for this notification service. We design a dynamic pub/sub broker middleware that constructs pub/sub brokers into a structured overlay and provide mechanisms to moderately reposition pub/sub users on brokers and reposition brokers on the overlay to adapt to users’ dynamic subscriptions. We implement our solution and conduct extensive experiments showing that under highly dynamic subscriptions our solution can still maintain an efficient notification structure that provides 30% less delay and overhead, and a reconfiguration cost reduction of 80% as compared to other state-of-the-art pub/sub architecture.

- At the dissemination layer of instant information sharing, we study reliable and timely event dissemination in extreme situations (e.g., disasters). The challenge is geographically correlated regional failures caused by catastrophic disasters hinder the ability to reach recipients inside the corresponding failed region. We design a reliable geo-social notification middleware forming a P2P overlay that is aware of (a) the geographies in

Figure 1.3: Thesis contributions in each quadrant of the societal scale information sharing framework.
which the message needs to be disseminated and (b) the social network characteristics
of the intended recipient, in order to maximize/increase the coverage and reliability.
It combines efficient location-based message delivery and opportunistic social diffusion
through out-of-band channels to achieve high reliability of information dissemination.
We implement the middleware and conduct extensive evaluations showing that our
solution reaches up to 99.9% of desired recipients even under massive regional failures
of infrastructures.

- For the class of delay-tolerant information sharing applications, at the information
  layer, we study the mobile question and answer paradigm where information consumers
  explicitly request for information by submitting queries. Potential providers by taking
  advantage of their smartphones have the flexibility to move, collect information and
  response with the requested information anytime anyplace. We design a broker middle-
  ware aims to provide mobile information seekers with timely, trustworthy and accurate
  answers while ensuring that information providers are not inappropriately burdened,
  by taking advantage of both static and dynamic context and semantics from mobile
  users (e.g., geolocation, social network connections, expertise and interests, profiling of
device sensors, and device battery level). We develop a prototype system consisting of
an Android client app and a Q&A broker. We conduct extensive experiments with real
world datasets. The results indicate that our solution is efficient and provides superior
worker selection. Furthermore, our middleware is general, scalable and flexible to be
leveraged by multiple mobile Q&A systems.

- At the dissemination layer of delay-tolerant information sharing, we study the problem
  of efficient dissemination of social media contents to mobile users to enable them to
  access to social media anytime anyplace without requiring to be online all the time.
We design middleware solutions that help mobile devices selectively prefetch or downloads contents that have high likelihood of being viewed, using knowledge of users
social network and current network/system conditions. We develop an Android app providing offline access to Facebook. We use data traces gathered from our app to drive extensive evaluations which show that our proposed solution exhibits superior viewing performance and is energy efficiency.

The dissertation is organized as follows. In Chapter 2, we present limitation of existing work on information sharing at societal scale. In Chapter 3, we study efficient publish/subscribe for instant information sharing applications. Chapter 4 is dedicated to present our research on providing reliable and fast dissemination for instant information sharing applications. In Chapter 5, we present our research on providing efficient mobile crowdsourcing and question and answer. Chapter 6 addresses the problem of efficient dissemination of social media contents to mobile users. Finally, in Chapter 7, we conclude the thesis and discuss some interesting future research directions.
Chapter 2

Related Work

In this thesis, we focus on two classes of societal scale information sharing applications: 1) instant information sharing highlighted by mission critical notification applications and 2) delay-tolerant information sharing exampled by user-generated contents sharing. Now we review both classes in their commercial systems and research work.

**Instant information sharing:** Commercial Mass Notification Systems (MNSs) that aim to provide reliable instant notification of mission critical information, e.g., emergency alerts, have emerged. Leading examples are CityWatch[7], CodeRED[8], Everbridge[10], Blackboard Connect[4], and etc. Common features provided by the services include: 1) custom pre-set scenario message templates allowing the construction of semi-customized messages in a few clicks; 2) multi-channel notification to reach recipients via websites, email, SMS, social media such as Facebook and Twitter; 3) flexible recipients targeting through organization internal portal, group subscriptions and GIS mapping; 4) Comprehensive and robust reporting and analytics capabilities. Besides MNSs, in the mobile application market, a large number of mobile notification apps for specific needs such as weather alert (e.g., MyWarn[17]) and traffic alert (e.g., Beat the Traffic[3]) are populated. They take user subscriptions and
utilize mobile push notifications to provide real time customized notifications to smartphones. The above systems provide services through Web portals or notification servers that are built on centralized implementations. Although many MNSs provide multi-channel message dissemination, the source of the messages for different channels is common, as the centralized in-network server that creates the message and triggers the notification. Therefore, they have limited reliability under extreme situations (e.g., network failures and network partitions).

Timely and reliable information sharing or flash dissemination has been gathering extensive attention in the research community. Publish/subscribe is a popular research paradigm for instant notification services. Pub/Sub systems are either content-based where subscribers receive notifications when the content of the message matches their interest [47, 133, 125, 27] or topic-based [48, 168, 35, 55] when the “topic” of the publication matches their interest. Today the large majority of publish-subscribe systems designed to operate in large-sale, wired scenarios adopt a dispatching network composed of a set of brokers or peers, which cooperate to route messages from publishers to subscribers. A focus of content-based pub/sub research is subscription management techniques such as subscription matching, covering, subsumption and merging [102, 151, 149, 69, 40, 86]. Furthermore, in both content-based and topic-based pub/sub research, extensive efforts have been done to study optimal overlay design [54, 129, 83, 123] for efficient notifications. Interested readers can refer to [36] for a survey. Relatively few efforts have studied notification reliability in pub/sub. Esposito et al. [68] studied mission critical data dissemination via publish/subscribe. The work proposed a cluster-based P2P structure for pub/sub and replicated cluster coordinators and introduced multiple-tree redundancy to tolerate to unpredictable failures.

Several work has also explored P2P and overlay based dissemination techniques such as gossip P2P and Application Layer Multicast (ALM) to provide fast information dissemination resilient to network failures. Deshpande et al. [64, 63] investigated instant dissemination of rich contents in highly heterogeneous and failure prone environments using P2P. They designed
a peer-based approach along with a gossip protocol to maximize the speed of dissemination and handle both content and network heterogeneity and be resilient to transmission failures. Kim et al. [87] considered flash dissemination of small messages from a single source to a large number of targeted receivers using ALM. To achieve reliable dissemination the work exploited path diversity by using a forest based structure rather than traditional tree based ALM. Also it designed multidirectional multicast algorithms that nodes in addition to communicating the data to children also selectively disseminating the data to parents and siblings to achieve enhanced reliability. The above work focuses on dealing with random failures in networks but didn’t consider the impact of the geographically correlated failures that could happen under catastrophic disasters. In [92] Kim et al. has studied the impact of geographically correlated failures on overlay based data dissemination. They propose new overlay network construction methods that incorporate proximity-aware neighbor selection methods to improve the resilience of the overlay network to geographically correlated failures.

Several research work considered instant information sharing in non-conventional networks. Xing et al. [157] studied reliable and fast broadcast of mission-critical data over ad hoc networks. The work considered the dissemination in two stages: the dissemination of metadata and actual data. The dissemination of metadata is to allow receivers be aware of the upcoming dissemination, and the dissemination of the actual data is to deliver actual information to dissemination-aware receivers. The authors designed protocols for each stage and as an integration the overall protocol is able to offer adaptive reliability. Similarly, probabilistically reliable broadcast protocols to enhance delivery ratio by increasing redundancy [139] or employing acknowledgements [113] have been proposed in MANET and Ad hoc networks.

Although research work has studied instant sharing of mission critical information, most attention has been put on handling the unpredictable dynamics (i.e, failures) in networks but little has at the same time taken into account the dynamics of information receivers and/or information providers. This dynamics should not be ignored in societal scale information
sharing applications. In combine with the dynamics in networks, it can make fast and reliable information sharing extremely hard. Furthermore, with the increasing visibility of social network knowledge from common users, new opportunities emerge to improve reliability of dissemination under severe failures (e.g., large scale geographically correlated failures).

**User-generated content sharing:** Societal scale sharing of user-generated contents and information has gain great popularity and the number of commercial applications has been skyrocketing over years. Key examples include multimedia (e.g., video and photo) sharing such as YouTube and Flickr, file sharing such as 4Shared.com, social media such as Facebook and Twitter, question and answer such as Quora and Answer.com. Almost all the above systems are realized as centralized web portals or web services. Dissemination efficiency and specificity of communication networks and devices are rarely considered in their design: they build the applications and leave it to network providers and ISPs to take care of the data delivery between the application servers and users’ devices. However, ISPs are application agnostic and the heterogeneous performance requirements of various applications are hardly guaranteed.

Extensive research efforts have been spent to facilitate the sharing of user-generated contents. A survey of the key solutions is presented in [132]. Two broad classes of solutions include Content Delivery Networks (CDNs) and Peer-to-Peer Networks. CDNs aim at addressing the ”flash crowds” (i.e., sudden requests peaks) problem for content publishers to make their content widely available. Main solution involves replicating content on different sites and possibly on different ISP networks, and redirect content requests to one of the replicas. CDNs do not directly involve the delivery of information to end-users. Therefore, they are orthogonal to the research considered in this thesis.

The research on P2P networks have covered a broad range of aspects from content search, replication, and delivery to user mobility, trust and cooperation. With respect to information delivery, contributions have been made to design structured, unstructured and hierarchical
overlay networks, and P2P dissemination protocols for different overlays such as P2P multicast or Application-Layer Multicast (ALM), mesh and gossiping. Interest readers is referred to [114, 30] for extensive surveys on P2P overlay networks. Recent research on P2P systems start exploiting social phenomena such as friendship and communities of users with similar tastes to increase application performance. For example, Pouwelse et al.[135] presented a social-based P2P file-sharing paradigm by maintaining social networks and using them in content discovery, content recommendation and downloading. The system is based on BitTorrent[5] and it adds social-based functionality and imports existing user contacts from other social networks, e.g., MSN. Rather than using direct content-based searching, the system performs content discovery and recommendation from users have similar interests. Cheng et al.[51] proposed a P2P assisted delivering framework that explores the clustering in social networks for short video sharing. The system builds a bi-layer overlay where at the lower layer peers sharing the same interest form swarms, and at the upper layer peer builds neighborhood relations among all swarms that contain the peer. The system incorporates an efficient indexing scheme and a prefetching strategy benefiting from the embedded social networks. Li et al.[108] explored social relationship, interest similarity, and physical location between peers in OSNs in P2P assisted video sharing. Similar to [51], the authors proposed social network based P2P overlay construction algorithm and prefetching algorithm to improve the user experience and scalability of the Video-On-Demand (VOD) services.

Diverse network environments have been studied for sharing user-generated contents as well. Kelenyi and Nurminen[90] studied the challenge of applying BitTorrent file-sharing application over mobile devices due to high energy consumption by P2P data transferring. It proposed using cloud service to download contents transparently and transfer them to mobile device in an energy efficient way. [165, 98] designed scheme and protocol to facilitate content sharing in vehicular ad hoc networks (VANET) using P2P technologies. Zhang et al.[165] designed P2P content sharing using popularity-aware schemes to ensure popular data is more likely to be shared with others so that the overall query delay and hit ratio is
improved. Lee et al.[98] proposed a network coding based le swarming protocol for VANET P2P to deal with typical mobile network issues such as dynamic topology and intermittent connectivity. Leontiadis et al.[99] considered the environment of hybrid (i.e., partially structureless) vehicular network and designed protocol for persistent content sharing that contents “stick” to areas where passing vehicles need to receive them. [82, 158, 130, 119] studied content sharing in mobile social networks and delay tolerant networks. Ioannidis et al.[82] mathematically evaluated the optimality and scalability of distribution of dynamic content over a mobile social network. Mashhadi et al.[119] designed mechanisms to leverage information about nodes colocation and their social network to maximizes both precision (i.e., nodes receive only content they are interested in) and recall (i.e., all relevant content is received by interested nodes) of content sharing in DTNs. [158, 130] looked at specific spatial sharing applications in DTNs and designed protocols to optimize the dissemination. Xing et al.[158] looked at a similar spatial dissemination problem as in [99] where contents “stick” to areas and received by users in the areas. However, it considers DTNs without infrastructures. It aims to maximize delivery reliability without incurring significant storage/transmission overhead. Ott et al.[130] generalized the spatial dissemination problem of [158, 99] and referred as “float content” service, and explored the parameter space of oating content systems to identify under which conditions such systems are feasible through rigorous mathematical analysis and extensive simulations.

Despite of the research contributions, non-conventional networks such as VANET and mobile DTNs are not likely to be widely available to support societal scale information sharing in the near future. For content sharing over conventional networks (e.g., wired, cellular and WiFi), applications can be supported using P2P technologies are limited. The limitation of P2P based approach is that P2P applications implemented at the upper level are totally customized with respect to the particular overlay network they are designed for, or even for its particular implementation. For example, search techniques designed for a structured overlay are completely useless in unstructured networks and vice versa. Popular stream-
ing applications such as PPLive[18] implement their own delivery structures and overlays. P2P services are typically created to answer a very concrete need of a community of users, and are designed to provide a “quick&dirty” solution to these need [132]. Skype[38] and BitTorrent[5] are clear examples of this approach. Efficient information sharing in other non-P2P applications are not sufficiently studied.
Chapter 3

Instant Information Sharing: Information Layer

In this chapter, we design efficient mechanisms to track users’ dynamic interests of information at societal scale for the class of instant information notification applications. Figure 3.1 highlights the contributions of this chapter in the design framework for societal scale information sharing. Societal scale information notification applications calls for a solution that is able to efficiently support changing information interests for a very large number of users. We conjecture that a topic-based publish/subscribe system can form the basis of an efficient architecture for this notification service. We propose DYNATOPS, a dynamic pub/sub broker middleware that provides efficient scalable societal scale event notifications for dynamic subscriptions via distributed broker networks.

In DYNATOPS, pub/sub users are moderately repositioned on brokers and brokers are moderately repositioned on the overlay structure for efficient information notification, to adapt to the publication and subscription dynamics. First, it uses a similarity-based user-broker mapping to map DYNATOPS users to nearby brokers. Under fast changing subscriptions such as
Figure 3.1: Contributions of Chapter 3 in the societal scale information sharing framework.

those caused by changing locations of users, conventional user-broker mapping policies show a dramatic increase in their subscription management overhead. The proposed technique efficiently groups users sharing similar interests to be managed by the same set of brokers, to effectively alleviate the overhead with regard to subscription updates among brokers. Our experimental results indicate over 40% reduction in brokers’ subscription management overhead as compared to other state of the art techniques under highly dynamic subscriptions. Second, more importantly, to provide efficient event notifications, DYNATOPS incorporates a broker reconfiguration process that is both subscription and structure-aware. Unlike existing topic-based pub/sub systems where topology changes are triggered in a self-organizing manner to reflect subscription changes, we propose a cost-driven reconfiguration process that changes topology in a planned manner. The key intuition behind our approach is that reconfiguration of the broker network is achieved by merely moving broker nodes to positions in an overlay structure where routing is already highly optimized - this reduces the effort for reconfiguration in a large scale system. Furthermore, we trigger reconfigurations only when they are likely to have high utility. We implemented our solution and with
extensive experiments we observe that under highly dynamic subscriptions our solution can still maintain an efficient notification structure that provides 30% less delay and overhead, and a reconfiguration cost reduction of 80% as compared to other state-of-the-art pub/sub architecture.

The rest of the chapter is organized as follows. Section 3.1 present the background and motivation of the work. Section 3.2 provides an overview of the system. We present the DYNATOPS solution in detail in Section 3.3 and Section 3.4. We evaluate the solution through extensive simulations and experiments in Section 3.5. The chapter is concluded in Section 3.6.

3.1 Introduction

The recent years have witnessed the growth of societal scale notification systems that have penetrated multiple aspects of our day-to-day life. Key examples include mobile social networks (e.g. Twitter and Foursquare), geomarketing (e.g. shopalerts.att.com), traffic and weather alerts (e.g. www.wunderground.com and weather.gov), emergency response (e.g. www.gdacs.org/alerts and www.mywarn.com), etc. These systems are large (e.g. Twitter is estimated to reach 500 million users in 2012 and currently handles over 340 million tweets daily), geographically distributed and largely subscription based. The notion of subscriptions in such systems is often simple, and instantiated by an average citizen – deals for local shops, traffic alerts for freeways, weather conditions for zipcodes, and location check-ins from friends. Moreover, we observe an emerging trend that the interests of such users, i.e. subscriptions, are short lived (duration of minutes or hours) and change dynamically based on users’ locations or contexts.

For instance, mobile subscribers are often interested in events and information within their
immediate vicinity; the dynamically changing location is a key aspect of what constitutes a subscription and consequently a relevant notification in this case. In disasters (e.g. earthquakes), people’s information needs (i.e. subscriptions) change with the evolution of the disaster event (e.g. pre-disaster or post-disaster). An individual’s region of interest, i.e. subscription, changes as he/she moves into or out of disaster region; people are anxious to check the safety of their beloved ones and the changing locations or statuses of families or friends cause subscription changes as well. Such shifts in notification needs calls for a system that is able to efficiently support simple, yet changing subscriptions for a very large number of users.

Publish/subscribe has for long been a popular communication paradigm to provide customized notifications to users in a distributed environment due to its loose coupling between the information providers (i.e. publishers) and consumers (i.e. subscribers). Pub/Sub systems are either content-based where subscribers receive notifications when the content of the message matches their interest [47, 133, 125, 27] or topic-based [48, 168, 35, 55] when the “topic” of the publication matches their interest.

One of the main incentives for content-based pub/sub is to enable delivery of relevant and meaningful notifications through rich and expressive subscription languages. Here, sophisticated subscription management strategies are usually required to determine matching subscribers and subsequently route only the relevant events; runtime overheads for subscription insertion/removal and event matching are much higher than topic-based implementations [46, 131, 85]. Over the last decade, significant progress has been made in the design of pub/sub systems to address scalability and efficiency of content-based pub/sub [85, 44, 103, 45, 84], largely motivated by the needs of enterprise systems. The design of pub/sub systems taking into account frequently changing subscriptions is beginning to receive interest, especially in the context of content-based pub/sub systems where the emphasis is on designing novel subscription languages and schemas. For example, [85] proposed
a parametric subscription language in content-based pub/sub systems to reduce broker runtime complexity in the event of a subscription update. The key idea is to insert variables into subscriptions which may be updated frequently so that only variable update is performed during subscription update. Techniques to manage the overheads due to event matching and dynamic subscription updates are however still a concern.

We conjecture that a topic-based pub/sub system can form the basis of an efficient architecture to deal with the simple, yet changing notification needs of a large number of users; for example by routing events through group multicast to peers that match subscription topics. Subscription insertion/removal and event routing on a broker can be easily done in $O(1)$ time. Indeed, topic-based pub/sub systems have been widely deployed in contexts where events divide naturally into groups (i.e. “topics”) and efficient information notification is demanded [112, 59]. More recent topic-based pub sub systems, e.g. Twitter, have scaled to a very large number of users.

Recent works on topic-based pub/sub usually builds structured or unstructured p2p overlay networks for event notifications. The construction of the overlay topology can be either subscription-dependent [76, 143, 55, 35, 121, 54] (i.e. the overlay topology is dependent on peer subscriptions) or subscription-independent [48, 168]. Subscription-dependent techniques are preferred by recent studies because they yield superior event notification efficiency through optimized overlay topology configuration. Therefore, the research emphasis of topic-based pub/sub has been focusing on constructing churn-resistant subscription-dependent overlays with low node degrees and high notification efficiency. However, the impact of dynamic subscriptions has been overlooked in the design of these systems. We show in Section 3.5 that frequent subscription changes can introduce high overhead and frequent topology reconfigurations if they are not properly taken into account in the design. They potentially bring great difficulty in troubleshooting and management of the pub/sub system.
3.1.1 Related Work

Conventional pub/sub systems assume that clients join the broker environment in one of the following ways: (a) connecting to any broker with no restrictions [89, 133], or (b) connecting to the closest broker [101, 81, 71]. [93] is the first to investigate client placements that optimize delivery delay and system load. However, the work focused on the placement of a few known publishers under fixed subscribers’ subscriptions and broker overlay topologies. Such schemes are inefficient in societal scale notification applications where (a) publishers are many (in general the entire user set) and (b) subscriptions change frequently based on user interests. In this work, we explore a similarity-based user placement that takes into account both the dynamics of user subscriptions and broker load. The proposed technique can effectively reduce subscription management overhead and improve notification efficiency under dynamic subscriptions of users.

Building and reconfiguring overlay networks that take into account nodes’ subscriptions for dissemination efficiency has been explored in both content-based and topic-based pub/sub systems. In content-based paradigm, Sub-2-Sub [153] is a content-based protocol that clusters nodes according to their subscriptions to construct a ring for each attribute. [34] proposed a self-organizing algorithm to cluster brokers that supposedly will be target for the same events in the near future. [162] discussed primitives that reconfiguration protocols need to implement to ensure high availability with minimum disruption under topology changes. In topic-based paradigm, many recent topic-based pub/sub systems [76, 143, 55, 35, 121, 54] build and maintain their overlays based on brokers’ subscriptions. For rendezvous-based pub/sub systems that maintain topic-routing trees, Magnet [76] clusters nodes with similar subscriptions on a skewed DHT, and explores a customized routing to reduce the number of relay nodes in the multicast trees. When a node’s subscription changes, the node needs to rejoin the DHT to be placed onto a new position based on its new subscription. On the other hand, [143, 55, 35, 121, 54] build relay-free overlays without topic rendezvous and
explore gossip-based dissemination and/or in-cluster flooding to disseminate event notifications. Tera and StAN [35, 121] creates dedicated topic overlays for each topic. A node joins overlays for the topics that it subscribes to by connecting to a node already in the overlay. SpiderCast and TCO [55, 54] build a single unstructured overlay that strives to maximize clustering of nodes according to their interest in topics. The focus of system is to keep low node degrees while maintaining the topic-connected property as the foundation of the relay-free routing. This is achieved by the neighbor maintenance routine in SpiderCast and the overlay construction algorithm in TCO, both of which can trigger an overlay topology reconfiguration when a node’s subscription changes. PolderCast [143] maintains a ring structure for each topic and combines deterministic dissemination over a ring with probabilistic dissemination similar to gossiping. Its overlay management mechanism also triggers topology updates whenever node churns or subscription changes.

While all the above systems trigger topology changes to reflect subscription changes in a self-organizing manner, in this work we design DYNATOPS with a different philosophy. We argue that in societal scale notification applications where subscriptions are short lived and frequent change is the norm, the topology reconfiguration should be incorporated in a more systematic and planned manner in the system design. This is to reduce the reconfiguration overhead from frequent subscription changes that we already know will happen. This is achieved through built-in mechanisms that detect when sufficient changes have occurred and that deal with those changes through planned overlay reconfigurations. We design a cost-driven reconfiguration process that detects when sufficient changes have occurred and deals with those changes through planned overlay reconfigurations that are likely to have high utility.
3.2 An Overview of DYNATOPS

DYNATOPS is a topic-based publish/subscribe architecture designed to offer an efficient event notification service for societal scale applications where dynamic subscriptions is the norm.

3.2.1 Architecture

Figure 3.2: DYNATOPS architecture.

Figure 3.2 shows the DYNATOPS architecture where pub/sub users connect to a distributed broker network for topic subscriptions and event publications. DYNATOPS brokers are organized into a structured overlay based on a DHT structure (e.g., Chord Ring [146]); Brokers are configured onto the overlay as overlay nodes with unique nodeIDs. DYNATOPS overlay handles event dissemination through a Rendezvous-based pub/sub protocol similar to those of Scribe [48] and Bayeux [168].

We illustrate the basics of Rendezvous-based pub/sub for a group of topics via a single dissemination overlay in Figure 3.3. The overlay maintains independent topic routing trees of different topics: each topic has a Rendezvous Point (RP), which is an overlay node responsible...
for the hashed key of the topic name in the DHT, and the RP serves as the root of the tree. The tree spans all brokers that subscribe to the topic and it is created in a top-down fashion in which the routing path from the root to other nodes is determined by the key-based routing. During event dissemination, the published events are first forwarded to the RP, and then the RP disseminates them along the tree. Note that, the tree may contain relay nodes that have not subscribed to the topic, called unrelated relays, because they are in the routing path between the RP and subscriber nodes. The existence of unrelated relays introduce unnecessary delay and message overhead for event notification. Next, we describe how DYNATOPS reduces unrelated relays through dynamic mappings.

### 3.2.2 Dynamic Mappings

The efficiency of the pub/sub system is determined by two mappings: 1) mapping of pub/sub users to their home brokers; and 2) mapping of brokers to overlay nodes of dissemination overlays. The mapping of users to brokers have a direct impact on the scale and dynamics of subscriptions at each broker. DYNATOPS handles this mapping dynamically through
a user placement algorithm. The aim of the user placement algorithm is to reduce brokers’ subscription changes so as to reduce subscription management overhead and alleviate the demand for broker network reconfiguration when the pub/sub users’ subscriptions are dynamic. This is done by dynamically aggregating users with similar subscriptions to be managed together by the same set of brokers.

One the other hand, the mapping of brokers to overlay nodes have a direct impact on the routing of topic subscriptions/publications in the broker network. DYNATOPS handles the mappings through a Structure and Subscription aware Reconfiguration (SSR) algorithm running on a Reconfiguration Manager. The reconfiguration manager is a logically centralized entity. It periodically collects statistics of the pub/sub environment, i.e. rates of event publications and snapshot of topic subscriptions at brokers. The reconfiguration algorithm aims to maintain efficient pub/sub routings based on updated pub/sub environment. In DYNATOPS, We explore the possibility of dynamically altering the brokers’ logical placement on the overlay to match the underlying (possibly changing) subscription needs. Figure 3.4 shows an example illustrating how DYNATOPS is able to optimize topic routing through proper reconfiguration of the overlay. We can see that before reconfiguration, event notification from RP to subscribing brokers in topics $t_1$ and $t_2$ both take maximum of 2 hops, and 2 messages and 3 messages respectively. After reconfiguration by swapping the positions of broker A and B, and swapping broker C and broker D, it results in an optimized routing for $t_1$ and $t_2$. Each topic takes only 1 hop and 1 message to disseminate an event.

### 3.3 DYNATOPS User Broker Mapping

The core of our user placement strategy is a distributed grouping algorithm that determines the broker to host a user. The goal of the algorithm is to ensure that users with similar subscriptions are grouped and managed by the same set of brokers.
Figure 3.4: Example of the structure and subscription aware reconfiguration.

The grouping of similar users brings two benefits to a pub/sub system. First, it reduces the subscription changes of brokers so as to minimize the subscription management overhead inside the broker network, which incurs when a broker joins/leaves a topic routing tree. The intuition here is that by clustering similar users, each broker tends to only subscribe to topics that all/most of its hosted users interest in, so that the subscription states of the brokers are more stable and less sensitive to a single user’s subscription changes. Second, it potentially reduce the number of brokers that require to subscribe to each single topic. In topic-based pub/sub implementations, the number of brokers subscribing to a topic directly impact the notification performance of that topic. More efficient topic routing trees with less hops and unrelated relays can be build when fewer nodes are in the trees.

The algorithm requires each broker maintains the *states* of other brokers in the system. The state includes the subscriptions and loads (explained later) of other brokers which could be partial and randomized. Such state can be readily implemented by distributed membership
protocols as those in most existing systems [76, 143, 55]. We show in our experimental evaluation that it is sufficient for each broker maintain states of a small number of other brokers (e.g., 5%) to outperform other user placement techniques.

Let $P$ denote the set of topics, we evaluate the similarity of random user $u$ to broker $b$ in their subscriptions $s_u, S_b \subseteq P$ by a user-broker similarity metric, $Sim_{b,u}$, which is defined as the normalized size of their intersection, in the range $[0, 1]$. Formally, $Sim_{b,u} = |S_b \cap s_v|/|s_v|$.

Our scheme is flexible and can accommodate other similarity metrics (e.g. Jaccard similarity), however, their analysis is out of the scope of the work. Similarity based user placement will select for each user $u$ a hosting broker that yields the highest user-broker similarity among all brokers. However, user placement based only on similarity metrics may result in severe load imbalance at brokers, especially when users’ subscriptions are highly skewed. Hence, to alleviate load imbalance, we also take into account a load metric, $L_b$, to reflect the load level of a broker, with range $[0, 1]$. In this work we assume homogeneity of brokers and define broker load as the normalized amount of local user subscriptions managed by a broker. Formally, $L_b = \frac{\sum_{u \in U_b} |s_u|}{\sum_{u \in U} |s_u|}$. (Heterogeneous broker capacities can be easily accommodated by introducing a capacity-based weight in the formulation) A user-broker utility combining the similarity and load metrics is defined as $Util_{u,b} = Sim_{b,u} - w_l \cdot L_b$, where $w_l$ is the relative weight for the load metrics.

The crux of user placement is the user join protocol which is executed every time a user subscribes to a new topic, or unsubscribe an existing topic:

1. the user sends a join request message containing its subscriptions to a random broker during bootstrap.

2. the broker upon receiving the join request, calculates the utility $Util_{u,b}$ for all its visible brokers, and determines the best broker for the user.

3. if the best broker is not the broker, it replies to the user with the redirect message
containing the address of the best broker. Otherwise, the broker accepts the join request by sending back a join reply message.

4. if the user receives a redirect message, it sends a join request message, with REDIRECT bit set, containing its subscriptions to the broker indicated by the redirect message. The broker accepts the request by sending back a join reply message.

### 3.4 DYNATOPS reconfiguration

DYNATOPS aims to reconfigure the broker network to minimize event notification overhead and delays. We provide a formal modeling of the notification performance metrics and use it to drive the broker network reconfiguration process through a cost-benefit analysis.

#### 3.4.1 Preliminaries

Let $B$ denote the set of brokers, and $G(V, E)$ denote the DHT topology for the DYNATOPS overlay. A vertex $v \in V$ is an overlay node with a fixed nodeID, corresponding to a fixed node position in the DHT geometry. Its edges to other vertices are DHT links/fingers to other overlay nodes. We assume $|V| = |B|$. DYNATOPS constructs the overlay by specifying the mappings in each overlay between the set of brokers $B$ and the set of overlay nodes $V$ (i.e. nodeIDs) in $G(V, E)$. Taking Figure 3.4 as an example, the mappings of broker A in the overlay before and after reconfiguration are nodes 0 and 1 respectively. Therefore, we can define the broker network configuration as a mapping matrix, $X = \{x_{b,v}\}$ between $B$ and $V$, such that:

$$x_{b,v} = \begin{cases} 
1 & \text{if broker } b \text{ maps to node } v; \\
0 & \text{otherwise.}
\end{cases}$$
Also, let $P$ be the set of topics. Let $S_b$ be the subscriptions of broker $b$. In the rendezvous-based pub/sub implementation, given the broker network configuration $X$, and $S = \{S_b : \forall b \in B\}$ as the set of broker subscriptions, the dissemination path for each topic can be determined easily (for both multicast and unicast implementations).

To formalize the performance of the pub/sub system, we define the topic notification delay, denoted by $d_p$ for topic $p$, as the mass of average delays for forwarding an event publication from the topic RP to each broker that subscribes to topic $p$. That is, let $d_{b,p}$ be the average delay for an event in topic $p$ to reach a specific subscribed broker $b$ from the RP, and it can be easily approximated by the RTTs between brokers during each hop along the path. Then we have $d_p = \sum_{b:p \in S_b} d_{b,p}$.

Meanwhile, we define the topic notification overhead, denoted by $o_p$ for topic $p$, as the total number of forwarded messages required to disseminate an event publication from the RP to all brokers that subscribe to topic $p$. It is the total number of edges in the topic routing tree. This overhead can be divided into two parts: (a) notification subscriber overhead, $o^{\text{sub}}_p$, as the number of messages consumed by subscribing brokers, or as the number of edges in the tree towards subscribing brokers; and (b) notification relay overhead, $o^{\text{relay}}_p$, as the number of the messages consumed by intermediate unrelated relays who have not subscribed to the topic, or as the number of edges in the tree towards relays. Apparently, the notification relay overhead is what we want to eliminate.

Note that in the above formulation we opt out the overhead and delay for forwarding the event from its publisher to the RP. This is because we assume any user in the system can be a potential event publisher (regardless of his subscriptions) so that the average path length from publisher to RP is independent of broker network configurations.

To evaluate the performance of the entire pub/sub system of all topics during a time span $T$, we define cumulative notification overhead and cumulative notification delay under a broker.
network configuration $X$ as follows:

$$O^{relay}(T, X) = \sum_{p \in P} N_p(T) \cdot o_p^{relay} \quad (3.1)$$

$$D(T, X) = \sum_{p \in P} N_p(T) \cdot d_p \quad (3.2)$$

where $N_p(T)$ is the total number of events of topic $p$ published during the period. Here, we assume the brokers’ subscriptions are static during this period so that all the topic routing trees are static and subsequently $o_p^{relay}$ and $d_p$ are independent of time instances.

Now we define notification cost, $C_n(T, X)$ as the pub/sub performance metric that combines both notification overhead and delay:

$$C_n(T, X) = O^{relay}(T, X) + w_d \cdot D(T, X) \quad (3.3)$$

where $w_d \geq 0$ is a relative weight of delay performance to overhead performance.

### 3.4.2 Reconfiguration Process

A question needs to be answered is when and how the system should be reconfigured. Since reconfiguration is not cost-free – it may incur both management overhead and potential disruption of event notifications, it is not plausible to reconfigure the broker network whenever subscriptions change on brokers. To avoid frequent reconfigurations, we define reconfiguration free period, $T_{free}$, as the minimum period between two consecutive reconfigurations in
**Algorithm 1: reconfiguration process**

**Step1-online monitoring:**
online monitoring the event publications and subscription changes on brokers to estimate $C_n(T_{free}, X)$.

**Step2-reconfigurability test:**
fast judging the demand of broker reconfiguration from $C_n(T_{free}, X)$. If demand is low, return to step1; Otherwise, continue to step 3.

**Step3-reconfiguration computation:**
calculating $X'$ to maximize $B_r(X, X') - C_r(X, X')$. If $B_r \leq C_r$ return to step1; Otherwise, continue to step4.

**Step4-reconfiguration protocol:**
coordinating the brokers to transit to $X'$.

DYNATOPS. Intuitively, if the benefit of making a reconfiguration cannot justify its cost, the broker network should not be reconfigured. Every $T_{free}$ period, DYNATOPS performs a cost-benefit analysis to evaluate the reconfiguration benefit against the cost.

Let $X$ be the broker network configurations before reconfiguration, and $X'$ be the one after. The benefit of reconfiguration is evaluated as the reduction in the notification cost during the next $T_{free}$ period after reconfiguration. Formally, we have

$$B_r(X, X') = C_n(T_{free}, X) - C_n(T_{free}, X')$$

(3.4)

On the other hand, the cost of reconfiguration is formulated as the overhead to transit the broker network configuration from $X$ to $X'$. The cost is denoted as $C_r(X, X')$.

We designed a reconfiguration protocol for efficient transitions of broker network and topic configurations in DYNATOPS (See Section 3.4.4), and provided upper bound on the protocol message overhead to reconfigure the broker network and topics (See Section 3.4.5). In practice, the cost can be easily estimated based on the bounds and adjusted by system administration.
Algorithm 1 shows the DYNATOPS reconfiguration process that is periodically executed by the reconfiguration manager. The reconfiguration manager monitors the rate of event publications as well as subscriptions to calculate $C_n$.

The reconfigurability test aims to prune out unnecessary reconfiguration computations when estimated reconfiguration demand is low for the next period (e.g., when the system is under-loaded and the number of published events are few). The intuition here is that when the broker network is already fairly optimized or the event notification cost is low, the marginal benefit of reconfiguration won’t be able to justify the cost of it. Hence, we compare the estimated $C_n$ with a cost threshold to decide the necessity for reconfiguration computation. A reconfiguration computation is desired only if it is larger than the threshold.

### 3.4.3 Reconfiguration Computation

DYNATOPS performs reconfiguration computation in order to determine optimal configuration $X'$ that maximizes $B_r - C_r$. In order to solve the optimization problem, we first define the basic reconfiguration problem without taking into account the reconfiguration cost: **Structure and Subscription aware Reconfiguration (SSR)** as follows.

**Definition 3.1.** $SSR(G(V, E), S, N_p(T))$: Being aware of the overlay structure $G(V, E)$, brokers subscriptions $S$ and $N_p(T)$ as the number of event publications in each topic during a period, find $X$ that minimizes the notification cost:

$$\arg \min_X C_n(T, X)$$  \hspace{1cm} (3.5)

The SSR problem is NP hard to solve. We present a greedy algorithm with upper bound complexity $O(|P|^2|V|^2\log(|V|))$ (see Section 3.4.5 for proof). Before describing the algorithm, we first introduce two new derivatives from the notification cost: *per topic notification cost*
and *per broker notification cost*. The *per topic notification cost*, \( C_{n|p} \), evaluates each topic for their contributions to the notification cost \( C_n(T, X) \) in equation 3.3. It can be formulated as follows:

\[
C_{n|p} = N_p(T) \cdot (o_p^{relay} + w_d \cdot d_p)
\] (3.6)

where \( o_p^{relay} \) and \( d_p \) are the notification relay overhead and notification delay of topic \( p \) we introduced earlier.

Similarly, the *per broker notification cost*, \( C_{n|b} \), evaluates each broker for their contributions to the notification cost. It can be formulated as follows:

\[
C_{n|b} = \sum_{p \in P} N_p(T) \cdot (o_b,p^{relay} + w_d \cdot s_{b,p} \cdot d_{b,p})
\] (3.7)

where \( o_b,p^{relay} \) equals to 1 if \( b \) is an unrelated relay in the routing tree of topic \( p \), and 0 otherwise; \( s_{b,p} \) equals to 1 if \( b \) subscribes to topic \( p \), and 0 otherwise; and \( d_{b,p} \) is the delay for an event in topic \( p \) to reach broker \( b \) from the topic RP.

It is not difficult to derive the following relationships between the derivative costs and the notification cost of the system:

\[
C_n = \sum_{p \in P} C_{n|p}
\] (3.8)

and

\[
C_n = \sum_{b \in B} C_{n|b}
\] (3.9)

Algorithm 2 shows the proposed algorithm, SSR-Greedy, for the SSR problem. Given an initial configuration of broker network, the algorithm iteratively improves the solution by
Algorithm 2: SSR-Greedy

**Input:** \( G(V,E), S, N_p^{(t_0,t_0+T)} \)

**Output:** List of \( X \) that along the improvement path of \( C_{(t_0,t_0+T)}(X) \)

\( \tilde{X} = \) initial configuration; \( \min\tilde{C} = C_{(t_0,t_0+T)}(\tilde{X}) \);

\( \text{List} = [] \) while \( \min\tilde{C} < \min C \) do

\( \min C = \min\tilde{C}, X = \tilde{X}, \text{List.add}(X) \)

foreach \( X' \in Nb(X) \) do

\( \text{tmpC} = C_{(t_0,t_0+T)}(X') \)

if \( \text{tmpC} < \min C \) then

\( \min C = \text{tmpC}, \tilde{X} = X' \)

end

end

end

examining the neighborhood (we will explain later) of the current best configuration and greedily moving to the new one that minimizes the notification cost in current iteration. The algorithm returns when a local optimal is reached. This iterative process forms an improvement path of configurations with improved notification costs. The algorithm returns all the configurations along the improvement path.

The neighborhood of a configuration \( X \) is a set of configurations that directly derivable from the current one. It consists of configurations that are derived by swapping the mapping of the broker with the highest external overhead grade with that of another broker. Apparently the neighborhood contains only \(|V| - 1\) configurations. By searching new configurations that have a different mapping for the bottleneck broker who has the worst broker cost, we have a better chance to improve the total cost.

To solve the original reconfiguration computation problem that maximizes \( B_r - C_r \), we adapt the SSR-Greedy algorithm by taking into account the reconfiguration cost \( C_r(X, X') \) for each configuration along the improvement path. Algorithm 3 shows the DYNATOPS reconfiguration algorithm. It consists of two steps: At the first step, the algorithm applies the SSR-Greedy algorithm to find a set of configurations with improved notification cost.
compared to the current configuration. At the second step, the algorithm calculates $B_r - C_r$ for each of the configurations and the best one is selected.

\textbf{Algorithm 3: DYNATOPS-Greedy}

\begin{itemize}
\item \textbf{Input:} $G(V, E), S, N_p^{(t_0, t_0 + T_{free})}, X$
\item \textbf{Output:} $X'$ that maximize $B_r(X, X') - C_r(X, X')$
\end{itemize}

\begin{algorithmic}
\State \textbf{step one:}
\State List = SSBR - Greedy();
\State \textbf{step two:}
\State $C = C_{(t_0, t_0 + T_{free})}(X)$;
\State bestUtil = 0;
\State $X' = X$;
\ForEach $	ilde{X}$ in List do
\State $\tilde{C} = A$.get($\tilde{X}$);
\State Util = $C - \tilde{C} - Cst(X, \tilde{X})$;
\If{Util > bestUtil} then
\State bestUtil = Util;
\State $X' = \tilde{X}$;
\EndIf
\EndFor
\end{algorithmic}

3.4.4 Reconfiguration Protocol

Figure 3.5 shows the data structures maintained by each DYNATOPS broker. The data structures can be partitioned into two layers: 1) the overlay layer structures, which consists of brokers’ overlay nodeID and fingers for constructing the overlay; and 2) the pub/sub layer structures which consists of brokers’ subscription and topicTrees to realize pub/sub functions.

For the overlay layer structures, DYNATOPS incorporates the distributed protocol of Chord DHT [146] to maintain its dissemination overlays; The protocol maintains fingers of each overlay node to ensure a converged DHT under node churns. For the pub/sub layer structures, DYNATOPS incorporates a distributed pub/sub protocol which is the same as the
**Data Structures:**

- **bid**: the broker’s identifier
- **version**: the version number of the configuration
- **nodeID**: logical node ID of the broker in the overlay
- **fingers**: tuples `<nid, address>` indicating DHT routing fingers in the overlay
- **subscription**: a set of topics subscribed by the broker
- **topicTrees**: set of records `<topic, parent_nid, child_nids>` for joined topic routing trees

Figure 3.5: DYNATOPS broker data structure.

one specified in Bayeux [168] to maintain its pub/sub topic routing trees; The protocol updates `topicTrees` by allowing a broker to join/leave topic routing trees when its subscription changes.

A system configuration is determined by the reconfiguration manager and distributed to each broker. It consists of the pair `(X, P)` which specifies the `nodeIDs` at the overlay layer that keeps track of the brokers’ logical positions in the dissemination overlays, and the `topicSpace` at the pub/sub layer that keeps track of the topics handled by each of the overlays. Each system configuration has a unique version ID. For a converged system configuration, all brokers should run under the same version. Our reconfiguration protocol ensures convergence of the configuration among all brokers and seamless transition between different configurations so that correct event routing at all times is guaranteed.

Let `s` denote the version of the system configuration before a reconfiguration process. Let `s'` denote the new version after reconfiguration. Now we describe the reconfiguration protocol exchanged between the reconfiguration manager and brokers:

1. reconfiguration initialization.

   - **reconfiguration manager**: broadcasts to all brokers a RCFG_INIT message containing `{s, s'}` as the pair of versionIDs.
• *broker*: upon receive the message, replicates data structures of $s$ as those of $s'$, then sends to the reconfiguration manager a ACK_RCFG_INIT message.

2. overlay layer reconfiguration.

• *reconfiguration manager*: sends to each broker a RCFG_OVERLAY message containing its $\text{diff}(\text{nodeID}(s), \text{nodeID}(s'))$ as the overlay layer configuration changes to be made.

• *broker*: If no changes, sends to reconfiguration manager a ACK_RCFG_OVERLAY message. Otherwise, updates its $\text{nodeID}(s')$ and rejoin the DHT of $s'$ with the updated node ID; sends to reconfiguration manager a ACK_RCFG_OVERLAY message when done.

3. pub/sub layer reconfiguration.

• *reconfiguration manager*: broadcasts to all brokers a RCFG_PUBSUB message.

• *broker*: 1) for each overlay finds the broker $b$ who will replace its position in $s'$; copies its own $\text{topicTrees}(s)$ records of that overlay to $b$’s $\text{topicTrees}(s')$.

• *broker*: 2) compares *subscription* and $\text{topicTrees}(s')$: for each subscribed topic in *subscription*, joins the routing tree if the topic is not in $\text{topicTrees}(s')$; also, for topics in $\text{topicTrees}(s')$ but not in *subscription*, leaves the tree if the broker is a leaf.

• *broker*: 3) sends to reconfiguration manager a ACK_RCFG_PUBSUB message when above steps are done.

4. reconfiguration finalization.

• *reconfiguration manager*: broadcasts to all brokers a RCFG_FINISH message containing $\{s, s'\}$ as the pair of version IDs before and after the indicated reconfiguration.
• broker: makes data structures of \( s' \) as the default in use. Sends to reconfiguration manager a ACK:\_\text{RCFG}\_FINISH message.

It is important for each broker to maintain two sets of data structures (\( s \) and \( s' \)) during the reconfiguration process in order to avoid event losses and incorrect routing. An event published during the reconfiguration is forwarded by the data plane and tagged with the version of the data structures used for table lookup so that its next hop can use the same version to ensure consistency. To cope with uncertain delays in the network, each step of the protocol is enforced by atomic operation [61]. That is, next step will not be triggered unless all brokers have acknowledged accomplishment of the current step.

3.4.5 Analysis

In this section, we first show the NP-hardness of the presented reconfiguration problems and then provide theoretical analysis on the proposed reconfiguration algorithm and reconfiguration protocol.

Lemma 3.2. The Structure and Subscription aware Reconfiguration (SSR) problem (eq. 3.5) is at least as hard as the following overhead optimization problem:

\[
\arg \min_x O^{\text{relay}}(T, X)
\]

Proof. By equation 3.3, the above overhead optimization problem is a special case of the SSR problem where \( w_d = 0 \). The original SSR problem is at least as hard as this special case.

Lemma 3.3. The overhead optimization problem in equation 3.10 is NP-hard to solve.

Proof. The decision version of the Overhead-one problem can be put as follows:
Overhead-Decision: Given a positive value $C$, determine whether there exits a configuration $X$ such that the $O^{\text{relay}}(T, X) = C$.

We reduce a well-known NP-complete problem, the Dominating Set Problem (DCP), to the above decision problem. Due to space limitation, we omit the reduction detail. It is known that if the decision problem of an optimization problem is NP-complete, then the original problem must be NP-hard. Therefore, the overhead-one problem is NP-hard. Since overhead-one problem is a special case of the overhead optimization problem, the overhead optimization problem is NP-hard too.

Given the above two lemmas, we can easily derive the following theorem and corollary on the NP-hardness of the presented problems.

**Theorem 3.4.** The Structure and Subscription aware Reconfiguration (SSR) problem (eq. 3.5) is NP-hard to solve.

**Corollary 3.5.** The reconfiguration problem to determine optimal configuration $X$ that maximize $B_r - C_r$ is NP-hard to solve.

Now we prove that our proposed SSR-Greedy and reconfiguration algorithms solve the above problems efficiently.

**Theorem 3.6.** The computational complexity of the SSR-Greedy algorithm is upper bounded by $O(|P|^2|V|^2\log|V|)$.

*Proof.* We proof the theorem by proving the following two lemmas, which provide bounds on the number of iterations the algorithm can run and the computational complexity of each iteration in the algorithm. The total computational complexity is the multiplication of the two.

**Lemma 3.7.** The SSR-Greedy algorithm is upper bounded by $O(|P| \cdot |V|)$ iterations to end.
Proof. It is easy to see that the greedy algorithm is loop free. In each iteration the algorithm must make improvement on the notification cost $C_n$ to proceed until $C_n$ reaches its minimum or zero. By equation 3.3, the cost consists of overhead cost $O_{\text{relay}}$ and delay cost $D$. We know $O_{\text{relay}}$ is integer valued, and by manipulating the delay unit we can approximate the RRT delay between any pair of brokers in integer too. Thus, the improvement step on $C_n$ for each iteration is at least 1 as the minimal positive integer, and the total number of possible iterations of algorithm is bounded by the maximum value of $C_n$. Consider a single topic routing tree, both its unrelated relays and cumulative delays from the RP to its leafs are bounded by $O(|V|)$. Then, $C_n$ is bounded by $O(|P| \cdot |V|)$ by taking into account all topic routing trees in the system. Therefore, the algorithm is also bounded by $O(|P| \cdot |V|)$ iterations to end.

Lemma 3.8. The computational complexity of an iteration in SSR-Greedy algorithm is $O(|P||V|\log|V|)$.

Proof. By algorithm 2, each iteration explores $|V| - 1$ neighboring configurations of the current one. A neighboring configuration is derived by a broker-exchange (i.e. exchange positions of a pair of brokers in an overlay). For a broker-exchange, we need to evaluate the per broker notification cost $C_{n,b}$ of the two brokers, which is $O(|P|\log|V|)$. Thus, the total complexity to evaluate all neighboring configurations in their notification cost in an iteration is $(|V| - 1) \cdot O(|P|\log|V|)$, which is $O(|P||V|\log|V|)$.

Theorem 3.9. The computational complexity of the DYNATOPS-reconfiguration problem is upper bounded by $O(|P|^2|V|^2\log|V|)$.

Proof. The algorithm consists of two steps. The first step is the SSR-Greedy algorithm, whose complexity is $O(|P|^2|V|^2\log|V|)$ as theorem 3.9. It is easy to see that the second step has a linear computational complexity to the number of elements in the list of improved configurations. The number is equal to the number of iterations of the SSR-Greedy algorithm,
which is bounded by $O(|P| \cdot |V|)$ as lemma 3.7. Thus, the DYNATOPS-reconfiguration problem is upper bounded by $O(|P|^2|V|^2 \log |V|)$.

Besides the proposed algorithms, DYNATOPS takes advantage of an efficient reconfiguration protocol to ensure seamless reconfigurations of the broker network based on the computed optimal configurations. Now we analyze the overhead efficiency of the proposed reconfiguration protocol.

**Theorem 3.10.** For a reconfiguration process, let $\Lambda(X, X')$ be the number of brokers that need change its mappings, then the control message overhead for reconfiguring the broker network from configuration $X$ to $X'$ is upper bounded by $O((\log^2(|V|) + |P|\log(|V|))\Lambda(X, X') + |V|)$.

**Proof.** The reconfiguration control messages are 1) exchanged between the reconfiguration manager and the brokers in each of the four stages of the reconfiguration protocol; and 2) exchanged among brokers in stage two (overlay layer reconfiguration) and three (pub/sub layer reconfiguration). First it is easy to see that the messages exchanged between the reconfiguration manager and the brokers are $O(|V|)$. Now we consider messages exchanged among brokers. To reconfigure the overlay layer, overlay nodes with changed mapping leave and rejoin the Chord DHT. According to Chord, each such node incurs $O(\log |V|^2)$ message overhead. Thus, stage 2 incurs $O(\log^2(|V|)\Lambda(X, X'))$ overhead. For stage 3, the control messages are issued to recover pub/sub for remapped brokers. For brokers moved in the overlay, they need to repair the pub/sub of their subscribed topics. It takes $O(\log |V|)$ for a broker to leave/join a topic. Thus, the overhead to repair remapped brokers are $O(|P|\log |V|)\Lambda(X, X')$. Therefore, the total control message overhead is upper bounded by $O((\log^2(|V|) + |P|\log(|V|))\Lambda(X, X') + |V|)$. 

49
3.5 Performance Evaluation

To evaluate the performance of DYNATOPS, we created two models that emulate the real-world subscription dynamics, and compared DYNATOPS with several well-known topic-based pub/sub implementations for its subscription management and event publication efficiencies.

3.5.1 Dynamic Subscriptions Modeling

We considered two models of subscription changes: (1) a location-based subscription model that emulates users’ dynamic subscriptions in many geosocial networking applications and location-based services; and (2) a generic Poisson dynamic subscription model that emulates changing interests of users in timely and popular topics.

Location-based Subscription Model

To create a large-scaled dynamic subscription model, we considered a twitter dataset [53] containing 3 million location checkins (in longitude and latitude coordinates) from over 40K Twitter users in the U.S. for 3 months period from Aug. 1st 2010 to Oct. 31st 2010. To convert the dynamic location checkins into dynamic subscriptions, we divided the U.S. geography into grids of one degree of latitude by one degree of longitude, the size of which is about 3500 mile$^2$. This results in $20 \times 60$ (i.e. 1200) grids, and we considered each of them as a location topic (i.e. totally 1200 topics). Users’ checkins over time at different grids are considered as traces of their movement.

We presented our preliminary findings on users’ grid checkins in Fig. 3.6(a) and 3.6(b). We observed over 450K grid visits events (we treat users’ successive checkins to locations in the
same grid as a single event), from which we extracted over 160K unique (user, grid) pairs. 1/3 of them are single-time visit (i.e. a user visits to the grid only once) while the other 2/3 are at least repeated once by the user (Fig. 3.6(a)). We also analyzed the linger time of all the events. We treated the lapse of time before a user checked into a new grid as the linger time he/she stayed in the current grid. Our results indicates that about half of the 450K visits have a linger time less than a day. On the other hand, there are 10% of the visits are relatively long-lived, with a linger time over a week (Fig. 3.6(b)).

For users’ dynamic subscriptions, we assume a user always subscribed to the grids he/she was residing in. Furthermore, during experiments we let each user randomly choose 4 other users as friends, and constantly follow/subscribe to the current grids/locations of his/her friends (Fig. 3.6(c)). This mimics a geosocial networking application, e.g. foursquare, where users can share their locations and activities with friends.

To experiment the pub/sub system, we considered 1200 brokers such that each broker is located inside a grid. The RTT delay between a pair of brokers and that between a broker and a user are random variables with their means proportional to their geographical distances.

**Generic Poisson Subscription Model**

In this model, we emulated a time period of $T = 100 hrs$ with time granularity of $\Delta t = 1 hr$. We experimented with 50K users, 100 brokers and 1000 topics. Each user subscribes to 5 topics according to the patterns of their subscriptions. The dynamics of subscription changes over time is modeled as a Poisson process. We considered three patterns for users’ topic subscriptions:

- **Uniform Distribution**: users subscribe to topics from the topic space in a uniform random manner.
• Zipf Distribution: topic popularity follows Zipf distribution. The probability for the $i^{th}$ topic in the topic space is proportional to $(i + 1)^{-\sigma}$, $i = 1, 2, \ldots, |P|$.

• Multimodal: users’ subscriptions fall into modes. We evenly partitioned the topic space into $m = 100$ modes, and each mode contains $n_{\text{mode}} = 10$ topics. A user first randomly select a interested modes and then choose topics uniformly at random from the selected mode.

![Graph](image)

(a) distribution of users’ visited grids in number of repeated visits
(b) distribution of users’ grid-visits in their linger time

![Map](image)

(c) Location-based dynamic subscriptions from Twitter data

Figure 3.6: Dynamic subscription model from Twitter dataset.

### 3.5.2 Comparison Systems

We evaluated DYNATOPS along multiple dimensions by extensive simulations. The key dimensions that serve as metrics for our study include subscription management overhead,
as the number of control messages exchanged between brokers to update topic trees when brokers’ subscription changes, and to update information in clusters; notification delay as the average latency for publications to be delivered from publishers to subscribers; notification overhead as the number of messages to deliver event publications; and reconfiguration overhead as the number of control messages for the reconfiguration protocol.

We compared DYNATOPS with several existing topic-based pub/sub systems of different categories: 1) Bayeux, a well-known pub/sub system atop the Tapestry DHT [166]; Bayeux was picked because it is a rendezvous-based pub/sub system on a structured overlay – providing a common ground for comparing with our system. Unlike DYNATOPS, Bayeux does not optimize its brokers based on subscriptions or perform reconfigurations, giving us an opportunity to test the value of these techniques. 2) Topic Connected Overlay (TCO), a pub/sub overlay [54, 129, 49, 50] that eliminates unrelated relay brokers to provide optimal notification overhead efficiency by connecting brokers that subscribe to the same topic to form a connected subgraph; Since TCO optimizes broker topology at all times for notification efficiency, it again gives us an opportunity to test the performance and efficiency of our cost-driven broker reconfiguration policy. 3) GeoPS [29], a pub/sub service specific to location-based subscriptions. It takes advantage of geography-aware overlay hierarchy and geocasting technique for efficient subscription management and publication notifications.

Furthermore, to study the value of the proposed schemes, we considered three versions of DYNATOPS system: (a) DYNATOPS(BNR) explores the benefit of the broker network reconfiguration scheme with an equivalent user placement scheme as other systems; (b) DYNATOPS(UP) explores the benefit of the user placement scheme only, and (c) DYNATOPS is the overall mechanism that incorporates both user placement and broker network reconfiguration schemes.

We implemented our simulator and all the above systems in Java. In our simulation, the broker networks of Bayeux and GeoPS were considered to be static. For the Bayeux simu-
lation, brokers join the Tapestry Overlay with random IDs uniformly distributed over the ID space. For the GeoPS simulation, we divided the geography into power-of-2 grids on both edges of a rectangular geography as required by the GeoPS system. Specifically, we divide the U.S. into $32 \times 32$ grids (i.e. 1024 in total). TCO requires its broker overlay to be re-configured whenever topic-connected property is no longer hold due to subscription changes on brokers. We reconfigure TCO by running the DCB-M algorithm [50] for partitions of nodes having changed subscriptions. Since there is no existing reconfiguration protocol to refer to, we assume the message overhead to conduct each reconfiguration is the number of links that are changed in the network topology. Users are placed on brokers at the start of each simulation, and their subscriptions change over time following the specific subscription model. Since Bayeux and TCO do not have a specific user placement policy, we considered two commonly used policies in the simulation: 1) Static selection, where a user is statically assigned a broker and 2) Location-based selection, where each broker is responsible for users in a specific region and users handover to new brokers when they move to different regions.

3.5.3 Experimental Results

Basic Results

We experimented DYNATOPS and compared its performance with existing systems under the two subscription models.

*location-based subscription model:* Figure 3.7 shows the results for the location-based subscription model. We compared DYNATOPS with Bayeux and GeoPS in their subscription and publication performances. For Bayeux, we considered both static and location-based user placement policies indicated by “Bayeux(static)” and “Bayeux(loc)”. Furthermore, we considered proximity neighbor selection(PNS) in its overlay construction, along with location-based user placement, indicated by “Bayeux(loc+PNS)”. When experimented with
Figure 3.7: Location-based subscription model results.
user placement technique, we formed broker clusters based on their geographical proximity to avoid extensive maintenance overhead, and let each cluster manage users in a continuous geographical area close to it.

We observe that by grouping users with similar subscriptions, DYNATOPS significantly reduces the subscription management overhead against other systems (Fig. 3.7(a)). Moreover, we observe that the overhead first decreases with the increase of the cluster size $c$, which indicates the reduction in topic tree updates in the broker network. Under large cluster sizes (e.g. $c = 50$), however, the overhead for updating brokers’ states in a cluster becomes significant, so the total subscription management overhead starts to increase.
Fig. 3.7(b) shows the standard deviation in subscription load on brokers. We observe that pure similarity based user placement ($w_l = 0$) worsens the load imbalance on brokers, especially when the cluster size is large. However, the issue is greatly improved by adjusting the weight for the load factor in the algorithm. We also evaluated the number of user handovers due to mobility in GeoPS and DYNATOPS. We observe that DYNATOPS reduces the user handovers by over 80% against GeoPS as shown in Fig. 3.7(c).

Both user placement and broker network reconfiguration can improve notification delay (Fig. 3.7(d)) and overhead (Fig. 3.7(e)). This is because by clustering users with the same topic the user placement can reduce the number of brokers that need to subscribe to the topic so as to reduce the size of the topic routing tree. Combining the two algorithms DYNATOPS is highly efficient and it provides over 60% improvement against Bayeux and 30% against GeoPS in delay, and over 40% improvement against Bayeux and 20% against GeoPS in overhead (DYNATOPS $w_d = 1$). Furthermore, the notification performance improvement increases as the increase of the cluster size (Fig. 3.7(f)). This is because with larger cluster size the user placement makes the brokers’ subscriptions more skewed, which favors the broker network reconfiguration to reduce unrelated relays in the topology.

![Figure 3.7: Poisson subscription model results in different patterns (subscription change rate = 0.1, $T_{free} = 10hrs$).](image)

**Poisson subscription model:** Fig. 3.8 shows the results of the Poisson subscription model
where users subscriptions change with varying Poisson rates. We compared DYNATOPS with Bayeux and TCO for subscription and publication performances. In both Bayeux and TCO, we assume users are statically assigned brokers in a uniformly random manner. For publication performances, we also considered DYNATOPS(BNR) where the user placement is the same as those for Bayeux and TCO, to evaluate DYNATOPS under the broker network reconfiguration technique along. For simplicity, in the experiment we assumed a unit RTT delay between any pair of brokers so that the notification delay is dominated by the number of overlay hops.

Fig. 3.8(a) to 3.8(d) show the experimental results where users subscriptions follow the multimodal pattern under varying Poisson rate of user subscription changes. We observe that DYNATOPS reduces the number of brokers’ subscription changes by 80% against other systems (Fig. 3.8(a)), resulting in significantly less subscription management overhead. Fig. 3.8(b) and 3.8(c) show the publication delay and overhead performances of different pub/sub systems. TCO achieves a better overhead performance than Bayeux and DYNATOPS(BNR) because of the topic-connected property. However, we observe a worse delay performance because its suboverlay construction is not delay aware. On the other hand, DYNATOPS considering both user placement and broker network reconfiguration outperforms Bayeux and TCO on both delay and overhead. It provides 10% improvement in delay and over 50% improvement in overhead against the compared systems.

We also compared the reconfiguration cost between DYNATOPS and TCO (Fig. 3.8(d)). The reconfiguration overhead for TCO increases dramatically with the increase of the rate of users subscription changes. This makes the scheme infeasible to be applied in highly dynamic environment. On the other hand, DYNATOPS’s reconfiguration overhead is less sensitive to the rate of users subscription changes. It is over 80% less than that of TCO when users’ subscriptions change fast.

The notification performance of each system under different subscription patterns of users
are shown in Fig. 3.9(a) and Fig. 3.9(b). We observe that TCO has a low delay under zipf pattern where subscription is highly skewed but worse under other subscription patterns. On the other hand, DYNATOPS outperforms other systems under all subscription patterns.

Figure 3.10: Results under varying reconfiguration free period. For location-based model, \( c = 10 \); For Poisson model, subscription change rate = 0.025 and \( c = 5 \).

**Reconfiguration free period**

To gain better understanding of the performance, we experimented with various reconfiguration free period \( T_{free} \) in both location-based and Poisson models. In Fig. 3.10. We observe that with the increase of the reconfiguration free period, the notification performance degrades slightly and the reconfiguration overhead decreases because less reconfigurations were
triggered. It is worth noting that DYNATOPS only experienced a slight degradation in notification efficiency when the reconfiguration free period increases. This is because the user placement technique stabilized brokers subscriptions such that they do not experience dramatic subscription changes over time in spite of dynamic users’ subscriptions.

### Scalability

We also experimented with various sizes of the broker network and of the topic space to evaluate the time efficiency and performance of the DYNATOPS-Greedy configuration algorithm. We ran the algorithm on a Dell workstation with a QuadCore 2GHz CPU and 2G memory. The performance of the output DYNATOPS configurations are compared against that of a bootstrap configuration from consistent hashing of brokers public keys or IP addresses for their nodeIDs. This is the configuration approach adopted by most existing DHT-based pub/sub systems [48, 168, 138, 62]. The performance was evaluated under two subscription patterns on brokers: uniform distribution subscriptions and multimodal subscriptions. Figure 3.11 and Table 3.1 show the performance and computation time of the algorithm. For varying \(|B|\), we fixed \(|P| = 100\), and for varying \(|P|\) we fixed \(|B| = 100\). We observe that DYNATOPS configuration always provides improved notification performance against consistent hashing under various size of broker networks and topic spaces. The improvement is larger under skewed subscription patterns than the uniform pattern. Furthermore, the proposed configuration algorithm is efficient to compute DYNATOPS configurations for a large broker network and topic space.

<table>
<thead>
<tr>
<th>size</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>50</th>
<th>500</th>
<th>5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>28ms</td>
<td>184ms</td>
<td>96s</td>
<td>98ms</td>
<td>502ms</td>
<td>1.9s</td>
</tr>
</tbody>
</table>

Table 3.1: The computation time of the DYNATOPS-Greedy algorithm under various size of broker network \(|B|\) and of topic space \(|P|\).
Figure 3.11: Performance savings of DYNATOPS configurations against CHash under various size of broker network $|B|$ and of topic space $|P|$. We varied $|B|$ with fixed $|P| = 100$, and varied $|P|$ with fixed $|B| = 100$. “M” denotes multimodal pattern and “U” denotes uniform pattern.

3.6 Conclusion

In this chapter, we studied the problem of societal scale instant information sharing featured by changing information interests for a very large number of users. We conjecture that a topic-based publish/subscribe system can form the basis of an efficient architecture for this service. We propose and develop DYNATOPS, a publish subscribe middleware for societal scale applications, that can deal with dynamic, yet short lived subscriptions. DYNATOPS users are moderately repositioned on brokers for efficient subscription management and brokers are moderately repositioned on the overlay structure for efficient event notifications, to adapt to the publications and subscription dynamics. Unlike existing systems where the overlay topology changes in a self-organizing manner in response to the changes of subscriptions, DYNATOPS performs planned reconfiguration utilizing a cost-driven reconfiguration process. The proposed approach can significantly reduce the reconfiguration cost while maintaining a high notification performance as compared to state-of-the-art systems.
Chapter 4

Instant Information Sharing:
Dissemination Layer

In this chapter, we study the dissemination layer problem for the class of instant information sharing applications. Figure 4.1 highlights the contributions of this chapter in the design framework for societal scale information sharing. We focus on information notification in extreme situations e.g., disasters, and the goals are to deliver appropriate messages to all relevant recipients with very high reliability in a timely manner. The challenges of this problem is that geographically correlated failures as a result of the disaster hinder the ability to reach recipients inside the corresponding failed region. In this chapter we present GSFord, a reliable geo-social notification middleware that is aware of (a) the geographies in which the message needs to be disseminated and (b) the social network characteristics of the intended recipient, in order to maximize/increase the coverage and reliability. GSFord builds robust geo-aware P2P overlays to provide efficient location-based message delivery and reliable storage of geo-social information of recipients. Our technique efficiently accommodates non-uniform user distributions. The overlays also provide reliable storage of social network information under extreme regional failures. We show that GSFord is able to efficiently
disseminate event messages to dynamically defined recipients who are either geographically or socially correlated to the event. Under geographically correlated regional failures, GSFord reliably delivers messages to unfailed recipients outside the failed region through its online dissemination structure. To better reach those in failed regions, GSFord exploits a targeted social diffusion process triggered by using the stored social information. The social diffusion process enables propagation of event messages towards expected recipients through diverse out-of-band communication channels.

Figure 4.1: Contributions of Chapter 4 in the societal scale information sharing framework.

We evaluate GSFord through extensive simulations in terms of reliability and efficiency. We show that the importance of preserving the social information of recipients in GSFord to maximize the eventual coverage of the event notification system, especially in the presence of large geographically correlated failures. Specifically, once GSFord ensures that socially correlated recipients receive the initial event message, the coverage of GSFord increases by around 15%; additionally around 90% of nodes experience a decrease in dissemination latency of up to 80%. Our results also demonstrate that the GSFord overlay provides efficient and reliable geo-aware regional multicasting with reasonable network overhead. In particular,
the GSFord overlay maintains its performance even though the large-scale users are non-uniformly distributed.

The rest of the chapter is organized as follows. Section 4.1 describe the problem of information notification in disasters. We give our system’s overview, build up notations in Section 4.2. We present the GSFord solution in detail in Section 4.3 and Section 4.4. We evaluate the solution through extensive simulations and experiments in Section 4.5. The chapter is concluded in Section 4.6.

4.1 Introduction

The eventual goal of an event notification system is to deliver appropriate messages to all relevant recipients with very high reliability in a timely manner. We are motivated by the case of disaster alerting and warning systems where a notification system alerts impacted populations on how to take self-protective action and prevent loss of lives and property. In extreme situations, the eventual coverage/reliability of the message dissemination is of utmost importance.

Firstly, in natural or human-induced disasters, information needs are strongly correlated to the geographical location of the event. For instance, severe ground shaking and damage occur in areas that are within close geographic proximity to an earthquake epicenter; Tornadoes are likely to destroy infrastructures in areas that lie along their damage paths. Ideally, a notification system should exploit this geographical correlation to warn populations in the disaster region as soon as possible.

Secondly, there exists social correlation of information needs. Individuals are often interested in receiving accurate information on disasters (occurrence and progress) which may affect their loved ones and desire detailed information on the current status of their relatives and
friends. They are interested in receiving the information even if they are not physically located in the disaster region. For example, if a forest fire occurs near an elementary school, parents will need to be informed of the current situation as soon as possible; information on the current status of their children will help them make decisions on how to respond quickly and effectively (e.g. plan details on how families will evacuate).

Geo-correlated failure propagation caused by disasters adds to the complexity of event notification. For example, when an earthquake occurs, one can expect catastrophic network failures or blackouts in the affected region [21]. Recipients inside and outside the region can be affected by the failure of the dissemination infrastructure. Scalability also poses additional challenges - as the number of participants increase, personalized warning can consume significant resources (e.g. bandwidth, processing power) and possible bottlenecks can result in delays and warning inconsistencies.

Typical approaches to disaster alerting include sirens, subscription-based warnings from a centralized broker (USGS Shakecast, SMS Gateways) and cell broadcasts (currently implemented in parts of Asia and Europe). These systems lack information specificity by providing a broadcasting service to the entire set of participating recipients. Application layer multicasting (ALM) [65, 87, 70] can be used to multicast specific messages to a fine-grained set of recipients, but these systems are not aware of the geography. Subscription based systems [142] and geography aware P2P systems [31, 163, 29] have been proposed for dynamic geography aware data retrieval or dissemination. However, these efforts did not consider the need to reach socially correlated remote recipients who are currently not located in the disaster region. Additionally these efforts lack explicit fault tolerance against the geographically correlated regional failures [92, 126].
4.2 GSFord Approach

Figure 4.2 provides an overview of GSFord. GSFord operates over a **global target geography** (GTG) that defines the global geographical region over which geo-social notifications occur. We begin by describing some basic concepts, notations and assumptions used in GSFord to capture the social and geographical aspects for efficient and effective notification.

![Figure 4.2: The overview of GSFord.](image)

4.2.1 Social Concepts of GSFord

The eventual recipients of notifications in GSFord are real users located inside the global target geography interested in receiving the notification and socially connected to each other using disparate communication channels. More formally, the set of users in the GTG form a social network graph, \( G := (V, E) \), where a user \( u \in V \) is connected to another user \( v \in V \) if there exists a social link \( e_{u,v} \in E \). In the above case, \( v \) is referred to as a social
acquaintance of $u$. Let $A(u)$ denote the set of social acquaintances of $u$, so

$$A(u) = \{ v : v \in V, e_{u,v} \in E \}$$

The degree centrality of a user is defined as the number of social acquaintances he/she has. We use $C_D(u)$ to denote the degree centrality of a user $u$, so

$$C_D(u) = |A(u)|$$

There is a subset of users in $A(u)$ who are closely tied to $u$ in a society, such as family members, and therefore highly interested in the current status of $u$ when disasters and the ensuing event notification process occurs. They are referred to as the social friends of $u$. We denote the set of social friends of $u$ as $F(u)$ where $F(u) \subseteq A(u)$. $F(u)$ and $A(u)$ can be obtained in many ways, such as direct user input, e.g. the list of emergency contacts that organizations usually require individuals to provide. It may also be obtained by profiling and analyzing a known social network graph, such as online social network graphs from Facebook, Twitter or even phone network graphs from elementary school offices, based on the connectivity of social entities and the frequency of correspondence [94].

In GSFord, every user has a unique public Social ID (SID) that hides his real identity. We denote a user $u$’s SID as $SID_u$. We assume the presence of a registration server, which resides in an authorized domain, can generate unique SIDs for prospective users based on their personal information (name and contacts) and manage the mapping between a user’s real identity and his/her SID. During the registration, the server also transforms the list of social friends of a user $u$ into a list of SIDs, say $\{SID_v : v \in F(u)\}$, by referring to their mappings. We also assume that the registration server is not a bottleneck since (a)
registration and mapping is a one-time process, and (b) it is usually performed apriori, i.e. before notification; issues of surge and overload at the registration server are out of scope of the thesis.

4.2.2 Geographical Concepts of GSFord

GSFord has prior knowledge of the GTG; For simplicity, this is mapped into a 2 dimensional rectangular region, denoted as $GTG := ((0,0), (x_{max}, y_{max}))$, where $(0,0)$ and $(x_{max}, y_{max})$ are the coordinates for the bottom-left and top-right corners of the rectangular region respectively. A user executes the GSFord application on a personal device; we call the logical host on which the user resides as a GSFord Physical Node (PNode), and denote the PNode for a user $u$ as $PN_u$. We assume that a PNode has knowledge of its location (using positioning technologies such as GPS, WiFi fingerprinting, etc.), and the location is mapped to a point in the GTG, denoted as $Loc(PN_u) := (x, y)$. The PNode, $PN_u$, is used by the user to join and receive notifications from GSFord. Moreover, it maintains the user’s geo-social information, which includes its current location $Loc(PN_u)$, the user’s public social ID $SID_u$, the user’s degree centrality $C_D(u)$ and the list of SIDs of his social friends $\{SID_v : v \in F(u)\}$.

We select a subset of PNodes that are more trusted and reliable and refer to them as Trusted PNodes (T-PNodes). T-PNodes typically correspond to users that represent public figures and authorities, e.g. desktop machines governed by organizations such as local government agencies (fire, law enforcement) and university/company administrative authorities. T-PNodes are few (compared to the total number of users) and are maintained by authorized entities at their local sites. Furthermore, T-PNodes are uniformly distributed across the global target geography.

The message of an event is tied to (intended for) a specific sub region inside the GTG. For example, the alerting message of a campus fire is tied to the campus and its proximity. This
region is called the **Possibly Affected Region (PAR)** of the message - for simplicity, we assume this region is rectangular as well and is denoted as $PAR$ where $PAR \subseteq GTG$. Crisis events (e.g. earthquakes, tornadoes) may damage the communication infrastructures inside the PAR and cause a geographically correlated regional failure. We define the regional geographical failure as the **Possibly Damaged Region (PDR)**, denoted as $PDR$; we assume $PDR \subseteq PAR$.

We consider the set of users inside a PAR as the **geographically correlated target recipients** of an event message, and the set is formulated as:

$$TR_g = \{u : u \in V, \text{Loc}(PN_u) \in PAR\}$$

As we mentioned earlier, the social friends of the geographically correlated target recipients are also possible recipients of the message. This set of users are called as the **socially correlated target recipients** and formulated as:

$$TR_s = \{v : v \in F(u), \forall u \in TR_g\}$$

Note that, unlike $TR_g$, users in $TR_s$ may be located outside the PAR.

### 4.2.3 Overlays of GSFord

Using the concepts delineated above, we develop two distinct overlays - **Delivery Overlay** and **Information Overlay** in GSFord. The delivery overlay aims to reach PNodes associated with a given region (e.g. a PAR) efficiently and effectively (section 4.3.2). On the other hand, the main purpose of the information overlay is to capture and maintain the geo-social information of participating users despite extreme damage (section 4.3.3). To enable this, the information overlay replicates the stored information by using the conjugate
region-based replication technique (section 4.3.3). Moreover, to mitigate security/privacy concerns, such as revealing personalizing information (e.g. social friends), the information overlay is constructed/maintained at only T-PNodes.

4.2.4 Event Notification over GSFord

The delivery and information overlays are created apriori given knowledge of the geography and social connections of individual users. When an event occurs, a new message related to PAR is created. The message is conveyed to the geographically correlated target recipients ($TR_g$) by the delivery overlay. To reach the socially correlated target recipients ($TR_s$), the information overlay explores the stored geo-social information to identify and locate them and forwards the message via the delivery overlay.

One of the key objectives of GSFord event notification is to reach users in the extremely damaged region affected by a disaster who are less likely to be reachable via the delivery overlay. For this purpose, GSFord encourages the socially correlated target recipients ($TR_s$) to initiate targeted social diffusion to propagate the received message further by customizing the contents of the forwarded message (section 4.4).

**Privacy Concern:** The reader might be concerned that the social information of a user provided to the GSFord system might be exploited to obtain the real world identity and connections of a user. We reduce the risk of such information disclosure in GSFord by (a) encoding user information into a public social ID using a one-way encryption technique at a trusted registration server and (b) storing and utilizing the encoded social graph and geo-social mappings at the more trusted PNodes during dissemination. In other words, information distributed at the T-PNodes cannot be used to reveal the real-world identities and connections. The registration server itself can be built using a trusted third-party, e.g. a public certification service such as Verisign[22].
4.3 GSFord Overlays: RRTree-based Geo-social Aware Overlays

In this section, we develop a peer-oriented geo-aware multicast overlay structure called RRTree (Responsible Region Tree). The RRTree-based geo-aware P2P overlay is a self-organizing and fault-tolerant overlay to support (a) efficient geographical regional multicasting, even with non-uniformly distributed PNodes, and (b) DHT-style reliable storage of social information under extreme geographically correlated regional failures.

Existing efforts in the design of geo-aware overlays do not explicitly take into account random or geographically correlated regional failures in the overlay construction process. For example, CAN-based approaches [163, 155] divide a region into multiple smaller regions; however, they do not consider the geographical relationship of a region to the physical nodes in that region. The most related existing effort is GeoPeer [31, 152]; here, a geo-aware overlay is constructed using a Delaunay triangulation method with Long Range Contacts (LRCs) to compensate for inefficient routing. In contrast to GeoPeer, the RRTree-based overlay enables explicit fault tolerance against random overlay failures as well as geographically correlated failures. Additionally, existing structures are not designed for geographical regional multicasting under non-uniform user distributions.

Below, we describe the responsible region concept, the formation of the RRTree structure and how this structure is used to construct and maintain the delivery and information overlays in GSFord.
4.3.1 RRTree-based Geo-aware P2P Overlay Structure

Given a global target geography \((GTG)\) and the PNodes in this geography, we aim to design an overlay structure which is dynamically adjusted to the unexpected distribution of PNodes and tolerant to the geo-correlated failures. We introduce the concept of a Responsible Region \((RR)\) that represents a rectangular region inside the \(GTG\) (see Figure 4.3) - the \(RR\) concept is used to build a hierarchically nested logical structure, the Responsible Region Tree \((RRTree)\) overlay structure. The hierarchical structure begins at the root \(RR\) \((RR_{root})\) that corresponds to the entire \(GTG\). Each \(RR\), beginning with \(RR_{root}\), is divided into \(N_c\) child \(RRs\) (illustrated with \(N_c = 2\) for simplicity) where each of the child \(RRs\) represents non-overlapping partitions that completely cover the corresponding parent \(RR\). The subdivision of a parent \(RR\) into child \(RRs\) terminates when the number of PNodes in the corresponding region meets a threshold. As illustrated in Figure 4.3, every PNode is associated with a unique leaf \(RR\) \((RR_{leaf})\) of the RRTree. The level of a \(RR\) is its depth in the RRTree structure (e.g. the level of \(RR_{root}\) is 0).

The RRTree structure inherently maintains the following property: The set of leaf \(RRs\) forms a non-overlapping partition of \(RR_{root}\). In other words, the region covered by the \(RR_{root}\), i.e. \(GTG\) can be obtained by a direct union of the regions covered by the leaf \(RRs\), i.e \(RR_{root} = \bigcup RR_{leaf}\). Furthermore, given any two adjacent levels in the RRTree, a parent
RR subsumes the regions covered by its child RRs.

The process of RRTree growth and RRTree shrinkage ensures that the above property is maintained. The RRTree grows by splitting a \( RR_{leaf} \). When a physical node, \( PN_u \), joins the RRTree-based overlay, it sends a join request to a previously known PNode in the RRTree (selection from a set of PNodes in the RRTree for bootstrapping); the request is then routed to the corresponding \( RR_{leaf} \). If the number of PNodes in the \( RR_{leaf} \) does not exceed a split-threshold as a result of the join, the new PNode is accommodated; if the number of PNodes associated with the \( RR_{leaf} \) becomes greater than the split-threshold, \( Th_s \), the \( RR_{leaf} \) splits into \( N_c \) smaller RRs, and the child RRs becomes leaf RRs. During the splitting process, PNodes may be associated with new corresponding leaf RRs based on their locations. We assume that a \( RR_{leaf} \) splits equally and alternately (e.g. horizontally and vertically) with \( N_c = 2 \) as illustrated in Figure4.3.

The RRTree shrinks by merging \( N_c \) leaf RRs having the same parent \( RR \). When the total number of PNodes in all of the leaf RRs, i.e. total number of PNodes under a parent \( RR \), falls below the merge-threshold, \( Th_m(< Th_s) \), the leaf RRs directly merge with the parent \( RR \), forming a new leaf \( RR \). Note that the PNodes corresponding to the original leaf RRs are now associated with the new leaf \( RR \) (i.e. original parent \( RR \)). Note that merging is not initiated at the intermediate RRs of the RRTree. This is easily proved; if a \( RR \) has a non-leaf child \( RR \), the number of PNodes accommodated by the \( RR \) is greater than \( Th_s \), which is greater than \( Th_m \).

Since the RRTree is a logical structure, information about an \( RR \) is basically stored at multiple PNodes located inside the \( RR \); we call these PNodes **Struct Nodes** to indicate that they are structure maintenance PNodes. In order to sustain both random overlay failures and geographically correlated regional overlay failures, there are at least \( Th_{str} \) struct nodes of a \( RR \) and they are sparsely distributed over the \( RR \). If the locations of PNodes are non-uniformly distributed, a \( RR \) may not have a sufficient number of struct nodes located
inside the RR. In this case, we temporarily use PNodes outside the RR, for example PNode C in Figure 4.3 is used as a Temporary Struct Node. The temporary struct nodes are observed only in a leaf RR that does not have sufficient number of PNodes, and they should be immediately replaced with the newly joined PNodes in the RR. To minimize the number of temporary struct nodes, we use the following constraints on threshold values - 1) \( Th_s \approx N_c \times Th_{str} \) and 2) \( Th_m \approx 0.5 \times N_c \times Th_{str} \). These constraints are especially useful when PNodes are uniformly distributed over GTG. The reliability of a RR can be described as \( 1 - (P_f)^{Th_{str}} \), where \( P_f \) is the probability of failure of a PNode.

Because of the concept of the struct nodes, a PNode may be responsible for storing the information of multiple RRs. We call a RR whose information is stored at a PNode as a backup RR. The number of backup RRs of a PNode depends mainly on its level in the RRTree. Since higher level RR is always covered by the lower level RR, a level \( L \) PNode may store up to \( L + 1 \) RRs which cover the PNode. According to this, the estimated number of backup RRs of a level \( L \) PNode is the summation of the estimation that each of the \( L + 1 \) RRs becomes the backup RR of the PNode. Since selecting a struct node for a level \( l \) RR resembles a uniform random walk starting from the level \( l \) RR to a PNode through the RRTree, the probability that a PNode becomes a struct node of a level \( l \) RR \( (P_l) \) can be calculated by multiplying probabilities of a random walk at each level until it meets the PNode. According to this, \( P_l \) can be represented as \( \frac{1}{n_l} \), where \( n_l \) is the number of PNodes in the level \( l \) RR. The total estimated number of backup RRs of a level \( L \) PNode \( (|N_B|) \) can be calculated like the equation 4.1.

\[
|N_B| = Th_{str} \times P_L + Th_{str} \times P_{L-1} + \cdots + Th_{str} \times P_0
\]  \hspace{1cm} (4.1)

\[
= Th_{str} \times \sum_{l=0}^{L} P_l
\]  \hspace{1cm} (4.2)
If the locations of PNodes are uniformly distributed, \( n_l \) can be considered as \( \frac{M}{N_c} \) where \( M \) is the total number of PNodes in \( GTG \), and the equation 4.1 can be represented like the equation 4.3.

\[
|N_B| = Th_{str} \times \left( \frac{1}{n_L} + \frac{1}{n_{L-1}} + \cdots + \frac{1}{n_0} \right) \tag{4.3}
\]

\[
= Th_{str} \times \left( \frac{N_c^L}{M} + \frac{N_c^{L-1}}{M} + \cdots + \frac{N_c^0}{M} \right) \tag{4.4}
\]

\[
= Th_{str} \times \frac{1}{M} \times \left( \frac{N_c^L - 1}{N_c - 1} + N_c^L \right) \tag{4.5}
\]

In the equation 4.3, \( \frac{N_c^{L-1}}{N_c - 1} \) is always less than or equal to \( N_c^L \) if \( N_c \) is a positive integer value. Additionally, since \( P_l \) is less than or equal to 1, \( \frac{N_l}{M} \) is also less than or equal to 1. According to these, \(|N_B|\), the estimated number of backup nodes of a PNode is always less than \( 2 \times Th_{str} \) like following equation:

\[
|N_B| \leq Th_{str} \times \frac{1}{M} \times 2N_c^L \leq Th_{str} \times 2 \times \frac{N_c^L}{M} \leq 2 \times Th_{str} \tag{4.6}
\]

**Basic Routing in RRTree and Region Hopping Table**

We first describe the basic unicast routing process in the RRTree; the following section describes the regional multicasting process. Unicast routing related messages in the RRTree are typically initiated at leaf \( RRs \); they may be join requests from new PNodes or content routing requests from one PNode to a location. We explain the flow of the unicast routing process using a join request. A join request (that contains a new PNode and its current location as the target location) initiated from a leaf \( RR \) must be forwarded to the corresponding
leaf $RR$ of the target location by following the RRTree structure. If the initiating leaf $RR$ covers the target location, the PNode joins the leaf $RR$ following the join rules described earlier; else the join request is forwarded to the parent $RR$ of the leaf $RR$. If the parent $RR$ covers the target location, it forwards the join request to the corresponding child $RR$ which covers the target location. Otherwise, it forwards the message to its parent $RR$ and the process is repeated. The routing process in a $RR$ accomplished by the struct nodes in the $RR$. The routing process continues until the join request meets the target leaf $RR$. Content routing follows the same process - the message with a target location is generated at a leaf $RR$, ascends the RRTree, and descends to the corresponding target leaf $RR$.

We observe that when routing occurs by ascending/descending the RRTree, multiple hops are required to reach a horizontally close $RR$; this can cause bottlenecks at struct nodes higher up in the RRTree closer to $RR_{root}$. To alleviate such overload, we introduce the concept of a Region Hopping Table (RHT), maintained at each PNode, that allows messages to hop to other non-overlapping regions without ascending/descending the RRTree.

Given a PNode $PN_u$ accommodated in a level $L$ leaf $RR$ in the RRTree, the RHT of $PN_u$ is a $L \times N_c$ table, where $N_c$ is the given number of child $RR$ of a $RR$. Each RHT entry ($RTE$) in the $l_{th}$ RHT row ($r = 1, 2, \cdots, L$) contains links to each of $N_c$ $RR$s at level $l$. The first $RTE$ in the $l_{th}$ row corresponds to the level $l$ $RR$ that subsumes the location of $PN_u$, $Loc(PN_u)$, and denoted as a self-RTE ($SRTE$) which indicates the loopback link to $PN_u$. The other $RTE$ in the $l_{th}$ row corresponds to $N_c - 1$ sibling $RR$s of the $RR$ related to the first $RTE$, which do not subsume $Loc(PN_u)$. For example, Figure 4.3 shows the RHT of the PNode E ($L = 2, N_c = 2$) which is a $2 \times 2$ table where the first row contains links to level 1 $RR$s and the second row contains links to level 2 $RR$s. A RHT entry, except $SRTE$s, contains randomly chosen contact PNodes residing in the corresponding $RR$, as the links to the $RR$ - this can be obtained during maintenance or routing.

Unlike the RRTree, which only has knowledge of its immediate neighbors (parent and child
RRs), the RHT of a PNode covers the GTG. That is, \( RR_{root} = \bigcup RR_{RTE}, \forall RTE \neq SRT \in RHT \), where \( RR_{RTE} \) is the \( RR \) corresponding to the \( RTE \). If PNodes are uniformly distributed, a RHT of a PNode has an average of \( \log_{N_c} N \) rows, where \( N \) is the total number of PNodes. According to this, a point-to-point routing with the RHT takes an average of \( O(\log_{N_c} N) \) hops.

4.3.2 Delivery Overlay: Efficient Regional Multicasting

The delivery overlay (DOv) is composed of PNodes and constructed by using the RRTree-based overlay construction method. The purpose of the delivery overlay of GSFord is supporting efficient and effective geographical regional multicasting to all of the PNodes inside a given region. To start a geographical regional multicasting of a message, a PNode initiates a message in the format of \( M = [T, CT, MP] \), where \( T \) represents eventual target region of the message, \( CT(\in T) \) indicates the target region for the next immediate forwarding, i.e. where the current PNode wishes to forward the message and \( MP \) is the message payload. Initially, \( CT = T \); GSFord realizes the efficient regional multicasting by resending the original message with modified \( CT \) at each propagation. Note that the aim of the regional multicasting is that the message must be eventually forwarded to all of the PNodes corresponding to \( T \). As is typical of geographical routing protocols, in order to forward a message to a unique PNode \( PN_u \), \( T \) must be set to \( Loc(PN_u) \).

Figure 4.4 shows the pseudo code of the geographical regional multicasting with the RHT. When a PNode receives a message, it first checks that the message needs to be forwarded (\( M.terminatedFlag == NULL \)). Then, it finds out the subsequent PNodes which need to convey the message by selecting those routing table entries, \( RTE \) in its RHT whose corresponding \( RR \) partially overlaps with \( CT \). Before forwarding the message to a next PNode of a \( RTE \), \( CT \) is updated to cover the region of overlap between the original \( CT \)
in the message and the \textit{RR} of the \textit{RTE}; this is done to prevent in unnecessary message forwarding. The multicasting continues until a leaf \textit{RR} is encountered which subsumes \textit{CT}. Once the message arrives the target leaf \textit{RR}, the message marked with the termination flag and forwarded to all of the PNodes located inside \textit{CT}.

However, while RHT supports efficient geographical regional multicasting, information in the RHT can become stale under overlay failures. As a result, we may not reach contact PNodes of a \textit{RTE}; furthermore the stale information can cause loops in the routing. The unreachable PNodes are easily detected when trying to forward a message. To detect a routing loop, the message needs to piggyback the previous routing path. Whenever the stale RHT information is detected, the message is forwarded through RRTree (see Figure 4.5) and the RHT replaces the stale contact PNodes with the newly obtained information from RRTree.

Figure 4.5 shows the pseudo code for the geographical regional multicasting protocol using the RRTree. Since a PNode may act as the struct node for multiple \textit{RR}s (denoted as RRList in Figure 4.5), it first forwards the message to all of the \textit{RR}s which overlap with \textit{CT}, and updates \textit{CT} by excluding the overlapped portion from the original \textit{CT}. Note that \textit{CT} is not
updated to the overlapped portion before forwarding the message. If there are no RR that overlaps with CT, the message is forwarded to the parent RR of the largest RR known by a PNode, and the multicasting process is terminated. In the delivery overlay, while the multicasting with RRTree is only used for the fail over purpose, a PNode mainly uses the regional multicasting with RHT to forward a message.

4.3.3 Information Overlay: Social Information Storage

GSFord not only supports efficient geographic regional multicasting through the delivery overlay, but also captures the social information of users in the information overlay (IOv). While the delivery overlay sends a message to the set of the geographically correlated target recipients (TRg), the information overlay determines the set of the socially correlated target recipients (TRs) for a given geographical region. To do this, the information overlay stores (a) the social information of PNodes inside each leaf RR of the delivery overlay including SIDs of social entities (u) of the PNodes (PNu), degree centrality CD(u) and the list of SIDs of social friends (F(u)), and (b) the geo-social mapping which is the translation mapping
between a \( SID \) and the location of the corresponding PNode. \( TR_s \) can be obtained in the form of a set of \( SID \)s from the social information and the given target geographical region, and the delivery overlay conveys the message to \( TR_s \) by querying the geo-social mapping of each \( SID \).

As discussed earlier, social network information of individuals is considered sensitive data and release of this content to arbitrary PNodes (i.e. associated social entities) may cause privacy concerns. To alleviate this, the information overlay is constructed and maintained at T-PNodes rather than over all PNodes. That is, T-PNodes only reply to queries of the social information initiated by other T-PNodes.

The information overlay consists of T-PNodes and is managed using the RRTree-based overlay structure. As with the delivery overlay, \( RR \)s in the information overlay correspond to geographic regions (usually distinct from the regions of the delivery overlay). The root \( RR \) of the information overlay, \( RR^{IOv}_{root} \), is the same as the root \( RR \) of the delivery overlay, \( RR^{DOv}_{root} \). A leaf \( RR \) in the information overlay, \( RR^{IOv}_{leaf} \), is associated with the T-PNodes whose locations are inside the \( RR^{IOv}_{leaf} \). When a new PNode, \( PN_u \), joins the delivery overlay, it finds out the corresponding \( RR^{IOv}_{leaf} \) that subsumes the location of \( PN_u \). \( PN_u \) provides its social information, \( [PN_u, Loc(PN_u), SID_u, C_D(u), \{SID_v : v \in F(u)\}] \) to the T-PNodes associated with the \( RR^{IOv}_{leaf} \). Eventually, the TPNodes (i.e. \( IOv \)) store the social information of all current online PNodes in GSFord.

### RegionID Assignment for Storing Geo-social Mapping

To reaching the socially correlated target recipients of a message, we need to determine the location of the selected social entities using a geo-social mapping process. The geo-social mapping, \( [SID_u, Loc(PN_u)] \), is indexed by \( SID_u \).

We define the concept of a region ID (\( RID \)) to each \( RR \) in the information overlay, \( RR^{IOv} \).
- the $RID$ is a unique identifier (bit vector) for each $RR^{IOv}$ and has a fixed number of bits (160 in our case). A $RID$ assigned dynamically during splitting/merging of a $RR^{IOv}$, as illustrated in Figure 4.6(a) to guarantee that the ID ranges covered by leaf $RRs^{IOv}$ are non-overlapping and these ranges together cover the entire ID space. The $RID$ of $RR_{root}^{IOv}$ is $\ast$, a *wildcard mask symbol*, which means it matches any values in the whole ID space. If a $RR^{IOv}$ splits into two child $RR^{IOv}s$, its $RID$ splits into two distinct sub $RIDs$ for each of the child $RR^{IOv}s$. For example, when $RR_{root}^{IOv}$ splits, the two sub $RIDs$ are $0\ast$ and $1\ast$. If $RR_{root}^{IOv}$ splits horizontally, the upper child $RR^{IOv}$ takes $0\ast$ and the lower child $RR^{IOv}$ takes $1\ast$. If $RR_{root}^{IOv}$ splits vertically, the left child $RR^{IOv}$ and the right child $RR^{IOv}$ take $0\ast$ and $1\ast$, respectively.

We enable a DHT-style mapping of $SIDs$ to $RIDs$; here, a $SID$ is hashed (using the SHA-1 hash function) to yield a bit vector, $Hash(SID)$, of the same length as $RID$. The geo-social mapping of a $SID$ is stored in the leaf $RR^{IOv}$ whose $RID$ has the best prefix-match (using bitwise comparison) to $Hash(SID)$; specifically the geo-social mapping is stored in the T-PNode corresponding to the selected leaf $RR^{IOv}$. The SHA-1 hash function generates uniformly distributed hashed value w.h.p., and the storage/query load of the geo-social mapping over the entire information overlay can be balanced. To find the leaf $RR^{IOv}$ corresponding to $SID_u$, we can conduct the RHT/RRTree routing process with slight modified routing tables as illustrated in Figure 4.6(a). That is, during the routing process, $RID$ is used to find next
To cope with the information losses and inaccessible networks due to extreme geographically correlated regional failures, we replicate the social information stored in a leaf $RR^{IOv}$. For this purpose, we introduce the notion of **conjugate region** ($RC$) of a region ($R$), which is a geographically distant region to $R$, less likely to be impacted by the regional failure related to $R$. The conjugate region concept is inspired by the popular “out-of-state relative” concept used for emergency contacts; here two users located in a disaster region communicate through a pre-established “geographically distant” out of state relative over the highly congested telephone network (caused by a surge of incoming calls in the disaster region).

Let $R$ denote a region in $GTG$, we define a function $f_C : R \rightarrow RC$. This mapping between $R$ and $RC$ should satisfy the following properties: (a) the mapping is known apriori by GSFord and easily maintained, (b) $RC$ should be less likely to fail simultaneously with $R$ under a geographical regional failure, and (c) the geographical distance of any $R, RC$ pair should be similar in order to ensure fairness of information reliability. According to these properties, if a leaf $RR$ of the information overlay replicates the stored information in the conjugate leaf $RC$, the information overlay of GSFord can maintain the social information and geo-social mappings reliably even under large geographically correlated regional failures.

To satisfy the desired properties of conjugate region, we come up with a simple mapping function as

$$f_C(x, y) = (x_C, y_C) = ((x + x_{max}/2)\%x_{max}, (y + y_{max}/2)\%y_{max})$$

The function generates the conjugate coordinate ($x_C, y_C$) from a source coordinate ($x, y$).
on GTG of GSFord. By using \( f_C \), a region \(((x_1, y_1)(x_2, y_2))\) can find its conjugate region \(((x_{1C}, y_{1C})(x_{2C}, y_{2C}))\), which might be a wraparound region in GTG as illustrated in Figure 4.6(b). According to \( f_C \), we can virtually draw two conjugate axes, and these two axes divide GTG into four quadrant virtual regions. The top-left region and the top-right region are matched to the bottom-right region and the bottom-left region as the conjugate region, respectively. The concept of the conjugate region can also be interpreted using the region ID concept as illustrated in Figure 4.6(a). Since each bit of a region ID divides the region ID space into two sub regions, we can use two bits to divide GTG into four quadrant virtual regions. The conjugate region ID of a region ID is obtained by conducting XOR bit operation of first 2 bits of the region ID.

Lemma 4.1. Let PDR be a failed region inside GTG, and PDR\(_C\) be the conjugate region of PDR. The information overlay of GSFord guarantees that it can retrieve both of the geo-social mapping and the social information, stored in the leaf RR\(_s\) subjected to PDR, from the T-PNodes corresponding to PDR\(_C\) with the lower bound of the probability, \((1 - \left( \frac{\text{Size}(\text{PDR})}{\text{Size}(\text{GTG})} \right)^{\text{Th}_{\text{str}}} \times (1 - \left( \frac{2 \cdot \text{Size}(\text{PDR})}{\text{Size}(\text{GTG})} \right)^{\text{Th}_{\text{str}}}),\) only if (1) the \(\text{Size}(\text{PDR}) < \frac{1}{4} \cdot \text{Size}(\text{GTG})\) and (2) PDR is overlapped less or equal than two quadrant regions of GTG.

Proof. According to the conjugate function \( f_C(x, y), \ PDR \cap PDR_C = 0, \) only if the size of PDR is less than a quarter of the size of the GTG. That is, with this condition, we can guarantee that PDR\(_C\) is not affected by the geographical regional failure on PDR. However, the regional failure may affect the reliability of the RRTree structure by causing a failure of the struct nodes of a RR. However, with the second condition and \( f_C(x, y), \) the regional failure may affect only the top two levels of RRTree structure (RR\(_\text{root}\) and level 1 RR\(_s\)). Since the struct nodes of a RR are selected uniformly at random, the probability of failure of a struct node of RR\(_\text{root}\) and a level 1 RR is \(\left( \frac{\text{Size}(\text{PDR})}{\text{Size}(\text{GTG})} \right)^{\text{Th}_{\text{str}}} \) and \(\left( \frac{2 \cdot \text{Size}(\text{PDR})}{\text{Size}(\text{GTG})} \right)^{\text{Th}_{\text{str}}} \), respectively. That is, the probability of retrieving the information successfully from PDR\(_C\) is \(\left( 1 - \left( \frac{\text{Size}(\text{PDR})}{\text{Size}(\text{GTG})} \right)^{\text{Th}_{\text{str}}} \right) \times (1 - \left( \frac{2 \cdot \text{Size}(\text{PDR})}{\text{Size}(\text{GTG})} \right)^{\text{Th}_{\text{str}}} ).\) Moreover, since each PNode has a distinct
RHT to reach $PDR_C$, the above probability becomes the lower bound.

4.4 Event Notification

Key intuitions behind efficient event notification in GSFord include (a) reaching recipients in geographically correlated target recipients, $TR_g$, who can propagate the message further via the delivery overlay and (b) leveraging reachable recipients in socially correlated target recipients, $TR_s$, to forward the message to recipients in the $PDR$ through social diffusion over diverse out-of-band communication channels. Moreover, the observations about the social diffusion process suggest customizing the social diffusion process by (a) selecting good initiators and (b) modifying content of the messages, can help trigger more accurate and enable faster targeted social diffusion.

The overall of notification process of GSFord is illustrated in Fig. 4.7. The Three distinct steps include (a) reaching $TR_g$ using the delivery overlay; (b) obtaining $TR_s$ from the in-
formation overlay; and (c) message content customization and delivery of the customized messages to PNodes in $TR_s$.

Reach geographically correlated target recipients: Let us assume that an event message $M = [PAR, CT, MP]$ is generated by an authorized source (e.g. USGS, local government) communicated to a T-PNode in GSFord. This initial T-PNode starts the geographical regional multicasting of $M$ to reach $TR_g$ by using the delivery overlay.

Obtain socially correlated target recipients: At the same time, in order to reach $TR_s$, the initial T-PNode generates a conjugate message of $M$, $M_C = [PAR_C, CT, MP]$, where $PAR_C$ is the conjugate region of $PAR$, and forwards $M_C$ to T-PNodes corresponding to $PAR_C$ via the information overlay. Upon receiving $M_C$, each of the T-PNodes in $PAR_C$ performs a lookup to determine socially correlated recipients ($TR_s$) of the message in the form of $SIDs$. Their corresponding PNodes of $TR_s$ are obtained by querying the geo-social mapping of the obtained $SIDs$.

Customize message content: After determining $TR_s$ and their locations, T-PNodes may modify the original content of the message asking users in $TR_s$ to contact social friends in the $PAR/PDR$. We develop 3 levels of customization with increasing degree of complexity: coarse, PAR-targeted, and PDR-targeted. As the complexity of the customization increases, triggering the more targeted and more aggressive social diffusion is expected.

**COARSE customization**: The intuition here is that the high centrality users can potentially diffuse the received message to a larger number of users in the $PAR$. To do this, T-PNodes pick top-$K$ users in $PAR$ with highest degree centrality. The customized message $MP_u$ for a user $u$ is $MP_u = MP + \{SID_v\}$ where $v$ are $u$’s social friends among the top-$K$ users in $PAR$, i.e. $v \in F(u) \cap TK_{CD}(PAR)$.

**PAR-TARGETED customization**: The PAR-targeted customization indicates the social friends to be contacted more specifically. That is, the customized message is $MP_u =$
$MP + \{SID_v\}$ where $v$ are $u$’s social friends located in $PAR$, i.e. $v \in F(u), PN_v \in PAR$.

**PDR-TARGETED customization**: In the case that $PDR$ of an event is correctly defined, T-PNodes can customize the message much more precisely. The customized message is $MP_u = MP + \{SID_v\} + FLAG_{PDR}$ where $v$ are $u$’s social friends located in $PDR$, i.e. $v \in F(u), PN_v \in PDR$. $FLAG_{PDR}$ indicates that $MP_u$ needs to be handled as an urgent message in order to encourage users to contact the social friends in $PDR$ by all means as soon as possible.

After customization, the individually modified message, $MP_u$, is forwarded to the PNode of a user $u$ through the delivery overlay. Once the GSFord client on the PNode receives the message, it translates the list of SIDs into human-readable form (e.g. names of friends) based on its local information. If the customized message has the flag of $FLAG_{PDR}$, the GSFord client phrases the message content as urgent.

### 4.5 Evaluation

To gain better understanding of the ability of GSFord to disseminate messages to relevant recipients over a given geography efficiently and reliably, even under geographical correlated regional failures, we evaluated GSFord along multiple dimensions through implementation and extensive simulations. The key dimensions that served as metrics for our study include (a) improvements in reachability/coverage of a message, typically expressed as around 15% of increase, (b) efficiency and reliability of the delivery and information overlays of GSFord under different user distributions, and (c) scalability of the delivery/information overlay of GSFord.
4.5.1 Evaluation Settings

To emulate both of the propagation of messages over the geography-aware delivery overlay of GSFord and the social diffusion of messages over a social network graph, we mapped a social network graph into the global target geography, GTG. We obtained a social network graph, \( G := (V, E) \), of 50K nodes \( V \) and 880K edges \( E \) by sampling the online social network graph crawled from Facebook [144]. We believe that the graph derived from an online social network serves as a reasonable representation of a real-life social network. The social friends \( F(u) \) of a social entity \( u \) are randomly drawn from the neighbors in the social graph \( G \) and the other neighbors are set to acquaintances \( A(u) \). In this evaluation, the number of social friends follows a normal distribution with \( \mu = 4 \) and \( \sigma = 1 \).

We set GTG as a 131K by 131Kmeters square region and distribute all of the nodes, \( u \in V \), on GTG. We used two types of node distributions: uniformly random distribution and non-uniform distribution. For the case of non-uniform distribution, we use a truncated Gaussian distribution with \( \mu = 45\text{Kmeters} \) and \( \sigma^2 = 45\text{Kmeters} \) to generate each \( x \) and \( y \) coordinate of a node. This setting mimics the demographics of the Southern California region including Los Angeles County, Orange County and Riverside County[6]. Each \( u \in V \) runs a PNode on the given location and PNodes form the delivery overlay of GSFord. To construct the information overlay of GSFord, we assume that there are a given number of T-PNodes. In this evaluation, we use 100 T-PNodes distributed uniformly at random over GTG.

For event notification, we assume that a randomly chosen T-PNode generates a message with a given target PAR. To simulate an extreme disaster causing geographically correlated regional failures, we defined a damaged region, PDR, which is co-centered and inside the PAR, and all of PNodes in the PDR remain disconnected from the other PNodes/T-PNodes. \( PDR_r \) is the ratio of the size of the PDR to the size of the PAR, and ranges from \([0, 1]\).

We simulated the dissemination of messages over the delivery overlay of GSFord with several
parameters such as link latency, bandwidth availability and context switch overhead obtained from a network emulator, Modelnet[16] with a network topology generated by Inet[14] to accurately mimic Internet scale parameters. We obtained these parameters by running the GSFord implementation on ModelNet cluster (section 4.5.4). The emulation yielded processing delays which varied from 3ms to 6ms, for a PNode to send a message to another PNode. We obtained inter-PNode link latencies ranging from 80ms to 150ms, with the average latency being 100ms. Moreover, we set bandwidth constraints ranging 100Kbps to 500Kbps and network packet loss rates of 1% to 5% for each link.

Since the order of latency of the social diffusion over a phone or an email communication channel is substantially longer than the latency of the dissemination over the delivery overlay of GSFord, we separately modeled the social diffusion process using an Independent Cascade (IC) model[91, 42]. In the IC model, the diffusion process unfolds in discrete time-steps. It assumes that nodes can switch from being inactive, interpreted as being unaware of the message, to being active, which is being aware of the message. When a node $u$ becomes active in step $t$, it is given a single chance to activate each currently inactive neighbor $v$ with a probability $P(u, v)$ independent of the history thus far.

To take into account the disparate levels of interest, social entities may have, to specific messages and the likelihood of communication between neighbors, we enhanced the IC model. Specifically, we exploited multiple probabilities based on the relationship of neighbors and their interest in forwarding the messages. This comes from our observations of how real world social diffusion occurs: (a) the diffusion probability between social friends is much higher than acquaintances, and (a) the diffusion probability of relevant targeted messages that request specific actions is higher than generic messages. To realize this, we use $P_a(u, v)$, $P_f(u, v)$, $P_{f_{par}}(u, v)$, and $P_{f_{pdr}}(u, v)$ as a diffusion probability of a social link $e_{u,v}$ for acquaintances, social friends, social friends with PAR messages, and social friends with PDR messages, respectively. Each of $P_a(u, v)$, $P_f(u, v)$, $P_{f_{par}}(u, v)$, and $P_{f_{pdr}}(u, v)$ is generated by using
Gaussian distribution with the mean, 0.2, 0.4, 0.6 and 0.8, respectively.

We also enhanced the IC model to consider the latency of delivering messages over multiple communication channels. We considered two types of communication channels: an Email channel ($\text{email}_{u,v}$) and a Phone channel ($\text{phone}_{u,v}$). The delay of $\text{email}_{u,v}$ is generated by the exponential distribution with $\mu = 200$ minutes and the minimum delay sets to 15 minutes[144]. Unlike email, the delay of $\text{phone}_{u,v}$ is more bounded and we generate it with the Gaussian distribution with $\mu = 15$ minutes between 10 to 20 minutes. The preference of communication channels depends on both the interests of the message and the relationship of neighbors. If a node tries to forward the more interest messages to friends, the likelihood of using the phone channel increases.

To understand the impact of social diffusion and the information overlay, we compared GSFord variations under different combinations of geo-aware overlays and social diffusion in section 4.5.2. GeoOverlay models geo-aware overlays[31, 152]; we use the delivery overlay of GSFord as a representative. GeoOverlay+SD refers to the case where the delivery overlay is combined with a basic social diffusion process (i.e. without the information overlay). We also compare the value of broadcasting to the entire tree - GeoOverlay+B+SD represents the case where a basic social diffusion follows a broadcasting step, where a message is broadcasted over the delivery overlay to the entire GTG. In general, such broadcast can be realized using any application layer multicast as a broadcast method[87, 37]. GSFord variations are GSFord+COARSE, GSFord+PAR, and GSFord+PDR which are using coarse, PAR-targeted, and PDR-targeted customization method, respectively.

In section 4.5.3, we compare the RRTree-based geo-aware delivery overlay of GSFord, denoted as GS with previous geo-aware overlays. In particular, we simulated the Delaunay triangulation based geo-aware P2P overlay, GeoPeer [31, 152], denoted as GP.
4.5.2 Reachability of GSFord

![Diagram of reachability comparison](image1)

(a) $PDR_r = 0.5$.

![Diagram of reachability comparison](image2)

(b) $PDR_r = 0.8$.

Figure 4.8: Reachability comparison of GSFord with other systems.

![Diagram of reachability comparison](image3)

(a) Eventual reachability.

![Diagram of reachability comparison](image4)

(b) Message Overhead of Social Diffusion.

Figure 4.9: Performance comparison of GSFord with other systems.

We first evaluate the reachability of GSFord to reach the geographically correlated target recipients, $TR_g$, of a random $PAR$ under a random $PDR$ - the damage is caused by a geographically correlated regional failure. The reachability is defined as the ratio of number of social entities in $TR_g$ receiving the message to the size of $TR_g$. Note that we do not consider the social entities outside $PAR$ for calculating the reachability, because they most likely receive the message reliably via the geo-aware delivery overlay. We set $PAR$ to be a 16K by 16Kmeter rectangle centered at a random coordinate inside the global target region,
<table>
<thead>
<tr>
<th>User Distribution</th>
<th>Max. Latency (msec)</th>
<th>Avg. Latency (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GSFord</td>
<td>GeoPeer</td>
</tr>
<tr>
<td>Uniform</td>
<td>1924.4</td>
<td>1609.4</td>
</tr>
<tr>
<td>Non-Uniform (Crowded)</td>
<td>1934.6</td>
<td>1881.8</td>
</tr>
<tr>
<td>Non-Uniform (Non-Crowded)</td>
<td>1455.8</td>
<td>1594.5</td>
</tr>
<tr>
<td></td>
<td>GSFord</td>
<td>GeoPeer</td>
</tr>
<tr>
<td>Uniform</td>
<td>1093.4</td>
<td>1309.7</td>
</tr>
<tr>
<td>Non-Uniform (Crowded)</td>
<td>1038.4</td>
<td>1497.8</td>
</tr>
<tr>
<td>Non-Uniform (Non-Crowded)</td>
<td>907.2</td>
<td>1242.9</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison between GSFord and GeoPeer.

$GTG$. Experimental results indicate that the reachability of uniform and non-uniform user distributions is very similar; we show the results of the uniform user distribution to conserve space.

In Figure 4.8, we show the reachability and the delay of the event notification process of different systems with different sizes of regional failures. Figure 4.8(a) and 4.8(b) show the reachability as a function of time when $PDR_r$ is set to 0.5 and 0.8, respectively. We observe that social diffusion can aid the dissemination of a message significantly when the geo-aware overlay is subject to a regional failure, i.e., where reachability of $GeoOverlay$ is bounded by $1 - PDR_r$. Note that since there is a significant time difference in $GeoOverlay$ based dissemination (Internet latencies) vs. the social diffusion process, $GeoOverlay$ results appear constant. More detailed performance of $GeoOverlay$ is presented in section 4.5.3.

While social diffusion can improve message reachability, GSFord can achieve much faster dissemination with even better reachability by leveraging the social information of the target recipients in the $PAR$ retrieved from the information overlay. The performance of basic social diffusion ($GeoOverlay+SD$) is significantly affected by $PDR_r$ in terms of the dissemination time as well as the reachability. Note especially that when $PDR_r = 0.8$, the reachability continues to be less than 0.8 (even 5 hours after the initial event message is disseminated). Using a broadcasting technique ($GeoOverlay+B+SD$) can expedite the speed of the basic
social diffusion; this is however achieved at the cost of a huge number of irrelevant messages that reach all nodes in the delivery overlay. On the other hand, GSFord (with targeted customization with $PAR/PDR$) achieves higher reachability than $GeoOverlay+B+SD$, and reach over 0.8 of target recipients within around 30 minutes after the initial message dissemination, when $PDR_r = 0.8$.

For better understanding of reachability and overhead of each dissemination system under different $PDR_r$, Fig. 4.9 presents the eventual reachability and the average number of messages that a recipient inside the $PAR$ received through the social diffusion process. We observed that leaning to the basic social diffusion is not a good idea to increase the eventual reachability. Employing broadcasting technique ($GeoOverlay+B+SD$) expedites the social diffusion process. However, it achieves only similar reachability of $GSFord+COARSE$ despite spending more overhead. On the other hand, GSFord incurs more overhead of social diffusion adaptively to $PDR_r$ and achieves higher reachability. Especially, $GSFord+PDr$ achieves almost 1 of reachability under any $PDR_r$ by aggressively encouraging social diffusion process.

In the series of results, we noticed that the performance of GSFord can be tuned by altering customization methods. $GSFord+COARSE$ takes less overhead, but slow dissemination speed and relatively low reachability. When GSFord uses targeted customization method with $PAR/PDR$, it spends more overhead to achieve high reachability and fast dissemination speed. According to this, GSFord system can categorize the message into normal, medium and urgent based on the importance of the message, and use the proper customization method for each type of message. That is, normal, medium and urgent message is customized with coarse, PAR-targeted, and PDR-targeted method, respectively.

In Figure 4.9(b), we show the average number of messages that a recipient inside the $PAR$ received through the social diffusion process. While the basic social diffusion ($GeoOverlay+SD$) takes around 3 messages, the broadcasting case ($GeoOverlay+B+SD$) takes around 4 messages to improve the reachability and the speed. However, we observed that the mes-
The average latency of GSFord lies between the basic social diffusion and the broadcasting cases. That is, GSFord can achieve higher reachability with reasonable message overhead of social diffusion. Also, we observed that GSFord with knowledge of the \( PDR \) (\( GSFord+PDR \)) can adaptively initiate a customized social diffusion to achieve very high reachability with reasonable overhead. More specifically, we observe that \( GSFord+PDR \) takes around 1 more message under \( PDR_r = 1.0 \) than both \( GSFord \) and \( GeoOverlay+B+SD \).
4.5.3 Performance of RRTree-based Geo-aware Overlay

In this section, we compare the performance of the RRTree-based geo-aware delivery overlay of GSFord with GeoPeer, a Delaunay triangulation based geo-aware overlay[31]. Both protocols, the delivery overlay of GSFord and GeoPeer, consider the locations of nodes as an identifier of the overlay that is used for routing purposes. In GeoPeer, a node has two types of contacts: direct neighbors obtained from the Delaunay triangulation with the locations of nodes and long ranged contacts (LRC) for improving routing performance. For a regional multicasting, first GeoPeer forwards a message to the node nearest to the center of the target region, and the node floods the message in the scope of the related region. During the regional multicasting, each node refers to both types of contacts [31].

Figure 4.10 shows the reachability of the delivery overlay of GSFord and GeoPeer as a function of time under different user distributions. We also consider the impact of the geographical failure ($PDR_r = 0.5$). We observed that the performance of the delivery overlay of GSFord is stable under different user distributions. On the other hand, GeoPeer takes higher average delivery times if users are distributed non-uniformly. We also note that the delivery overlay of GSFord can deliver messages faster than GeoPeer in the presence of a geographical regional failure. That is, the RRTree based geo-aware overlay can support efficient and reliable geographical regional multicasting with both unexpected user distributions and random geographically correlated regional failures. The main reason is that the RRTree based overlay dynamically adjusts both RRTree and RHT of a PNode based on the population of the region corresponding to the PNode.

Table 4.1 shows more detail comparisons between the delivery overlay of GSFord and GeoPeer in terms of latency, reachability and message overhead. In many cases, the delivery overlay of GSFord exhibits lower average latency and much lower message overheads than GeoPeer. Since GSFord uses around 6 times fewer messages, we observed that GSFord may cause a
slight reduction in reachability than GeoPeer. But, while the delivery overlay of GSFord sacrifices 2.7% of reachability for the crowded region, GeoPeer also loses around 1.4% of reachability for the non-crowded region. This is because the fault tolerance of GeoPeer simply relies on the density of nodes without any explicit fault tolerance against node failures. That is, if the population of region becomes low, GeoPeer may lose fault tolerance under regional failures. The slight loss of reachability of the delivery overlay of GSFord is eventually compensated for by the effective social diffusion triggered by the information overlay.

To evaluate the scalability of GSFord, we measured the average latency and message overhead of the geographical regional multicasting under different number of nodes as illustrated in Figure 4.11. In Figure 4.11(a), the average latency of the delivery overlay of GSFord increases logarithmically along with the number of nodes; we also observed that the average latency of GSFord is less than GeoPeer. Figure 4.11(b) shows the message overhead, defined as the ratio of total number of messages used for the regional multicasting to the number of target recipients, as a function of the total number of nodes. We observed that GeoPeer uses fewer messages than GSFord when the number of total nodes is small. But, as the number of total nodes increases, while the message overhead per a target recipient converges to 1 in the delivery overlay of GSFord, the message overhead of GeoPeer increases significantly. In the comparison with GeoPeer, GSFord achieves less average latency and much lower overheads, especially with a large number of nodes.

Another metric related to the scalability of GSFord is the number of backup RR and backup Leaf RR. The backup Leaf RR is the Leaf RR among the backup RRs. In Figure 4.12(a) and 4.12(b), we observed that the average number of backup RRs and backup Leaf RRs of a PNode exhibits almost a constant value regardless of the number of total PNodes in the RRTree. This result well fits into the estimated value obtained by the equation 4.6. With bigger $Th_{str}$, the threshold value of the number of struct nodes, the RRTree can tolerate higher node churn. However, if $Th_{str}$ becomes too big such as 15 or bigger, the average
number of backup RR, especially backup Leaf RR, fluctuates along with the number total PNodes. The main reason is the temporary struct nodes. Generally, the average number of backup Leaf RR is 1 since a PNode needs to store the information of the Leaf RR where it resides. If there are temporary struct nodes, they needs to store the information of the Leaf RRs where it does not locate. According to this, the increase of the temporary struct nodes casued by the high $Th_{str}$ incurs the load imbalance in the aspect of the number of backup RRs.

It is because there are too many temporary struct nodes and they incur the load imbalance

### 4.5.4 Implementation of GSFord

We implemented the initial version of GSFord notification system. We enhanced the CrisisAlert system[124] to incorporate a GSFord registration server, which supports web interfaces to manage the information including the target geography and the registered recipients as well as to initiate an event message tied to a given target geography. The PNode application using the RRTree-based overlay construction protocol is implemented in Java. Figure 4.13 shows detail components of a PNode application.
Both the CrisisAlert-based GSFord registration server and PNodes are tested on an emulated Internet using Modelnet[16] and Inet[14]. The Modelnet emulator machine has a dual 2.6Ghz CPU with 2GB RAM and runs a custom FreeBSD Kernel with a system clock at 100HZ. We used five IBM Linux machines with 900Mhz CPU with 500MB RAM, running individual 50 PNodes per machine; each of these nodes were mapped into a physical location of a stub router generated by the Inet[14] topology generator. PNodes arbitrarily join GSFord and construct the GSFord delivery overlay, and a PNode located at (0,0) coordinate of the global target geography is used as the initial contact node of a regional multicasting.

In Figure 4.14, we present the performance of regional multicasting implemented on the GSFord delivery overlay. We observe that the delivery overlay reaches the recipients outside of the damaged region ($PDR$) caused by a regional failure, reliably and quickly; i.e. under $0.5 \ PDR_r$, about 50% of the recipients receive the message within 1.4 secs. Note that the social diffusion process (not depicted in Figure) increases the reliability to 99%, albeit with...
some delay. We also studied the average latency as a function of distance of the center of PAR from the message initiator. We observed that the latency increases along with the distance. That is, the RRTree-based overlay preserves the locality property.

### 4.6 Concluding Remarks

In this chapter, we studied the dissemination problem in societal scale instant information sharing. Specifically, we consider the event notification in extreme situations and presents a geo-social notification middleware, GSFord, where information needs are strongly correlated to both the geographical location of events as well as the social relationships of people. We presented the design of the GSFord system using a reliable geo-aware overlay structure, RRtree over which we build a reliable multicasting protocol. Furthermore, GSFord exploits a social diffusion process to improve/maximize notification coverage and reliability of notification. Reliable storage of social information at trusted points enables GSFord social entities to forward the alert messages to their social friends inside PAR/PDR, even under extreme geographically correlated regional failures. Our results indicate that even under 80% infrastructure damage, 90% recipients are reached within 30 minutes via GSFord.
Chapter 5

Delay-tolerant Information Sharing: Information Layer

In this chapter, we study efficient and accurate targeting of providers of information. Figure 5.1 highlights the contributions of this chapter in the design framework for societal scale information sharing. We look at the mobile Q&A paradigm as an instance of the class of delay-tolerant information sharing applications. While Q&A systems have been studied in the literature, adopting ordinary Q&A systems in mobile environments is fairly challenging because of the highly dynamic and diverse mobile user contexts. We conjecture Crowd-sourcing is a promising way to build mobile Q&A systems. The key challenge is to provide mobile information seekers with timely, trustworthy and accurate answers while ensuring that information providers are not inappropriately burdened. We tackle this challenge by introducing SmartSource, a crowdsourcing based mobile Question & Answer (Q&A) middleware. SmartSource takes advantage of both static and dynamic context and semantics from mobile users (e.g., geolocation, social network, expertise/interest, device sensor profiles, battery level) to identify sources of information (i.e. workers) that are trusted by the user and accurate enough for the questions at hand. Given a question, the SmartSource broker
middleware executes a scalable and efficient worker selection algorithm that uses a Lyapunov optimization framework to maximize the utility of worker selection while guaranteeing the stability of the overall system. An associated assignor selection is used to scale the selection process to a large number of users.

Figure 5.1: Contributions of Chapter 5 in the societal scale information sharing framework.

We implement the SmartSource prototype system on an Android testbed and thoroughly evaluate the system using real world applications and data, in particular those that involve geospatial questions and answers. Experimental results indicate that SmartSource is efficient and provides superior worker selection compared to baseline approaches. SmartSource is also highly customizable: it employs a general utility function and provides a control knob to tradeoff the optimality and responding time. We argue that our SmartSource middleware integrates humans-in-the loop, and is generic, scalable and flexible. We firmly believe that SmartSource will pave a way for new mechanisms of interaction among mobile users.

Our key contributions include:

- Design of a generic mobile Q&A architecture that brings together information seek-
ers and providers to answer a broad range of questions that involve spatial-temporal context and expertise of users.

- Formulation of optimization problems (cast as worker selection and assignor selection problems) to balance the multiple design goals of accuracy, trustworthiness and overhead cost, at scale.

- Design of stable algorithms for worker selection and assignor selection using the Lyapunov queuing framework.

- Implementation of a prototype SmartSource system on a smartphone testbed and evaluating it through both user studies (for usage) and simulation experiments (for scalability).

The rest of the chapter is organized as follows. Section 5.1 present the background and motivation of the work. Section 5.2 provides an overview of the design of the middleware. We present the formal modeling and problem formulation of the middleware in Section 5.3. The details of the algorithm solutions are presented in Section 5.4. We present our prototype implementation and extensive evaluation in Section 5.5. Finally, the chapter is concluded in Section 5.6.

5.1 Introduction

Mobile devices, such as smartphones, tablets, glasses, and smartwatches, are getting extremely popular in our daily life. For example, a market report [15] points out that more than 70% of Americans have access to smartphones as of 2014 - this allows for anytime, anywhere access to information through mobile Internet connectivities for many users. We conjecture that this rapid increase in smartphone penetration worldwide will enable a new level of interactivity in information exchange. In particular, we anticipate that the time
is ripe for the rise of an interactive Mobile Question and Answer (Q&A) ecosystem that dynamically connects information seekers (i.e., humans with questions) to the best possible information providers (other humans who can provide timely and accurate answers). Unlike existing information retrieval systems (e.g., search engines) and web-based Question and Answer (Q&A) systems (e.g., AskJeeves), we argue for a novel mobile Q&A paradigm that provides fine-grained information that is personalized to the current needs of the information seeker. The envisioned mobile (Q&A) system will allow users to ask specific questions, such as: 1) “Is it raining outside Bren Hall right now?”, 2) “What’s the current south-bound speed at exit 7 of I-405?”, and 3) “How long is the line at the on-campus MacDonald’s?”; and to use a combination of methods (including responses from multiple other users) to provide answers to these questions. In this chapter, we propose such a flexible and interactive mobile Q&A middleware, called SmartSource that intelligently leverages crowdsourcing mechanisms to provide accurate and timely responses to questions.

Adapting search-driven Q&A systems in mobile settings is fairly challenging, primarily due to highly dynamic and diverse mobile user needs and contexts. For example, propagating a question about the road conditions near Corona Del Mar Beach to a broader population, e.g., mobile users in Orange County, California is not useful. Careful selection of the target recipients of a question is essential to (a) ensure that users can participate meaningfully (indeed, noisy apps are typically turned off or deleted by the user) and (b) lower resource consumption in the end-devices and networks. Additionally, mobile users may only want to receive answers from other trusted users such as his/her Facebook friends who are capable of responding.

The use of crowdsourcing as a driving mechanism to build Q&A systems is promising. Crowdsourcing refers to public platforms (e.g., Amazon Mechanical Turk (AMT) [2] and Crowd-Flower [9] ) that allow users to hire others to perform certain tasks; example applications include voting systems, image retrieval and assessment of image searches [161], multimedia
annotations, information sharing, and social games. Enabling such crowdsourcing in mobile devices [56, 96, 161, 148, 79, 167, 57, 60] has been a growing area of study. A recent large-scale measurement study with 85 smartphone users [56] reveals that a small number of smartphone users can provide an impressive coverage of a big city. Recent work in crowd-sensing explores the use of on-board and external sensing mechanisms, including efficient user localization [148] for a range of applications such as air quality monitoring [79], determining transportation times [167], building occupancy [57] and parking spot availability [60]; such content may further be piggybacked on other messages, calls or applications for efficient resource usage [96]. Several of the crowdsourcing examples above are intended to be used in a highly asynchronous manner where responses may arrive in the order of hours/days; the crowdsensing apps on the other hand are intended to work without any human intervention.

In contrast to the above efforts, the crowdsourcing based mobile Q&A system must be general purpose (usable for multiple applications), interactive (exploit human-in-the-loop), dynamic (near real-time responses in an asynchronous setting where participants join and leave at will) and opportunistic (aims to use the best possible responders for the question available). The SmartSource mobile crowdsourcing middleware achieves the above requirements through a combination of strategies. First, SmartSource provides relevant and trustworthy answers through careful selection of responders (e.g., nearby users for location based questions). Secondly, SmartSource enables dynamic/-flexible interaction by allowing mobile users to ask/answer questions anywhere and anytime, and choose whose questions to answer and whose answers to take. Finally, the SmartSource implementation supports cost-effective deployment by reducing end-to-end system overhead (e.g., do not distribute questions to less relevant responders) and being cognizant of device capabilities and resources (e.g., residual battery capacity).
5.2 Middleware Design Overview

In SmartSource, a key design problem is how to target potential information providers to answer questions so that providers are not burdened inappropriately and information requester can obtain trustworthy and timely information. The solution lies in identifying sources of information that are trusted enough for the question at hand; the choice of information sources uses multiple factors include:

- Experts - individual with expertise in the question domain are more trusted when questions are knowledge-driven;

- Friends and family networks - A requester is more likely to trust the answers coming from a close friend or family member as compared to a random responder;

- Those in the vicinity of a location are potentially more suitable candidates to answer location-based questions;

- A combination of the above.

Furthermore, because some questions may require to be answered by sensors embedded in smartphones of users who act as information providers (e.g., pollution sensor can be used to answer a question such as “how’s the air quality?”); in such cases, we need to target mobile users equipped with capable sensors and enough residual energy to answer the questions.

SmartSource uses both static and dynamic contexts and semantics from mobile users (and their smartphones) to recommend potential information providers for questions through a novel utility-driven worker selection algorithm. The contexts include: mobile users’ geolocation, social network connections, expertise and interests, profiling of device sensors, and device battery levels. Moreover, our design keeps scalability and stability in mind. We accommodate a distributed environment, and develop fine-grained control mechanisms based
on the Lyapunov optimization framework that maximizes the utility of worker selection while maintaining a stabilized load of the middleware. We present an overview on the design in the rest of this section.

### 5.2.1 SmartSource Architecture

Figure 5.2 depicts the overview of the SmartSource architecture. It consists of querists, workers, and Q/A broker(s).

![Figure 5.2: Overview of the SmartSource system.](image)

**Querists**: Querists are registered users. They send queries to the broker system to request for specific information they want to know. This can be done through either a mobile app or web portal of the system. In general, a query may consist of: 1) question, 2) location and time specifications, 3) requirements, and 4) reward. The question indicates what information the querist wants to get. Some questions may be location specific, such as “How many sea lions are at Pier 39?” The location specification may be a stadium, a park, a room,
or (more general) a tuple of longitude, latitude, and altitude. Also questions have time specification since querists usually request for information at current time or in near future. Time specification indicates the time frame of the requested information, such as “within an hour from now” or “from 1pm to 5pm”. Requirements can be the quality and quantity of the requested information such as the accuracy and the number of answers that the querist desires. In crowdsourcing applications, querists may put reward (as promised monetary or virtual benefit) into their queries to motivate unknown workers to work on their queries.

**Workers**: Workers are registered users as well. Typically they are mobile users with a mobile app of the system installed on their smartphones. In Q&A and crowdsourcing applications, workers can receive and decide what queries to answer or to work on in three ways: 1) Q&A applications publicly publish submitted queries onto web portals and workers can browse and search questions to answer; 2) to improve targeting, many Q&A applications incorporate a subscription service where workers subscribe to interested topics/fields and applications present queries matching their subscriptions on the workers’ home/wall page; 3) workers notice the applications when they are available and applications send notifications on recommended questions to workers. In this work, we focus on the enabling techniques for the third approach.

**Broker(s)**: The key coordination is performed at a broker - smartphone users need to register with the Q&A applications at the broker for the brokerage service. The broker keeps track of smartphone users’ static and dynamic contexts and carefully select potential workers for submitted queries. Depending on a query’s requirement the broker may select one or multiple workers to sent the query to. To descriptively differentiate queries submitted to the broker and copies send to workers in notifications, we call the copies of queries sent to workers as information tasks. Since workers may refuse to take the recommended tasks and eligible workers may not exist all the time, the brokerage service needs to re-select workers for unassigned queries when new workers are available.
We illustrate the Q&A brokerage service of SmartSource in an example. Consider that Alice is going to a night market, however, she wants to know which booths are popular and how long are the lines for them. She wants the answers back in an hour, so she sends a query to the broker as a querist. Some registered smartphone users located close to the night market are available to answer questions and they become potential workers of the query. The broker notifies the workers on their smartphones with respect to the task and they interact to accept or reject it. If they accept to answer, they may walk to the night market and take pictures of the booths and lines as answers and return them to Alice by the broker. Otherwise, if they reject the task, the broker keeps looking for new workers before the query expires.

Figure 5.3 shows the major software modules in the SmartSource broker middleware. The *query receiver* receives new queries from querists and put them into the *worker selection queue*. The *mobile user information collector* monitors the static and dynamic contexts of mobile users and maintains the *mobile user information* and *social relation information* databases. The crux of the middleware is the *worker selection algorithm* that determines the best candidate workers for queries in the worker selection queue based on the query requirements and worker status from the databases. The *task assignment module* interacts with the selected workers to notify them of the queried tasks and let them decide whether to accept the task. The middleware requires workers to confirm the assignment (e.g., by clicking an “accept” button) within a certain period after receiving the notification, if they are willing to provide answers. Otherwise, it is considered as not assigned. The *task assignment module* monitors the acceptance of tasks. Once a query is accepted by enough workers it is removed from the worker selection queue. It is important to maintain a stable size of the worker selection queue to keep the broker middleware operational. In this work, we design the worker selection algorithm such that it maximizes the utility of worker selection while maintaining a stabilized size of the queue.
5.2.2 Scalability Consideration

Assignors and relays: To support scalable information crowdsourcing to a large number of geographically distributed mobile users, we envision a distributed broker network where each broker takes responsibility for mobile users in a specific region. To be able to flexibly adjust to the distribution of users and provide an efficient structure for information exchanges among brokers, we adopt a distributed hierarchical geo-overlay to connect brokers. More specifically, we apply the RRTree protocol we introduced in chapter 4.3 to construct a geo-based nested tree structure consisting of non-leaf brokers, we call them relays, and leaf brokers, we call them assignors, as illustrated in Figure 5.4.
In the RRTree structure each node (i.e., broker) is responsible for a rectangular geographical region called Responsible Region (RR), and inherently maintains the following property. A parent RR in the tree subsumes the regions covered by its child RRs, and the root of the tree covers the global geography that is supported by the system. The structure is flexible and scalable, and handles the non-uniform distribution of users gracefully by allowing smaller RRs and denser broker deployment in highly populated locations.

Leaf brokers or assignors, are responsible for receiving queries from and assigning tasks to mobile users located inside their responsible regions. The key function of assignors is worker selection. The software module is the one indicated in Figure 5.3. Non-leaf brokers or relays, as ancestors of assignors in the geo-based tree, are responsible for forwarding queries across regions to different assignors for worker selection. We identify following applications that desire forwarding queries to different assignors: 1) in spatial Q&A, if the responsible region of the current assignor does not cover the requested locations of the location-based queries (e.g., querists and their requested locations are not in the same region), the queries need to be forwarded to the right assignors for worker selection; 2) in non-spatial Q&A where queries do not have restrictions on workers’ locations, queries can be forwarded from heavily loaded assignors to less loaded ones to achieve load balancing and allow faster query assignment.

The RRTree overlay implements a geographical routing protocol inherent to the geo-aware structure (see 4.3 for details). We incorporate the routing into the middleware to support query forwarding for spatial Q&A applications. On the other hand, to provide efficient load balancing for non-spatial Q&A applications, we design a distributed assignor selection algorithm for relays to determine best assignors for queries. The beauty of the algorithm is that it does not compromise the long term utility achieved by the worker selection algorithm.

Figure 5.4 illustrates the software modules of a relay. The routing module incorporates the geographical routing inherent to the RRTree structure to provide efficient query forwarding for spatial Q&A applications. To support scalable non-spatial Q&A with efficient load bal-
ancing in distributed environments, each relay keeps track of the backlogs of queues for a randomized subset of assignors in the overlay through the assignor manager. This can be easily achieved through peer sampling services such as [120]. Each relay has an assignor selection queue to hold non-location-based queries and the assignor selection algorithm determines for each query in the queue which assignor it should be sent to for worker selection.

5.3 System Model and Problem Formulation

In this section, we present a formal modeling of the middleware system and formulate the key problem to solve. We focus on the worker selection problem at assignor brokers first. The relatively minor problem, assignor selection at relay brokers, is presented in Section 5.4.3.

5.3.1 System Models

We first present general models used in our system design.

**Query model:** Let \( \Pi \) be the class of questions. We consider a general information query \( r \) from a querist \( q_r \) with optionally following attributes: 1) \( h_r \subset \Pi \) as the requested question; 2) \( g_r \) as the location(s) of the question; 3) \( T_r = < T_{r_{start}}, T_{r_{end}} > \) as the time frame of the question in interest; 4) for information collection queries, \( d_r \) as the required duration of collection and it must be smaller than the time frame of the question; 5) \( k_r \) as the number of workers or answers the querist desires to get; 6) \( b_r \) as reward or benefit (money or virtual credit) the querist promises to each worker.

**Task model:** To send queries to workers, SmartSource first converts queries to information tasks. A task is essentially the same as a query except that each task can be performed by
a single worker independently. Worker cooperation and collaboration are out of the scope of this work. Thus, for a query that desires $k_r$ answers the middleware creates $k_r$ identical tasks and select $k_r$ workers to send them to. Moreover, a query may not be immediately considered active for worker selection after it is received if the start time of the interested context $T_{r}^{\text{start}}$ is too far in the future. This ensures that workers only receive tasks that can be answered or worked on shortly. We define a time interval $T_\theta$ as the \textit{preparation period} and convert a query to active tasks at the first time instance $t$ when $t + T_\theta \geq T_{r}^{\text{start}}$. Queries that haven’t entered their active periods yet are cached at their receiving brokers.

**Worker model**: The middleware maintains up-to-date static and dynamic context of workers. A worker $w$’s profile contains the following information: 1) $y_w \subset \Pi$ as the capability of the worker to answer various questions. It could be the expertise for knowledge-oriented questions, or the capability of device sensors for information collection questions; 2) $i_w \subset \Pi$ as the interest of the worker to answer; 3) $l_w$ as the current location of the worker; 4) $E_w$ as the current energy level of the worker’s smartphone; 5) $n_w$ as the number of tasks the worker wants to receive, and $n_w = 0$ means the worker currently can take no more tasks (e.g., because the worker is already occupied); 6) $f_w$ as the desired labor reward per unit time; 7) $F_w$ is the friend circle of the worker, which is derived from the worker’s online social networks upon registration. Here we consider 1-hop friends only, but it can be readily extended to include multi-hop friends. Besides the above information the system also maintains $R_w$ as a list of rejected queries that the worker refuses to work on. This is required by the worker selection algorithm to ensure that the worker won’t receive the same task again once he rejects it.

Let $Q_i$ denote the task queue at a assignor broker $i$ for worker selection. We divide time into slots (e.g., 10 minutes), and model dynamics of the queue at the brokers.

**Task arrival**: Let $\lambda_i(t)$ be the number of new tasks arriving to broker $i$’s worker selection queue at $t$. Assume that $\lambda_i(t)$ is finite and limited by $\lambda_{\text{max}}$, i.e., $\lambda_i(t) \leq \lambda_{\text{max}} \forall t$. The time
averaged task arrival rate can be written as \( \lambda_i \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\lambda_i(\tau)\} \).

**Task worker selection and assignment:** Every time slot, assignor brokers run the worker selection algorithm (presented in Section 5.4.2) to select workers for tasks in their queues, and send the tasks to the workers. Let \( \mu_i(t) \) be the number of tasks at assignor \( i \) that have their workers selected at \( t \); let \( \mu^r(t) \) be the number of selected tasks of the same query \( r \), \( \mu^w(t) \) be the number of selected tasks to the same worker \( w \), and \( \mu^{r,w}(t) \) be the number of selected tasks of query \( r \) to worker \( w \). We assume \( \mu_i(t) \) is limited by \( \mu_{max} \), i.e., \( \mu_i(t) \leq \mu_{max} \forall t \). Again, the time averaged rate is \( \mu_i \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu_i(\tau)\} \).

A worker, upon receiving a task, decides whether or not to accept the task. If the worker accepts the task, the task is considered as successfully assigned and removed from the worker selection queue. Also, \( n_w \), as the number of tasks the worker wants to receive, is reduced by 1. If the worker rejects the task, the task is kept in the queue and considered for worker selection in next time slot. Also, the query is added to the worker’s rejected query list \( R_w \), so the worker will not get the task again. We require the worker to respond his decision (accept or reject) within the same time slot when he/she receives the task, otherwise the assignment is aborted and the task is left over for worker selection again in next slot. Let \( a_i(t) \) be the number of accepted/assigned tasks at assignor \( i \) at \( t \). We have \( a_i(t) \leq \mu_i(t) \forall t \). So it is limited too and the time average rate is \( \overline{\mu}_i \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu_i(\tau)\} \), and \( \overline{\mu}_i \leq \overline{\mu}_i \).

We assume \( \overline{\mu}_i \geq \varepsilon \cdot \overline{\mu}_i \) where \( 0 < \varepsilon < 1 \) and can be arbitrarily small. This is to ensure a positive worker selection feedback that at least some portion of the selected tasks can be successfully assigned.

**Task dropping:** Tasks that couldn’t get assigned for a long time in the system or have expired (i.e., the interested time frame of the context has passed) need to be dropped from the worker selection queue. Let \( x_i(t) \) be the number of tasks to drop at assignor \( i \) at \( t \), and its time average rate is \( \overline{x}_i \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{x_i(\tau)\} \).
Therefore, the evolution of the queue at an assignor broker \( i \) is written as:

\[
Q_i(t + 1) \triangleq \max (0, Q_i(t) + \lambda_i(t) - a_i(t) - x_i(t))
\]  

(5.1)

where \([\pi]^+ \triangleq \max (0, \pi)\). We define the stability of the queue as:

\[
\overline{Q}_i \triangleq \lim_{t \to \infty} \sup_{T} \frac{1}{t} \sum_{\tau = 0}^{t-1} \mathbb{E}\{Q_i(\tau)\} < \infty.
\]  

(5.2)

At \( t = 0 \) we set \( Q_i(t) = 0 \). The queue is stable as long as \( \overline{Q}_i \leq \overline{a}_i + \overline{x}_i \).

### 5.3.2 Problem Formulation

In this section we formulate the long term objective for worker selection. Considering a task of query \( r \) and a worker \( w \) at time \( t \), we first design working criteria to test the feasibility of assigning the worker onto the task. The criteria includes many factors that can be applied to many mobile Q&A applications: 1) \( E_w \geq E_\theta \) where \( E_\theta \) is the energy threshold for a worker to receive new tasks; 2) \( n_w > 0 \) as the worker wants to receive new tasks; 3) \( r \notin R_w \) as the query is not in the rejected list of the worker; 4) \( h_r \subseteq y_w \) as the \( w \)'s expertise or capability is able to answer the requested question of \( r \); 5) let \( \hat{T}_{r,w} \) be the estimated time duration for \( w \) to complete the task of \( r \), and for spatial sensing questions, it can be evaluated as \( \hat{T}_{r,w} = d_r + T_{r,w}^{mov} \), where \( d_r \) is the required sensing duration and \( T_{r,w}^{mov} \) is the estimated time that the worker needs to spend on moving from his current location to the nearest requested location and it can be derived from services such as Google Map [12]. Then we require \( t + \hat{T}_{r,w} < T_{r,\text{end}} \) as the worker has enough time to complete the task before the task expires; 6) \( b_r \geq f_w \cdot \hat{T}_{r,w} \) as the reward provided by the querist meet the demand of the worker.

**Utility function**: We propose a general utility function of assigning a task to a worker that
can accommodate many mobile Q&A applications as follows:

\[ U^A_{r,w}(t) = [I_{r,w} + \alpha \cdot S_{r,w} - \beta \cdot M_{r,w} - \gamma \cdot C_{r,w}]^+, \]  

where \( I_{r,w} \) is the interest similarity of query \( r \) and worker \( w \), \( S_{r,w} \) is the social closeness of the querist \( q_r \) and \( w \), \( M_{r,w} \) and \( C_{r,w} \) are the estimated moving cost and energy cost respectively for spatial crowdsourcing applications. Interest similarity \( I_{r,w} \) is evaluated using the Jaccard similarity coefficient on \( h_r \) as the requested question of \( r \) and \( i_w \) as the interested question of \( w \), i.e., \( I_{r,w} = |h_r \cap i_w|/|h_r \cup i_w| \). The social closeness between the querist and the worker directly affects the willingness of the worker to work on the query and affects the trust of the querist to the worker’s work. Several recent works has studied how to effectively calculate the social closeness between two users. It can be either explicitly from users’ rating or implicitly from users’ social communications [106, 104]. We evaluate moving cost \( M_{r,w} = T_{r,w}^{\text{mov}} \) as the time that the worker has to spend on moving to the desired location(s) of the task. Furthermore, we estimate energy cost \( C_{r,w} \) from the energy profile of \( w \)’s device sensors and the requested sensing question \( h_r \).

To avoid overflowing the queues, in every time slot the system also determines tasks to drop. Dropped tasks will lose the chance for worker selection in the future. We capture the loss of dropping a task of query \( r \) at \( t \) as:

\[ U^D_r(t) = [d \cdot P_r(t)]^+, \]  

where \( d \) is a constant and \( P_r(t) \) is time decay function, such as: \( P_r(t) = 1 - e^{-(T^\text{end}_r-t)} \). Thus, the system prefers to drop tasks approaching their deadlines because they have less chance to find workers before expire.

**Problem formulation:** We consider the following formulation with a long term goal for
worker selection in the system.

\[
\begin{align*}
\max: \quad & \bar{U} = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E} \left\{ U^A(\tau) - U^D(\tau) \right\}; \\
st: \quad & \bar{\lambda}_i \leq \bar{a}_i + \bar{x}_i \forall i; \\
& \mu^r,w(t) \leq 1 \forall t; \\
& \mu^w(t) \leq n_w.
\end{align*}
\] (5.5a)

The objective function in Eq. (5.5a) maximizes the long term utility of selecting workers and minimizes the loss of dropping tasks. The constraint in Eq. (5.5b) stabilizes the system. The constraint in Eq. (5.5c) ensures that the system will not send more than one task of the same query to a worker. The constraint in Eq. (5.5d) makes sure that the number of tasks assigned to a worker is no more than the number of tasks the worker wants to receive.

### 5.4 SmartSource Algorithms

We design optimal algorithms to solve the problem formulated in Eqs. (5.5a)-(5.5d) based on Lyapunov optimization [74], which is useful for solving long-term optimization problems. We start by presenting our design principles.

#### 5.4.1 Design Principles

We design a novel mechanism to take into account task urgency in worker selection, i.e., to prioritize tasks with approaching expiration deadlines over other tasks. We maintain a dynamic priority-weight \( u_k(t) \) for each unassigned task \( k \) in queue \( Q_i \) as an exponential
function written as:

\[ u_k(t) = c \cdot e^{-(T^\text{end} - t)}. \] (5.6)

For a new task \( k \) arrives or activates at \( t_k \), its priority-weight is initialized as \( u_k(t_k) = c \cdot e^{-(T^\text{end} - t_k)} \). When the task expires at \( t = T^\text{end} \), the weight is \( u_k(T^\text{end}) = c \). We can write the time evolution of \( u_k(t) \) as:

\[ u_k(t + 1) = u_k(t) \cdot e = u_k(t) + u_k(t)(e - 1). \] (5.7)

We use a novel virtual priority-weight queue \( Z_i \) at each broker holding the priority-weights of the unassigned tasks. Once a task is successfully assigned or dropped, its weight is removed from the virtual queue. In practice, the virtual queue is simply a counter as the sum of the weights. That is:

\[ Z_i(t) = \sum_{k \in Q_i(t)} u_k(t). \] (5.8)

Therefore, we can write the queuing dynamics of \( Z_i \) based on that of \( Q_i \) as follows:

\[ Z_i(t + 1) \triangleq [Z_i(t) + \rho_i(t) + \lambda'_i(t) - a'_i(t) - x'_i(t)]^+, \] (5.9)

where \( \rho_i(t) = \sum_{k \in Q_i(t)} u_k(t)(e - 1) \), \( \lambda'_i(t) = \sum_{k \in \lambda_i(t)} u_k(t_k) \), \( \pi'_i(t) = \sum_{k \in \pi_i(t)} u_k(t) \cdot e \), and \( \pi \) is a replacer for symbol \( \mu \), \( a \) and \( x \).

The stability of the virtual queue can be defined as:

\[ \mathbb{E} \triangleq \lim_{t \to \infty} sup_{t} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{Z_i(\tau)\} < \infty. \] (5.10)
From Eq. (5.8), we see at \( t = 0 \) \( Z_i(t) = 0 \). Also, the virtual queue is stable when the real queue \( Q_i \) is stable and \( u_k(t) \) is limited.

Define \( \Theta(t) \triangleq (Q(t), Z(t)) \) as the concatenated vector of the real and virtual queues. We define a quadratic Lyapunov function:

\[
L(\Theta(t)) \triangleq \frac{1}{2} \sum_i (Q_i(t)^2 + Z_i(t)^2).
\]

A one-step Lyapunov drift is defined as:

\[
\Delta(\Theta(t)) \triangleq \mathbb{E}\{L(\Theta(t + 1)) - L(\Theta(t))|\Theta(t)\}.
\]

Intuitively to stabilize the system we want to maintain \( \Delta(\Theta(t)) \) as small as possible, so that \( L(\Theta(t)) \) is small. This is done by sending out as many task assignments and droppings as much as possible in a slot. Taking into account the long term goal we defined in Eq. (5.5a), we construct the following per-slot objective:

\[
\min : \Delta(\Theta(t)) - V\mathbb{E}\{U^A(t) - U^D(t)|\Theta(t)\},
\]

where \( V \) is a weight that controls the relevant importance of the utility optimization over the stability of the queues.

**Lemma 5.1** (Lyapunov Drift). The one-step conditional Lyapunov drift satisfies the follow-
ing constraint at any time slot regardless of algorithms to control the system:

\[ \Delta(\Theta(t)) = \mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\} \]

\[ \leq B + \sum_i \mathbb{E}\{Q_i(t)\lambda_i(t) + Z_i(t)(\rho_i(t) + \lambda_i'(t))|\Theta(t)\} \]

\[ - \sum_i \mathbb{E}\{Q_i(t)\lambda_i(t) + Z_i(t)\lambda_i'(t)|\Theta(t)\} \]

\[ - \sum_i \mathbb{E}\{Q_i(t)\lambda_i(t) + Z_i(t)\lambda_i'(t)|\Theta(t)\}, \]

where \( B \) is a carefully chosen constant.

By applying the lemma to Eq. (5.13), we derive the following lemma:

**Lemma 5.2 (Optimization).** The minimization problem presented in Eq. (5.13) can be solved through the maximization problem \( \max : \Phi(t) \) where \( \Phi(t) \) is defined as follows:

\[ \Phi(t) = \sum_i \mathbb{E}\{VU_i^A(t) + \varepsilon(Q_i(t)\mu_i(t) + Z_i(t)\mu_i'(t))|\Theta(t)\} \]

\[ + \sum_i \mathbb{E}\{Q_i(t)x_i(t) + Z_i(t)x_i'(t) - VU_i^D(t)|\Theta(t)\}. \]

(5.15)

**5.4.2 Worker Selection Algorithm**

Based on the above lemma, we propose the following worker selection algorithm running at each assignor to maximize \( \Phi(t) \) as shown in Algorithm 4.

The worker selection algorithm: It selects (task, worker) pairs \( \mu_{k,w} \) and tasks to drop \( x_k(t) \) from the queue to maximize the objective function in Eq. (5.17) with two constraints. The first constraint guarantees that the tasks sent to a worker are not the same. The second constraint guarantees that the number of tasks sent to a worker are no more than what he asks for. The optimization problem can be reduced to the minimum cost maximum flow problem as we show in Figure 5.5. In the figure we illustrate an example of the worker...
Algorithm 4: Worker Selection Algorithm for Assignor

In time slot $t$ at assignor broker $i$, let $W_i(t)$ be the set of available workers, then determine 1) worker selection $\mu_{k,w}(t) = \{0, 1\}$ for tasks $k \in Q_i(t)$ and $w \in W_i(t)$; and 2) task dropping $x_k(t) = \{0, 1\}$ for tasks $k \in Q_i(t)$ by solving the following optimization problem ($r$ as the query of the task):

$$
\begin{align*}
\text{max:} & \quad \sum_{k \in Q_i(t), w \in W_i(t)} Y_{A_{r,w}}^A(t) \mu_{k,w}(t) + \sum_{k \in Q_i(t)} Y_{D}^D(t) x_k(t) \\
\text{st:} & \quad \mu_{r,w}(t) = \sum_{k \in r} \mu_{k,w}(t) \leq 1; \\
& \quad \mu_{w}(t) = \sum_{k \in Q_i(t)} \mu_{k,w}(t) \leq n_w,
\end{align*}
$$

(5.16)

where

$$
\begin{align*}
Y_{r,w}^A(t) &= VU_{r,w}^A(t) + \varepsilon \cdot Y_{r}(t); \\
Y_{D}^D(t) &= [Y_{r}(t) - VU_{D}^D(t)]^+; \\
Y_{r}(t) &= Q_i(t) + Z_i(t) \cdot u_r(t) \cdot e.
\end{align*}
$$

(5.17)

Note: since all tasks $k$ of the same query $r$ must have the same $u_k(t)$, here we use $u_r(t)$ in the above equation.
is always 0. Finally the capacity from the drop node to the destination is $\infty$ and cost is 0.

Figure 5.5: The representation of a sample optimization problem in minimum cost maximum flow problem.

By reducing to the minimum cost maximum flow problem we can now use any algorithm for that problem to solve our worker selection problem and it is known that the problem can be optimally solved in polynomial time. One of the well-know algorithms is the *Successive Shortest Path and Capacity Scaling* algorithm by Edmonds and Karp, which is a generalization of the Ford-Fulkerson algorithm [67]. The time complexity of the algorithm is $O(|V|^4 \cdot \text{maxcost})$ where *maxcost* is the maximum edge cost (relative to 0 as the minimum edge cost).

5.4.3 Assignor Selection Algorithm

In this section, we present our design of the assignor selection algorithm which runs at relays to provide load balancing. Again, we use Lyapunov optimization to design the algorithm,
and the beauty of the design is that the system long term goal defined in Eqs. (5.5a)-(5.5d) are not compromised by the addition of the algorithm.

We revisit the queuing dynamics of the brokers by taking into account the task flows from relays to assignors. Again, let \( Q_i \) denote the task queue at broker \( i \) (and the broker could be either an assignor or a relay). \( \lambda_i(t) \) is the number of new tasks arriving to broker \( i \). For non-spatial Q&A applications desiring worker selection load balancing, new tasks first arrive at relays for assignor selection. So \( \lambda_i(t) = 0 \) for assignors.

**Task assignor selection:** Every time slot, relay brokers run the assignor selection algorithm to determine assignors for tasks in their queues. Once selected, the tasks are removed from queues at the relay, sent to their selected assignors and put into the queues at the assignors. Let \( \omega_{i,j}(t) \) be the number of tasks at relay \( i \) selected to assignor \( j \) at \( t \). Let \( \omega_j(t) = \sum_{i \in Rl} \omega_{i,j}(t) \) be the total number of tasks received by assignor \( j \) from relays at \( t \) and \( \psi_i(t) = \sum_{j \in As} \omega_{i,j}(t) \) be the total number of tasks sent out of relay \( i \) at \( t \). \( Rl \) and \( As \) denote the set of relays and assignors in the system. Both \( \omega_j(t) \) and \( \psi_i(t) \) must be finite. We assume they are limited by \( \omega_{max} \) and \( \psi_{max} \) respectively, i.e., \( \omega_j(t) \leq \omega_{max} \) and \( \psi_i(t) \leq \psi_{max} \forall t \). The time averaged rates are \( \overline{\omega}_i \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\omega_i(\tau)\} \) and \( \overline{\psi}_i \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\psi_i(\tau)\} \).

A generalized queuing dynamics for \( Q_i \) at a broker \( i \) (indifferent for relays and assignors) is:

\[
Q_i(t+1) \triangleq [Q_i(t) + \lambda_i(t) + \omega_i(t) - \psi_i(t) - a_i(t) - x_i(t)]^+,
\]

(5.18)

where \( \omega_i(t) = a_i(t) = \mu_i(t) = x_i(t) = 0 \) if \( i \) is a relay, and \( \lambda_i(t) = \psi_i(t) = 0 \) if \( i \) is a assignor.

Relay brokers maintain virtual queue \( Z_i \) the same as we presented earlier. When a task is sent from a relay broker to an assignor broker, its priority-weight is transferred to the virtual queue of the assignor broker from that of the relay broker as well. Therefore, a generalized
queuing dynamics for $Z_i$ can be written as:

$$Z_i(t + 1) \triangleq [Z_i(t) + \rho_i(t) + \omega'_i(t) - \psi'_i(t) - a'_i(t) - x'_i(t)]^+. \tag{5.19}$$

We update the system Lyapunov function to contain $Q$ and $Z$ at relays as well. Using the generalized queuing dynamics in Eq. (5.18) and Eq. (5.19), the Lyapunov drift bound, now, becomes:

**Lemma 5.3 (Lyapunov Drift).** The one-step conditional Lyapunov drift for the entire system of both relays and assignors satisfies the following constraint at any time slot:

$$\Delta(\Theta(t)) = \mathbb{E}\{L(\Theta(t + 1)) - L(\Theta(t))|\Theta(t)\}$$

$$\leq B' + \sum_i \mathbb{E}\{Q_i(t)x_i(t) + Z_i(t)(\rho_i(t) + \lambda'_i(t))|\Theta(t)\}$$

$$- \sum_{i \in R} \sum_{j \in A_s} \mathbb{E}\{\nabla^Q_{i,j}(t)\omega_{i,j}(t) + \nabla^Z_{i,j}(t)\omega'_{i,j}(t)|\Theta(t)\}$$

$$- \sum_{i \in A_s} \mathbb{E}\{Q_i(t)a_i(t) + Z_i(t)a'_i(t)|\Theta(t)\}$$

$$- \sum_{i \in A_s} \mathbb{E}\{Q_i(t)x_i(t) + Z_i(t)x'_i(t)|\Theta(t)\}, \tag{5.20}$$

where $B'$ is a constant, $\nabla^Q_{i,j}(t) = Q_i(t) - Q_j(t)$ and $\nabla^Z_{i,j}(t) = Z_i(t) - Z_j(t)$.

We consider the same per-slot objective in Eq. (5.13) and by applying the lemma we derive an additional optimization problem specific for relays besides the original optimization defined in lemma 5.2.

**Lemma 5.4 (Optimization).** The minimization problem presented in Eq. (5.13) can be divided and solved through two maximization problems $\max : \Phi(t)$ and $\max : \Phi'(t)$ where $\Phi(t)$...
is defined in lemma 5.2 and $\Phi'(t)$ is defined as follows:

$$\Phi'(t) = \sum_{i \in R} \sum_{j \in A} \mathbb{E}\{\nabla Q_{i,j}(t) \omega_{i,j}(t) + \nabla Z_{i,j}(t) \omega'_{i,j}(t) | \Theta(t)\}. \quad (5.21)$$

It is not hard to see that the above optimization problem for relays is aligned with the optimization for assignors in lemma 5.2 and they together stabilize the queues and optimize the global long term objective of the system. We propose the following assignor selection algorithm running at each relay to maximize $\Phi'(t)$.

**Algorithm 5: Assignor Selection Algorithm for Relay**

Let $A_i$ denote the set of candidate assignors for a relay $i$. In time slot $t$ at relay broker $i$, for each task $k \in Q_i(t)$ select $j \in A_i$ that the following objective is satisfied:

$$\max: \quad V_k = (Q_i(t) - Q_j(t)) + (Z_i(t) - Z_j(t)) \cdot u_k(t) \cdot e. \quad (5.22)$$

If $V_k > 0$ then send the task $k$ to $j$. Otherwise, keep $k$ in $Q_i$ and do not send.

The **assignor selection algorithm**: It takes $O(mn)$ time, where $m$ is the number of candidate assignors of the relay broker and $n$ is the number of tasks in the assignor selection queue. We can see that the selection decisions are made based on the backlogs at assignors. Intuitively, by maximizing Eq. (5.22) the algorithm achieves load balancing since it prefers assignors with smaller backlogs. Also, if all candidate assignors are loaded (i.e., $V_k < 0$) the tasks are better off held by the relay before they are less loaded.

### 5.4.4 Performance Analysis

We present several theoretical performance bounds of the proposed algorithms. Let us denote $U^*$ as the maximum achievable time averaged utility of the long term optimization problem in Eqs. (5.5a)-(5.5d) by any stationary algorithms. Our proposed algorithms by solving the per-slot optimization in Eq. (5.13) achieve a time averaged utility $\bar{U}$ arbitrarily close to $U^*$. 123
Theorem 5.5 (Utility lower bound). Our algorithm achieves a lower bound of the time averaged utility:

$$U = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{U^A(\tau) - U^D(\tau)\} \geq U^* - \frac{B'}{V},$$

(5.23)

where $B'$ is a constant defined in Lemma. 5.3.

The theorem shows that by choosing an arbitrarily large $V$, the achieved utility is arbitrarily close to the maximum achievable value.

Theorem 5.6 (Total queue size upper bound). We define

$$\epsilon^Q = \min_i \mathbb{E}\{\psi_i(t) + a_i(t) + x_i(t) - \lambda_i(t) - \omega_i(t)|\Theta(t)\},$$

and

$$\epsilon^Z = \min_i \mathbb{E}\{\psi'_i(t) + a'_i(t) + x'_i(t) - \rho_i(t) - \lambda'_i(t) - \omega'_i(t)|\Theta(t)\},$$

and $\epsilon = \min(\epsilon^Q, \epsilon^Z)$. Our algorithms restrict an upper bound on the time average for the sum of all queues in the system as:

$$\bar{Q} = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{i \in Al} \mathbb{E}\{Q_i(t) + Z_i(t)\} \leq \frac{B' + V(U_{\text{max}} - U^*)}{\epsilon},$$

(5.24)

where $U_{\text{max}} \geq U^A(t) - U^D(t)\forall t$ is the maximum utility for any single slot (Apparently time averaged utility $U^*$ can not be larger than this value).

Theorem 5.7 (Individual queue size upper bound). Consider $U_{\text{max}}^D \geq U_{\text{opt}}^D(t)\forall r, t$ as the maximum utility loss of dropping a task, our algorithm restricts the storage limitation of a worker selection queue at an assignor broker as:

$$Q(t) \leq VU_{\text{max}}^D + \mu_{\text{max}} + \lambda_{\text{max}} \quad \forall t.$$

(5.25)
Theorem 5.6 restricts the total sizes of all queues in the system while theorem 5.7 provides a tighter restriction on individual worker selection queues. Both theorems indicate that a larger $V$ leads to a larger value of size limitation on the queues. That is, there is a trade-off $O(V, 1/V)$ for the size of the queues and the achieved long term time averaged utility.

5.4.5 Theoretical Proofs

Proof of Lemma 5.1

The one-step conditional Lyapunov drift is:

$$
\Delta(\Theta(t)) = \mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\} = \frac{1}{2} \sum_i \mathbb{E}\left\{Q_i(t+1)^2 - Q_i(t)^2 | \Theta(t)\right\},
$$

(5.26)

$$
+ \frac{1}{2} \sum_i \mathbb{E}\left\{Z_i(t+1)^2 - Z_i(t)^2 | \Theta(t)\right\},
$$

where the evolution of $Q_i(t)$ and $Z_i(t)$ are defined as:

$$
Q_i(t+1) \triangleq [Q_i(t) + \lambda_i(t) + \omega_i(t) - \psi_i(t) - a_i(t) - x_i(t)]^+, \quad (5.27)
$$

and

$$
Z_i(t+1) \triangleq [Z_i(t) + \rho_i(t) + \lambda_i'(t) + \omega_i'(t) - \psi_i'(t) - a_i'(t) - x_i'(t)]^+. \quad (5.28)
$$
Note that

\[
Q_i(t+1)^2 - Q_i(t)^2 = (\lambda_i(t) + \omega_i(t) - \psi_i(t) - a_i(t) - x_i(t))^2
\]

\[+ 2Q_i(t)(\lambda_i(t) + \omega_i(t) - \psi_i(t) - a_i(t) - x_i(t)), \tag{5.29}\]

similarly,

\[
Z_i(t+1)^2 - Z_i(t)^2 = (\rho_i(t) + \lambda_i'(t) + \omega_i'(t) - \psi_i'(t) - a_i'(t) - x_i'(t))^2
\]

\[+ 2Z_i(t)(\rho_i(t) + \lambda_i'(t) + \omega_i'(t) - \psi_i'(t) - a_i'(t) - x_i'(t)). \tag{5.30}\]

Let’s define a constant \( B \) as below:

\[
B(t) \triangleq \frac{1}{2} \sum_i (\lambda_i(t) + \omega_i(t) - \psi_i(t) - a_i(t) - x_i(t))^2
\]

\[+ \frac{1}{2} \sum_i (\rho_i(t) + \lambda_i'(t) + \omega_i'(t) - \psi_i'(t) - a_i'(t) - x_i'(t))^2
\]

\[\leq \frac{1}{2} \sum_i (\lambda_{\text{max}}(t) + \omega_{\text{max}}(t)
\]

\[- \psi_{\text{max}}(t) - a_{\text{max}}(t) - x_{\text{max}}(t))^2
\]

\[+ \frac{1}{2} \sum_i (\rho_{\text{max}}(t) + \lambda_{\text{max}}'(t) + \omega_{\text{max}}'(t)
\]

\[- \psi_{\text{max}}'(t) - a_{\text{max}}'(t) - x_{\text{max}}'(t))^2
\]

\[= B \tag{5.31}\]

126
The Lyapunov drift now becomes:

\[
\Delta(\Theta(t)) = \mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\}
\]
\[
\leq B + \sum_i \mathbb{E}\{Q_i(t)\lambda_i(t) + Z_i(t)(\rho_i(t) + \lambda'_i(t))|\Theta(t)\}
\]
\[
+ \sum_i \mathbb{E}\{Q_i(t)\omega_i(t) + Z_i(t)\omega'_i(t)|\Theta(t)\}
\]
\[
- \sum_i \mathbb{E}\{Q_i(t)\psi_i(t) + Z_i(t)\psi'_i(t)|\Theta(t)\}
\]
\[
- \sum_{i \in As} \mathbb{E}\{Q_i(t)a_i(t) + Z_i(t)a'_i(t)|\Theta(t)\}
\]
\[
- \sum_{i \in As} \mathbb{E}\{Q_i(t)x_i(t) + Z_i(t)x'_i(t)|\Theta(t)\}.\]

Note that \(\omega_i(t) = 0\) for \(i \in As\), \(\psi_i(t) = 0\) for \(i \in Rl\), and

\[
\omega_i(t) = \sum_{j \in As} \omega_{i,j}(t) \quad \omega'_i(t) = \sum_{j \in As} \omega'_{i,j}(t)
\]
\[
\psi_i(t) = \sum_{j \in Rl} \omega_{j,i}(t) \quad \psi'_i(t) = \sum_{j \in Rl} \omega'_{j,i}(t)
\]

We can rewrite the Lyapunov drift to:

\[
\Delta(\Theta(t)) = \mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\}
\]
\[
\leq B + \sum_i \mathbb{E}\{Q_i(t)\lambda_i(t) + Z_i(t)(\rho_i(t) + \lambda'_i(t))|\Theta(t)\}
\]
\[
- \sum_{i \in Rl} \mathbb{E}\{\nabla^Q_{i,j}(t)\omega_{i,j}(t) + \nabla^Z_{i,j}(t)\omega'_{i,j}(t)|\Theta(t)\} \quad (5.34)
\]
\[
- \sum_{i \in As} \mathbb{E}\{Q_i(t)a_i(t) + Z_i(t)a'_i(t)|\Theta(t)\}
\]
\[
- \sum_{i \in As} \mathbb{E}\{Q_i(t)x_i(t) + Z_i(t)x'_i(t)|\Theta(t)\},
\]

where \(\nabla^Q_{i,j}(t) = Q_i(t) - Q_j(t)\) and \(\nabla^Z_{i,j}(t) = Z_i(t) - Z_j(t)\).
Proof of Lemma 5.2

The per-slot objective is

\[
\min : \Delta(\Theta(t)) - V \mathbb{E}\{U^A(t) - U^D(t)|\Theta(t)\}. \tag{5.35}
\]

By applying lemma 5.1 into it and removing constants we derive:

\[
\begin{align*}
\max & : \sum_{i \in R_l} \sum_{j \in As} \mathbb{E}\{\nabla^Q_{i,j}(t)\omega_{i,j}(t) + \nabla^Z_{i,j}(t)\omega'_{i,j}(t)|\Theta(t)\} \\
& + \sum_{i \in As} \mathbb{E}\{VU^A_i(t) + Q_i(t)a_i(t) + Z_i(t)a'_i(t)|\Theta(t)\} \\
& + \sum_{i \in As} \mathbb{E}\{Q_i(t)x_i(t) + Z_i(t)x'_i(t) - VU^D_i(t)|\Theta(t)\} \tag{5.36}
\end{align*}
\]

We know that \(\overline{a_i} \geq \varepsilon \overline{\mu_i}\). Moreover, given that \(a'_i = \sum_{k \in a_i} u_k \cdot e\) and \(\mu'_i = \sum_{k \in \mu_i} u_k \cdot e\), assuming distribution of \(u_k\), we can derive the time averaged \(a'_i\) as \(\overline{a'_i} = \overline{\mu_i} \mathbb{E}\{u_k\} \cdot e\), and that of \(\mu'_i\) as \(\overline{\mu'_i} = \overline{\mu_i} \mathbb{E}\{u_k\} \cdot e\). We have \(\overline{a'_i} \geq \varepsilon \overline{\mu'_i}\). Therefore, the objective can be written as:

\[
\begin{align*}
\max & : \sum_{i \in R_l} \sum_{j \in As} \mathbb{E}\{\nabla^Q_{i,j}(t)\omega_{i,j}(t) + \nabla^Z_{i,j}(t)\omega'_{i,j}(t)|\Theta(t)\} \\
& + \sum_{i \in As} \mathbb{E}\{VU^A_i(t) + \varepsilon (Q_i(t)\mu_i(t) + Z_i(t)\mu'_i(t))|\Theta(t)\} \tag{5.37} \\
& + \sum_{i \in As} \mathbb{E}\{Q_i(t)x_i(t) + Z_i(t)x'_i(t) - VU^D_i(t)|\Theta(t)\}
\end{align*}
\]

Define \(\Phi_1(t)\) and \(\Phi_2(t)\) as:

\[
\Phi_1(t) = \sum_{i \in R_l} \sum_{j \in As} \mathbb{E}\{\nabla^Q_{i,j}(t)\omega_{i,j}(t) + \nabla^Z_{i,j}(t)\omega'_{i,j}(t)|\Theta(t)\}, \tag{5.38}
\]
\[
\Phi_2(t) = \sum_{i \in A_s} \mathbb{E}\{VU_i^A(t) + \varepsilon(Q_i(t)\mu_i(t) + Z_i(t)\mu'_i(t))|\Theta(t)\}
\]
\[
+ \sum_{i \in A_s} \mathbb{E}\{Q_i(t)x_i(t) + Z_i(t)x'_i(t) - VU_i^D(t)|\Theta(t)\}. \tag{5.39}
\]

so we have the per-slot objective as:

\[
\text{max} : \Phi_1(t) + \Phi_2(t), \tag{5.40}
\]

which can be split into \( \text{max} : \Phi_1(t) \) and \( \text{max} : \Phi_2(t) \).

**Proof of Theorem 5.5**

Let’s denote \( U^* \) is the maximum utility defined in Eq. 5.5a achieved by some stationary algorithm. Now from our per-slot objective function:

\[
\Delta(\Theta(t)) - V\mathbb{E}\{U(t)|\Theta(t)\}, \tag{5.41}
\]

by applying the results from Lemma 5.1, we have:

\[
\mathbb{E}\{L(\Theta(t + 1)) - L(\Theta(t))|\Theta(t)\} - V\mathbb{E}\{U(t)|\Theta(t)\}
\leq B + \sum_i \mathbb{E}\left\{Q_i(t)(\lambda_i(t) + \omega_i(t) - \psi_i(t) - a_i(t) - x_i(t))\right\}
\]
\[
+ \sum_i \mathbb{E}\left\{Z_i(t)(\rho_i(t) + \lambda'_i(t) + \omega'_i(t) - \psi'_i(t) - a'_i(t) - x'_i(t))\right\}
\]
\[
- V\mathbb{E}\{U(t)|\Theta(t)\}. \tag{5.42}
\]
Now define:

\[ \epsilon^Q = \min_i \mathbb{E}\{\psi_i(t) + a_i(t) + x_i(t) - \lambda_i(t) - \omega_i(t)\}, \]

and

\[ \epsilon^Z = \min_i \mathbb{E}\{\psi'_i(t) + a'_i(t) + x'_i(t) - \rho_i(t) - \lambda'_i(t) - \omega'_i(t)\}. \]

We take an expectation with respect to the distribution of \( \Theta(t) \) and use the law of iterated expectations to yield:

\[
\mathbb{E}\{L(\Theta(t + 1)) - L(\Theta(t))|\Theta(t)\} - V \mathbb{E}\{U(t)|\Theta(t)\} \\
\leq B - \epsilon^Q \sum_i \mathbb{E}\{Q_i(t)\} - \epsilon^Z \sum_i \mathbb{E}\{Z_i(t)\} - VU^*. \tag{5.43}
\]

Sum over all slots \( t \in (0, 1, ..., T - 1) \) and divide by \( T \):

\[
\frac{1}{T} \mathbb{E}\{L(\Theta(T)) - L(\Theta(0))\} - \frac{V}{T} \sum_{\tau=0}^{T-1} \mathbb{E}\{U(\tau)\} \\
\leq B - \frac{\epsilon^Q}{T} \sum_{\tau=0}^{T-1} \sum_i \mathbb{E}\{Q_i(\tau)\} - \frac{\epsilon^Z}{T} \sum_{\tau=0}^{T-1} \sum_i \mathbb{E}\{Z_i(\tau)\} - VU^*. \tag{5.44}
\]

Divide both sides in the above by \( V \), and simply rearrange the terms to achieve:

\[
\frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}\{U(\tau)\} \\
\geq U^* - \frac{B}{V} + \frac{\epsilon^Q}{VT} \sum_{\tau=0}^{T-1} \sum_i \mathbb{E}\{Q_i(\tau)\} + \frac{\epsilon^Z}{VT} \sum_{\tau=0}^{T-1} \sum_i \mathbb{E}\{Z_i(\tau)\} \tag{5.45}
\]

\[
+ \frac{1}{VT} \mathbb{E}\{L(\Theta(T))\} - \frac{1}{VT} \mathbb{E}\{L(\Theta(0))\}. \]
Taking the limits as $T$ goes to $\infty$, and notice that $E\{L(\Theta(0))\} = 0$, $E\{L(\Theta(T))\} \geq 0$, $\sum_i E\{Q_i(\tau)\} \geq 0$ and $\sum_i E\{Z_i(\tau)\} \geq 0$, we obtain:

$$\bar{U} = \lim_{T \to \infty} \frac{1}{T} \sum_{\tau=0}^{T-1} E\{U(\tau)\} \geq U^* - \frac{B}{V}.$$  \hfill (5.46)

**Proof of Theorem 5.6**

Similar to the proof of Theorem 5.5, now rearrange the terms of Eq. 5.44 differently we derive:

$$\frac{\epsilon^Q}{T} \sum_{\tau=0}^{T-1} \sum_i E\{Q_i(\tau)\} + \frac{\epsilon^Z}{T} \sum_{\tau=0}^{T-1} \sum_i E\{Z_i(\tau)\} \leq B - \frac{1}{T} E\{L(\Theta(T)) - L(\Theta(0))\} + \frac{V}{T} \sum_{\tau=0}^{T-1} E\{U(\tau)\} - VU^*.$$  \hfill (5.47)

Define $\epsilon = \min(\epsilon^Q, \epsilon^Z)$, we have:

$$\frac{\epsilon}{T} \sum_{\tau=0}^{T-1} \sum_i E\{Q_i(\tau) + Z_i(\tau)\} \leq \frac{\epsilon^Q}{T} \sum_{\tau=0}^{T-1} \sum_i E\{Q_i(\tau)\} + \frac{\epsilon^Z}{T} \sum_{\tau=0}^{T-1} \sum_i E\{Z_i(\tau)\} \leq B - \frac{1}{T} E\{L(\Theta(T)) - L(\Theta(0))\} + \frac{V}{T} \sum_{\tau=0}^{T-1} E\{U(\tau)\} - VU^*.$$  \hfill (5.48)

Let $U_{\text{max}} \geq U(t)\forall t$ be the maximum utility bound for any single slot, so $\frac{V}{T} \sum_{\tau=0}^{T-1} E\{U(\tau)\} \leq VU_{\text{max}}$. Bring it into the above inequality and divide both sides by $\epsilon$ and take the limits as $T$ goes to $\infty$ as in the proof of Theorem 5.5 we finally obtain:

$$\bar{Q} = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_i E\{Q_i(t) + Z_i(t)\} \leq \frac{B + V(U_{\text{max}} - U^*)}{\epsilon}.$$  \hfill (5.49)
Proof of Theorem 5.7

First, it is clear that a task must be dropped by the worker selection algorithm when it expires because its $U_r^D(t) = 0$ and the dropping condition is $Q_i(t) + Z_i(t) \cdot u_r(t) \cdot e - VU_r^D(t) > 0$. Let $U_{max}^D \geq U_r^D(t) \forall r \forall t$ be the maximum utility loss of dropping a single task. From the dropping condition, for any task in queue that the algorithm doesn’t drop and does not select workers, we must have $Q_i(t) + Z_i(t) \cdot u_r(t) \cdot e \leq VU_r^D$. Since $Z_i(t)$ and $u_r$ are non-negative, we must have $Q_i(t) \leq VU_r^D$. Since tasks can also be kept in queue when they select workers and $\mu_{max}$ is the maximum number of such tasks. Take into account possible new coming tasks as well, we obtain the bound on $Q(t)$ as:

$$VU_{max}^D + \mu_{max} + \lambda_{max} \forall t.$$  \hspace{1cm} (5.50)

5.5 Implementation and Performance Evaluation

5.5.1 SmartSource Implementation

We implemented a prototype system for spatial mobile Q&A and collected real traces from users. The system consists of (i) broker, as a Java application deployed on a Linux server and (ii) client, as an Android application distributed to mobile users. Fig. 5.6 shows the architecture of our prototype, and Fig. 5.7 shows the screenshots of the client app. Mobile users interact with the client app to ask and answer questions. The front-end user interface embeds the Google Map service to allow users to easily target locations for questions (Fig. 5.7(a)). The app provides five predefined questions such as “are the shops open?” and “is crowded?” to allow users ask questions in a click, or the users can type in their customized questions. The submitted questions and answers are stored in the database at the server side. Moreover, the client app records the user’s location and energy and period-
ically reports to the broker. The information is stored in the database as well. The worker selection algorithm is implemented at the broker and recommended questions are shown as markers on the map at the client (Fig. 5.7(b)). Users can click on a marker to answer the question (Fig. 5.7(c)). Users can also check the answers of their own questions.

Figure 5.6: Prototype architecture.

We distributed the client app to 7 students in NTHU to use for one week. Some statistic results are shown in Figs. 5.8(a) to 5.8(c). Fig. 5.8(a) presents the number of submitted and answered questions for every 8 hours during the period. The results show that over 80%
questions are answered. Also, the peak usage (i.e., question asking and answering) period is noon and evening. Fig. 5.8(b) shows the average delay for a question to get answered since it is asked. We observe that most of the time the queries can get the answers in less than 10 hours. This could be a sign that users use the system very frequently since they benefit (receive answers) from the system. The statistic result in Fig. 5.8(c) presents the average number of answers received for questions. We also collect the feedback from the users by questionnaire. The feedback shows that they use the application each day. Most of the users say that the answers are useful to them.

Although we implement the prototype system to be used in real life, the number of users and questions are still few and cannot provide enough information to understand the performance of the solution. Therefore, we conducted extensive simulations to evaluate the performance of our proposed algorithms.

![Figure 5.8: Statistics of the usage of the client app.](image)

(a) Number of queries and answered queries in each time period. (b) Average answering delay. (c) Average number of answers of queries that are answered by users.

5.5.2 Performance Evaluation

To get a better understanding of the performance of SmartSource, we conducted several experiments on real-world data. Below we first discuss our trace collection and experimental methodology. We then present our experimental results.
Trace Collection

We collected two real-world datasets for the evaluation. The first dataset is obtained from PTT [19], a Taiwan based BBS. It is arguably the largest BBS in the world with more than 1.5 million registered users. We collected 5700 posts in 10 days period from April 11 to April 20, 2014, and treat them as queries. We assume the PTT users as workers. To simulate spatial-temporal Q&A, we extract locations of the posts and users. We assume the location of the author of a post is the requested location of the query. We approximate users’ locations from their IPs using triangulation [156, 154]: we recruited three servers in Taiwan and let them ping the IPs to estimate their distances to the servers from the network delay (RTT). We also partitioned Taiwan into grids and determine the location of an IP by computing the Mean-Square-Error (MSE) of each grid’s and the IP’s distances to servers and finding the minimum. A random offset within a grid is assigned to each user. Moreover, we used random waypoint model to simulate workers movement when they are idle. When workers are performing tasks, they are moving towards the requested locations of the assigned tasks.

To simulate more complicated scenarios, e.g., taking into account the topics of questions in Q&A, we collected the second dataset. We crawled Quora [20] and collected 1188 questions under the topic group “San Francisco” and 1431 users which are answerers of the questions. The topic group contains many sub-topic tags (e.g., “restaurants in San Francisco”) and we recorded all the topic tags for the crawled questions and those followed by users, serving as the classes of questions of the queries and interests of the workers. We used the dataset for both spatial-temporal and knowledge-based Q&As. To simulate spatial-temporal Q&A, we imported the mobility trace dataset [134] of approximately 500 taxis collected over 30 days in the San Francisco Bay Area. We randomly assigned the taxis traces to workers and locations to queries. Furthermore, we simulated 3 types of smartphones with distinctive energy costs for sensing topics of questions, and randomly assign smartphones to workers. One the other hand, to simulate knowledge-based Q&A based on social networks [106], we
imported the Facebook social graph obtained by McAuley and Leskovec in [122], consisting of 193 circles and 4039 users. We randomly map Quora users to Facebook users.

**Experimental Methodology**

We conducted three sets of experiments. In the first set of experiments, we evaluated the performance of SmartSource against a few baseline approaches for spatial-temporal Q&A. We considered three baseline algorithms for worker selection, given a set of spatial-temporal queries and a set of workers available in each time slot: 1) Nearest: a greedy heuristic algorithm that iteratively picks the pair of query and worker that the distance from the worker to the query is the minimum among all query worker pairs; 2) Nearest Neighbor Priority (NNP): an optimal algorithm proposed in [88] to maximize the number of task assignment while minimizing the travel cost of the worker to move to the requested location; 3) Least Location Entropy Priority (LLEP): another algorithm proposed in [88] to assign higher priority to tasks which are located in worker-sparse areas. In the rest of the experiments, we evaluated the tuning of system parameters on the performance of SmartSource for both spatial-temporal and knowledge-based Q&As. We considered the following performance metrics: 1) complete ratio as the number of queries completed by workers before their deadlines over the total number of queries; 2) average responding time as the average time for a query to get answered since it is submitted. For spatial-temporal Q&A the time includes the query queuing time for worker selection, workers’ moving time to queried locations and the time for collecting the queried information. For knowledge-based Q&A, the time is primarily the queuing time and the time to provide answer; 3) algorithm time as the computational time of the worker selection algorithms; 4) worker busy time and energy cost in spatial-temporal Q&A as the total time and total energy cost for workers working on queries; 5) utility as the total assign utility (in Eq. 5.3) from the algorithm.

We built a Java based simulator driven by the datasets and simulated a Q&A environment
of 10 days with time slot of 10 minutes. Each query randomly picks a $k_r$ (i.e., number of required workers) between 1 and $k_r^{\text{max}}$, and has a deadline of 6 hours after posted. Each worker can only accept one task at a time, and will not receive new tasks before he/she finishes the current one. Moreover, workers may not always be available even they have no tasks to perform. We set a probability (50% or 33%) for their availability in each time slot. In our experiments with the PTT dataset, we considered 200 workers and varied the number of queries between 350 and 5600. In experiments with the Quora dataset, we fixed the number of queries to 1000 and varied the number of workers between 100 and 1000.

![Diagram](image-url)

Figure 5.9: Comparing Smartsource with others on spatial-temporal Q&A, $k_r^{\text{max}} = 3$, 50% worker availability.
Experimental Results

**Compared with other strategies.** Figure 5.9 and 5.10 shows the comparison of SmartSource against baseline algorithms on spatial-temporal Q&A. We used the PTT dataset and varied the number of queries submitted by queriests. Since all the baseline algorithms perform worker selection only based on the locations of queries and workers, for fairness in comparison we adjust the utility in SmartSource to consider moving cost only. We simulated two scenarios $s_1$ (Fig. 5.9) and $s_2$ (Fig. 5.10) with different worker demands from queries and different worker availability. Apparently $s_2$ represents scarcer worker resources due to higher worker demands and lower worker availability. We observe that with the increase of the number of queries, the complete ratio of all algorithms are decreasing due to the lack of worker resources. Compared with other strategies, SmartSource achieves 90% complete ratio,
which is over 10% higher than other algorithms in s1 (Fig. 5.9(a)) and it is less impacted by the increase in the number of queries. In s2 where the worker resources are scarcer, SmartSource is still able to maintain 80% complete ratio, which is over 20% higher than baselines (5.10(a)). This owes to its deadline-aware queuing and worker selection mechanism.

We also evaluated the query responding time and worker busy time as an indication how fast can queries be answered and how much work load for workers. We observe that in spatial-temporal Q&A, workers travel time to requested locations is an important factor impacting the query responding time and worker busy time. NNP has lowest query responding time (Fig. 5.9(b) and 5.10(b)) and worker busy time (Fig. 5.9(c) and 5.10(c)) because it assigns tasks to workers with minimized traveling distance. SmartSource also minimizes workers’ traveling cost but has a longer query responding time than NNP primarily because of the queuing delay of the queries before they are assigned. With the increase of number of queries, the total worker busy time (Fig. 5.10(c)) is increasing for all algorithms because more tasks were performed. We observe that although SmartSource workers performed more tasks than those in other strategies (because of a higher query complete ratio), their total busy time are close.

Figure 5.9(d) and 5.10(d)) show the total computational time of the algorithms for worker selection. We can see that the computational time of all algorithms are increasing with the increase of the number of queries. SmartSource has a similar computational time compared to other strategies but achieves better worker selection performance.

**Performance tradeoffs of the algorithm.** We evaluated the performance of the SmartSource worker selection under varying system parameters. We used the Quara dataset to take into account question topics, sensing energies and social networks in the simulation. We first evaluated the tuning of parameters in the utility function. Figure 5.11(a) to 5.11(c) presents its performance by varying $\gamma$ (i.e., weight for energy cost) in spatial-temporal sensing applications. We used the performance of NNP as a baseline. We observe that with
the increase of $\gamma$ the sensing energy cost of workers in SmartSource are decreasing (Fig. 5.11(a)). However, this is accompanied with an increase in worker busy time (Fig. 5.11(b)) and query response time (Fig. 5.11(c)). This is because the worker selection places higher priority on minimizing workers’ energy cost for sensing by compromising workers’ traveling distance, which leads to an increase in the overall working time on spatial queries. By tuning parameters in the utility function, SmartSource can be customized for different applications.

We further evaluated the performance of the SmartSource by varying $V$ (i.e., weight for utility over queue length). We considered a knowledge-based Q&A and set $\beta = \gamma = 0$ in the utility so the worker selection mainly considers the interest similarity and social closeness. Figure 5.11(d) to 5.11(f) presents the results. We observe that with larger $V$, SmartSource is able to achieve higher utility (Fig. 5.11(d)) and higher query complete ratio (Fig. 5.11(e)). However, although larger $V$ leads to higher utility, it also leads to larger queue size and longer queuing delay. This can be clearly seen from the increase in the query response time.
Moreover, the total utility becomes stable while $V$ keeps growing, because it is approaching the long term optimal utility.

5.6 Conclusion

We expect that mobile Q&A applications will be a dominant mode of information exchange in future systems since they pave the way for combining the capabilities of powerful technologies in existence today – mobile platforms, social networks and knowledge bases. Such a Q&A paradigm for information exchange opens up a new set of research issues. The ability to support personalized and reliable Q&A in unreliable settings is particularly interesting - a particular use case would be one of exploiting the crowd to gather and distribute situational awareness in disasters. Today, several notification systems are in use and being developed to relay critical content to officials and citizens in harm’s way using dashboards, subscription based notification applications etc. [39, 41, 97, 78]. Notification is usually triggered by the publication of a message; a more interactive paradigm is necessary when individuals seek specific information. The above Q&A paradigm enables this higher level of interactivity that is driven by the querier/seeker of information. The ability to provide seamless exchange in real-time and in the presence of faults is another huge challenge. Techniques to exploit the underlying network structure [39] may be extended to enhance the reliability of exchange using resilient overlay networks. Finally, human-oriented Q&A systems are extremely tolerant to inaccuracies in formulation of questions/answers as opposed to search-based systems. For example, a querist asking for the “crowd-level at the campus medical facility” can express the needs loosely; a responder might indicate “I have been waiting in line for an hour” - such loosely formed questions/answers are still useful in this paradigm. Translating this flexibility into the needed quality of queries/answers at lower levels is interesting and can be used to reduce system overheads. For example, the authors of [41] proposes a service for flexible
localization to satisfy the diverse localization quality levels required by different applications; studies in the literature [97, 78] consider the impact of uncertainty of the sources to ensure the returned answers of the queries are acceptable. The SmartSource framework will serve as a starting point for enabling this work.
Chapter 6

Delay-tolerant Information Sharing: Dissemination Layer

In this chapter, we consider efficient data dissemination for the class of delay-tolerant information sharing applications. Figure 6.1 highlights the contributions of this chapter in the design framework for societal scale information sharing. Specifically, we look at online social networks and social media applications, such as Facebook, Twitter, Weibo, etc. We study the problem of supporting efficient access to social media contents on social network sites for mobile devices without requiring mobile users to be online all the time. We develop middleware to take advantage of users’ geo-social knowledge to: (i) rank the social media streams by estimating probability that a given user views a given content item and (ii) invest the limited resources (network, energy and storage) on prefetching only those social media streams that are most likely to be watched when mobile devices have good Internet connectivity. The ranking scheme leverages social network information to drive a logistic regression based technique that is subsequently exploited by in-device or in-network prefetch scheduling algorithms in combination of knowledge of current network/system conditions to determine which social contents are sent to the devices.
To accommodate different deployment requirements, we design an in-device mobile middleware and an in-network broker/proxy middleware as alternatives. The advantage of the mobile middleware is its lightweight nature and easily deployable on mobile platforms. Moreover, it does not require external infrastructure supports. However in-device solutions require mobile devices to pull/retrieve multiple data items (e.g., posts, comments/likes of Facebook posts), and analyze the data to detect new updates. Executing these processes on mobile devices periodically is taxing on resource-constrained mobile devices. When external infrastructure support is not an issue, the broker/proxy based middleware, on the other hand, provides a better approach by employing a system of brokers and proxies in the network, in order to download, analyze, and deliver new updates from social network sites to mobile users.

Based on the concepts, we implement an Android app, oFacebook (offline Facebook), to automatically prefetch user-generated contents in social media streams to provide mobile users uninterrupted access to social networks. We conduct a user study using oFacebook to collect trace data from 10 users, to determine how users access media information on
Facebook via mobile devices. We perform extensive trace-driven simulations to evaluate the proposed in-device and in-network middleware respectively. Our experimental results indicate that our solution exhibits superior viewing performance and energy efficiency for mobile social media apps; it saves energy by 6.9 times for WiFi and 9.1 times for cellular connections.

The rest of the chapter is organized as follows. We describe the problem of efficient offline access to social media and social networks from mobile devices in Section 6.1. Section 6.2 provides an overview of the mobile middleware solution. We present the algorithmic details of the middleware in Section 6.3. In Section 6.4 we describe the motivation and present an overview of the broker/proxy based middleware solution. The details of the solution are presented in Section 6.5. We present our prototype implementation and extensive evaluation in Section 6.6. Finally, the chapter is concluded in Section 6.7.

6.1 Introduction

The phenomenal popularity of social networks, such as Facebook, Twitter, LinkedIn, Google+, and Instagram, has changed the way people interact today. Indeed, many people rely on these social networks to communicate with their friends, family, and community on a day to day basis. The ability to continue these interactions anytime, anywhere seamlessly is quickly becoming commonplace, and users on modern mobile devices expect to not just to access social networks but also exchange rich media contents, such as video, audio, and images, for an enhanced user experience. It is reported that 93% of Android smartphone users in India use social networks on their smartphones [25] and often this is the reason why they purchase smartphones in the first place. In North America, a recent IDC report on smartphone users indicates that 70% of them access Facebook via smartphones, and more strikingly, 40% of users feel connected when using Facebook, only trailing 43% for making voice calls and 49%
for texting [1]. In fact, the main finding of the IDC report is “mobile+social=connectedness”, i.e., people feel isolated without mobile access to social networks.

To ensure this constant connectedness, mobile users subscribe (and pay for) 3G/4G data plans that are often expensive and do not work for a host of reasons including: (a) wireless network availability is sporadic (accessibility of WiFi access points, unpredictable data rates in 3G networks), (b) mobile devices are battery-powered with stringent energy budgets that are easily depleted from constant connectivity to and interaction with WiFi/3G networks, (c) the shared network bandwidth is limited in public locations (where users want to access this information), and (d) dataplans are becoming volume-driven and hence costly.

We observe that the basic need (root cause) of always-on connectedness stems from the assumption that existing mobile social mobile apps, such as mobile Facebook app, expect always-on connectivity. The modus operandi today is that these apps synchronize with social networks when mobile users launch the apps and mobile devices are connected to the Internet.

We believe that offering offline access for these apps is necessary for mobile users in order to interact with social contents when Internet access is not available. There are many scenarios in which the offline access feature is useful. For example, a student takes a subway to his or her school. In the subway, the student does not typically have mobile Internet connections to social network sites, and thus the offline access feature is the only option allowing the student to view and interact with updates from his or her friends that are prefetched before the student steps into the subway.

A simple approach is to share or download social media streams when possible, as if they were traditional media streams from content providers, such as Netflix and Hulu. However, there are key differences between social media and traditional media streams that makes best effort downloads have inferior performance in our case:
• **Personalized preferences:** The demands of traditional media streams are dictated by global popularity (e.g., movies, television), whereas access to rich social media streams are driven by user preferences and content characteristics. Therefore, traditional one-size-fits-all, popularity based content ranking mechanisms do not work for social media streams.

• **Dynamic, variable size contents:** Compared to traditional media, social media sharing is exemplified by more dynamic uploads (by users); the average size of content that is shared during an update is also smaller than that of media content providers. For example, it is reported that majority of YouTube videos are shorter than 10 mins, while traditional video servers offer up to 2-hour videos [159].

• **Sporadic viewing situations:** Dynamic uploads by social network users and asynchronous access (by their friends) result in sporadic access patterns, e.g., users checking Facebook updates at a cafeteria line. Unlike media downloads where the item of interest can be pre-specified by the users, users may wish to view newly uploaded content in new situations, including those where mobile Internet access is unavailable, intermittent, or expensive (e.g., on a bus/train).

We consider the problem of efficient social media access on mobile devices and develop solutions to make the mobile social media experience seamless, personalized, and effective. We achieve this by: (i) *ranking* the social media streams based the probability that a given user views a given content item, and (ii) investing the limited resources (network, energy, and storage) on *prefetching* only those social media streams that are most likely to be watched when mobile devices have good Internet connectivity.

A comprehensive solution for these two tasks must account for multiple factors including network conditions, content characteristics, device status, and users’ social networks. These factors are typically outside the purview of content providers (e.g., current network con-
6.2 A Mobile Middleware Architecture for Social Content Delivery

Fig. 6.2(a) presents the architecture of our proposed mobile middleware (we call it O²SM) that bridges the mobile OS and social network apps running on a user’s mobile device. O²SM systematically detects user-generated rich content from the mobile user’s social media streams (e.g., fresh photos or videos posted by the user’s friends), and intelligently prefetches conditions for a particular user) or the app at the user side (e.g., characteristics of a specific content item). We aim to develop middleware approach to bridge the device OS and network environment to various social networking apps, such as Facebook, Google+, LinkedIn, and Twitter to determine what content to prefetch and when. Such a middleware approach allows us to: (i) reuse key techniques and implementations across multiple devices/users (ii) make better prefetch decisions based on global information, and (iii) control the overhead due to data transfers and environmental sensing.
contents in order to cope with intermittent Internet access. The choice of what to prefetch and when is crucial since the mobile device is limited in resources, such as battery level and storage space - in the social media context, it is imperative that the mobile user’s demands and preferences are taken into consideration while selectively downloading content. Furthermore, energy efficient download is also a must in order to achieve good user experience, e.g., overly-aggressive prefetching decisions may drain the device battery , which prevents the user from using his or her phone for necessary daily activities.

The O²SM middleware consists of five components: (1) system profiler, (2) user activity profiler, (3) meta-data collector, (4) content ranker, and (5) content prefetcher. The system profiler monitors network and battery conditions on the device via various performance metrics, including signal strength, network throughput, and battery level. It also monitors the size of total downloads to avoid filling up the storage space. Based on the current network conditions and past history of network profiles, the system profiler forecasts the network connectivity and throughput in the near future. The user activity profiler monitors the user’s activities on the device. In particular, it monitors the user’s access patterns to social media streams, e.g., when and how long the user navigates the social media streams, and forecasts the user’s future accesses. This information can be collected from the history log of the social media apps built atop the middleware or upon explicit user input. For example, if the user plans to go out for lunch in an hour, the user can request the middleware to aggressively prefetch social media contents before, so that he/she has something to watch while waiting in the restaurant.

The meta-data collector periodically pulls the meta-data from each of the social media streams to get updates of new rich contents since the last pull. This is done when the network is available and the remaining battery level is above an operational threshold, which is configurable by the user. The meta-data is usually small in size. Fig. 6.2(b) shows an example meta-data request and its JSON responses for a media content using Facebook.
Query Language (FQL) [11]. Our experiments show that the average meta-data size is around 700 bytes per content, and is negligible when compared with the typical data size of 140 KB of a photo or that of 3 MB of a 1.5-minutes video. The collected meta-data is saved onboard the meta-data database for later processing.

To support intelligent content prefetching, the content ranker invoked by the content prefetcher queries the database on fresh content items that have not been downloaded by the system as yet, and predicts the probability the user would view them. The ranking algorithm is based on social relations between entities in the social networks, user interests and content popularity. The ranked contents are fed to the content prefetcher to make decisions on which contents to prefetch and when to prefetch. The crux of this component is a cost-benefit driven prefetch scheduling algorithm which aims to maximize the overall prefetching gain (benefit-cost) by intelligently selecting contents to download and scheduling their download time. This is achieved by carefully taking into account the variations in the sizes of the contents (from content meta-data), their expected probability of being watched (from the content ranker), predicted network conditions such as the network connectivity and bandwidth (from the system profiler), as well as predicted user activity such as when the user will navigate his/her social media streams (from the user activity profiler). Once the contents are scheduled for downloads, they are inserted into a download queue. The downloader downloads the contents in the queue based on their scheduled times.

6.3 O2SM Mobile Middleware Algorithms

6.3.1 Social Media Content Ranking Component

Given resource constraints at mobile devices, a method to efficiently rank social media contents based on their importance is critical. Prefetching all media contents that accompany
heavy multimedia objects, e.g., images or videos, could not only be harmful to user, when he or she is in need of phone battery to survive in daily work, but also waste resources if the downloaded contents are not later viewed by user. This section is dedicated to present the background and our approach to develop a ranking system that works efficiently for mobile devices and scalably against the arrival of new contents.

**Background:** Traditional ranking systems can be classified into three categories: (1) content based approaches [28, 73, 58], (2) collaborative filtering approaches [43, 140, 115], and (3) social network based approaches [111, 128, 116]. Content based approaches [28, 73, 58] rank media contents using the correlation between the attributes of media content, such as descriptions, keywords, tags, images, and video features (e.g., colors), and user preferences. The basic process here matches up the attributes of the user profile (with preferences, interests), with the attributes of contents. The performance of this category is limited by dictionary-bound relations between the keywords obtained from users and the descriptions of media contents. More comprehensive feature extraction (e.g., image and video analysis) is expensive and hence not supported by most social media.

Collaborative filtering approaches [43, 140, 115] recommend media contents by first calculating the similarity between all users in the system based on their previous ratings of media contents. Ratings are, for example, represented using numeric scores from 1 to 5 and similarity is estimated using heuristic measures, such as the well known cosine function. The system, then, projects a ranking a user is likely to give a piece of recommended content by aggregating the ratings of the user’s k nearest neighbors on the content. Collaborative filtering approaches are useful for highly sparse data where the matrix (ratings) is partially missing. Issues for these approaches include cold start, i.e., all ratings are missing, and high computation complexity, i.e., a large number of users and contents involves an inference process, that is critical due to limited resources on mobile devices.

The last category ranks social media contents via social networks. Social networks are
characterized by heterogeneous entities, e.g., contents, users, and social relations. Recent studies [111, 128, 116] exploit relations among entities to recommend contents. For example, a user is likely to view contents generated by friends, whom the user interacts with frequently over social networks. The approaches in this category fit our system’s goals best because they could utilize the underlying social relations captured in our system. However, while the intuitions are appealing, a direct application of the existing techniques to our system is not easy. For example, the work in [111] is based on a social graph to evaluate social relations among entities, and works only for a fixed set of entities. It requires rebuilding the social graph and recalculation of weights when new contents arrive, which happens frequently in our system. Extensions to the collaborative filtering concept [128, 116] by employing relations to estimate similarity have been studied, they suffers from high computation complexity that are unsuitable to mobile deployment.

**Our Approach:** We inherit the spirit of the social network based recommendation systems [111, 128, 116] to rank social media contents based on social relations and interactions, but targets to work efficiently on mobile devices and be scalable to the arrivals of new contents. Besides, we take user-poster interests and content popularity into account to improve the ranking accuracy. In some sense, user-poster interests and content popularity can be considered as indirect content based features because they capture user preferences and content impacts on viewers. The main design principles for the ranking component in our mobile system are *light weight* and *fast provisions of ratings on newly incoming contents*.

Our approach first identifies and constructs social based features that can infer user-content interactions. We then employ a supervised learning approach to predict the probability of a social media content viewed by a user. We develop our ranking component to support content ranking on Facebook, but the approach is general enough to be readily applied to other social networks, e.g., Twitter as presented in [75].

We construct the following features to capture the underlying social relations. Let $u$ be a
user with a friend $f$. Table 6.1 presents key interaction types between $u$ and $f$.

1. **Post interactions**: This feature captures the interactions between the user and a friend of the user via posts on their social media sites, e.g., Facebook Wall page. It is clear that $u$ is likely more interested in $f$’s posts if interaction between $u$ and $f$ is high. We use the total number of interactions between $u$ and $f$ to quantify this feature. Post interactions include post($u, f$), post($f, u$), comment($u, f$), comment($f, u$), like($u, f$), and like($f, u$) as shown in Table 6.1. In Facebook, a user can post message on his or her friend’s Wall page or tag the friend in a photo. A Twitter user can retweet an interesting message or @reply a tweet from some person the user follows. All of these are considered post interactions. Some social networks, for example Facebook, allow users to subscribe or like a page (e.g., a soccer club like Manchester United or a university like Harvard University). Any updates from the page will be sent to the subscribers similar to those from a friend. For simplicity, we consider a page as a friend of $u$.

2. **Private message exchange**: Different from post interactions, which may be available in public, private messages are only available to the recipients. We use the total number of messages sent and received between user $u$ and friend $f$ of $u$ to quantify the feature

<table>
<thead>
<tr>
<th>Interaction Type</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post($u, f$)</td>
<td>User $u$ writes a post on poster $f$’s Wall page, tags $f$ in a photo, a video or an album of images.</td>
</tr>
<tr>
<td>Post($f, u$)</td>
<td>Vice versa.</td>
</tr>
<tr>
<td>Comment($u, f$)</td>
<td>User $u$ comments on a post from $f$, or on a tag of a photo, a video or an album from $f$.</td>
</tr>
<tr>
<td>Comment($f, u$)</td>
<td>Vice versa.</td>
</tr>
<tr>
<td>Like($u, f$)</td>
<td>User $u$ likes a post uploaded by $f$, or an image, a video or an album tagged by $f$, or $u$ likes a comment from $f$.</td>
</tr>
<tr>
<td>Like($f, u$)</td>
<td>Vice versa.</td>
</tr>
<tr>
<td>Message($u, f$)</td>
<td>User $u$ sends a private message to $f$.</td>
</tr>
<tr>
<td>Message($f, u$)</td>
<td>Vice versa.</td>
</tr>
</tbody>
</table>

Table 6.1: Representation of user interactions.
(i.e., message($u, f$) and message($f, u$) in Table 6.1). The higher number of exchanged messages, the higher the interaction level between $u$ and $f$ and higher the probability that $u$ views $f$’s posts. In Facebook, users can send and receive private email-like messages l to and from their friends (and even strangers). Twitter and Weibo users can communicate with each other through direct messages with a limited number of characters.

In addition to the above features, we extract and represent two others features - the level of interest of a user $u$ has to a specific friend $f$, and the popularity of a post.

1. **User interest w.r.t to a friend**: We measure the interest of $u$ w.r.t to a friend $f$ of $u$ by the number of view clicks to contents posted by $f$. The higher number of view clicks indicates a higher interest level of $u$ to $f$.

2. **Post popularity**: A post from friend $f$, with a high number of comments and likes, is likely interesting because it receives high attention from many viewers. It is therefore likely to be interesting to $u$ even the viewers are not $u$’s friends. We thus employ the number of comments and likes in a post to predict the viewing probability.

In order to predict the probability a content is viewed by a user, we employ a well-known supervised learning algorithm - logistic regression classifier. Logistic regression is an additive model used to predict the probability of a discrete event given a set of explanatory variables. It weights the impact of all constructed features with estimated coefficients. In the literature, the logistic regression has been used for prediction, such as friendship strength prediction [100]. We are the first applying the logistic regression to predict the probability of user-content interactions based on the social networks.

Now we describe how to apply the logistic regression to our content ranking component. The
logistic regression in the training process learns a model of the form:

\[ p(y_i|x_i) = \frac{1}{1 + e^{-(c_0 + \sum c_j x_{ji})}}, \]  

(6.1)

where \( x_i \) is a vector of the features \( (x_{i1}, ..., x_{in}) \) described above for content \( i \), and \( y_i \) is 1 if the user clicks to view content \( i \). Otherwise, \( y_i \) is 0. The model is represented by a vector of coefficients \( c \), in which \( c_0 \) is not multiplied with any feature and added to the vector as noise. In the training process, the model is learned by maximizing the log-likelihood of the logistic regression model. A well-known approach for the maximization problem is a trust region Newton method [110]. The resulting model is then used to evaluate the viewing probability for a newly arriving media content by simply applying Eq. (6.1) to the features of the new content. In the evaluation section, we will show that the training process that calculates the model parameters for our ranking component is indeed light weight and fast. This technique is attractive in our setting since the processing of newly arriving contents essentially reduces to that of applying Eq. (6.1). Furthermore, the simplicity of the scheme lends itself to be easily deployed on mobile devices.

### 6.3.2 Ranking-driven Social Media Prefetching

Prefetching social media contents has received attention at CDNs. [150] considered the problem of efficient geo-replicating contents across multiple data-centers spread around the world. However, little research has been done for prefetching social media contents to mobile clients. Mobile prefetching is an increasingly relevant topic of research. Techniques have been developed for determining when to prefetch based on network conditions such as WiFi and cellular signal strength [137, 66, 141]. Minimizing energy consumption during prefetching is critical for mobiles - studies [32, 72] indicate that WiFi is typically more efficient than cellular networks, and techniques to aggregate multiple data transfers to save energy have
been developed. [52, 109] exploit social networks to assist prefetching video prefixes. In general, prefetching has been explored in client-server and peer-to-peer settings using factors such as energy, transfer volume and user preferences.

Much of the earlier work focuses on what to prefetch and does not explicitly account for whether the prefetched content is consumed or not. A more comprehensive prefetching scheme must incorporate which items to prefetch for maximum benefit. [160] observes that users tend to launch a fairly fixed sequence of the apps, and user locations have a strong correlation with app usage so a decision engine was proposed to determine which apps to prefetch. In contrast to the above approaches, we propose a comprehensive scheme that exploits the previously described ranking mechanism to determine what to prefetch and develops an efficient scheduling technique for when to prefetch.

The O²SM prefetcher divides time into prefetching period or slot (e.g., 15 minutes), say $T_{\text{prefetch}}$, and runs the prefetch scheduling algorithm at the start of every slot. The algorithm selects a number of contents for download in each prefetching period optimizing for large time-scale prefetching. Algorithm 6 shows the flow of the scheduling algorithm. The goal of the algorithm is to provide the best viewing experience when the user navigates his/her social media streams, while keeping the prefetching cost low.

The O²SM prefetcher strives to balance the potential benefit of prefetching against its cost: On one hand, prefetching social media contents can provide users with offline viewing experience and hide the latency of data transfers over poor and intermittently connected wireless networks. One the other hand, if the prefetched content is never viewed, prefetching consumes resources (e.g., energy and cellular data usage) that could be used for other activities. Since the prefetching benefit is different for different contents and the prefetching cost of the same content varies over time when network condition changes, the prefetch scheduling algorithm determines what to download and when to download by formulating a Offline Online Social Media Prefetch Scheduling (O²SMPS) problem. In the following sections we describe
Algorithm 6: Prefetch Scheduling

**Step 1-Read Unviewed Ranked Contents:**
read the list of unviewed ranked contents output from the Content Ranking component.

**Step 2-\(K\) Slot-ahead Forecast:**
forecast the network conditions and user context for the next \(K\) slots.

**Step 3-Offline Online Social Media Prefetch Scheduling (O\(^2\)SMPS):**
formulate an O\(^2\)SMPS problem using a cost/benefit analysis to decide which contents to download in each of the next \(K\) slots.

**Step 4-Contents Download:**
sequentially download contents that are scheduled for the current slot.

the cost/benefit modeling and the O\(^2\)SMPS problem in details, and discuss the forecasting techniques used in our system.

**Cost/Benefit Modeling**

Let \(I = \{i_1, i_2, \ldots, i_{|I|}\}\) be the list of ranked contents and \(q_i\) is the likelihood of content \(i\) will be viewed by the user. We evaluate the cost/benefit of content prefetching in the next \(K\) slots to determine the best prefetching time by forecasting the network conditions and user activity in the future. Let \(t = 1, 2, \ldots, K\) be the slotted time for the next \(K\) prefetching slots. The system monitors the use of the 3G and WiFi network interfaces for Internet access. Let \(p_{\text{wifi}}(t)\) be the probability that the device uses the WiFi interface for data transfer at slot \(t\), and \(bw_{\text{wifi}}(t)\) is the corresponding download bandwidth. Similarly, \(p_{\text{cell}}(t)\) and \(bw_{\text{cell}}(t)\) are the probability and bandwidth for the 3G interface at slot \(t\). The system also predicts how likely the user will actively navigate his/her social media streams. Let \(p_{\text{nav}}(t)\) denote the predicted likelihood that the user may become active at slot \(t\). We discuss the network and user activity profiling and prediction techniques in the Sec. 6.3.2.

Inspired by [80], we estimate the cost and benefit of prefetching each content along three dimensions: *viewing performance*, *energy use*, and *data plan use*. We consider two costs involved in prefetching: the *energy cost* and the *cellular data plan cost*. The cost for prefetching
a content may vary over time when the network conditions change. Formally, we define the estimated cost of prefetching a content \(i\) at slot \(t_{pre}\) as:

\[
C(i, t_{pre}) = w_e \times C_{\text{Energy}}(i, t_{pre}) + w_d \times C_{\text{cell}}(i, t_{pre}),
\]

where \(w_e\) and \(w_d\) are weighting coefficients for energy and cellular data consumption respectively. The coefficients are configurable to the user. For instance, if the user uses an unlimited cellular data plan so that the cellular data usage is not a concern, then \(w_d\) can be set to zero. On the other hand, if the user is discreet about energy use, he may prefer a large value for \(w_e\).

The benefit of prefetching is threefold: the viewing performance benefit, the energy benefit and the cellular data plan benefit. The viewing performance benefit comes from the improved user experience when the user views a prefetched content. The energy and the data plan benefits are the saves in the energy use and data plan use that would be otherwise consumed when the user views the content. Clearly, the benefit depends on: (i) probability of the content to be viewed and (ii) network condition at the time when the user navigates the streams. Formally, we define the estimated benefit to prefetch a content \(i\) in case the user navigates his/her social media streams in slot \(t_{nav}\) as:

\[
B(i, t_{nav}) = B_{\text{view}}(i, t_{nav}) + w_e \times B_{\text{energy}}(i, t_{nav}) + w_d \times B_{\text{cell}}(i, t_{nav}).
\]

We next define \(B_{\text{energy}}(\cdot)\). We adopt the energy models developed by PowerTutor [164]. For downloading a content under WiFi at slot \(t\), the energy cost is modeled as \(e_{\text{wifi}}(i, t) = c_{\text{wifi}} \times \frac{s_i}{bw_{\text{wifi}}(t)}\), where \(c_{\text{wifi}}\) is a power coefficient for WiFi interface. For downloading under 3G, the energy cost is \(e_{\text{cell}}(i, t) = c_{\text{cell}} \times \frac{s_i}{bw_{\text{cell}}(t)} + e_{\text{tail}}\), where \(c_{\text{cell}}\) is a power coefficient for 3G and \(e_{\text{tail}}\) is an estimated 3G tail energy cost. Since contents are prefetched in batches, to estimate the tail energy cost we let \(e_{\text{tail}} = c_{\text{tail}} \times \frac{T_{\text{tail}}}{l_{\text{avg}}}\) where \(c_{\text{tail}}\) is the power coefficient.
for 3G tail energy, $T_{tail}$ is the typical 3G tail time (usually > 10 seconds), and $l_{avg}$ is the history average of the number of contents downloaded in a batch by the prefetcher. To predict the 3G tail energy consumption for on-demand fetches from the user, we let $e_{tail} = c_{tail} \times \min(T_{inactive}, T_{tail})$ where $T_{inactive}$ is the history average of the idle period between two consecutive content requests from the user. Then, the expected energy to download a content $i$ at slot $t$ is:

$$E(i, t) = \frac{p_{\text{wifi}}(t)}{p_{\text{wifi}}(t) + p_{\text{cell}}(t)} \times e_{\text{wifi}}(i, t) + \frac{p_{\text{cell}}(t)}{p_{\text{wifi}}(t) + p_{\text{cell}}(t)} \times e_{\text{cell}}(i, t).$$

Since the prefetching energy cost $C_{\text{energy}}(i, t_{\text{pre}}) = E(i, t_{\text{pre}})$, the energy benefit becomes $B_{\text{energy}}(i, t_{\text{nav}}) = q_i \times E(i, t_{\text{nav}})$, which takes the probability of the content to be viewed into consideration.

We define $B_{\text{cell}}(\cdot)$ in a similar way. The expected cellular data plan use for downloading a content $i$ at $t$ is:

$$D(i, t) = \frac{p_{\text{cell}}(t)}{p_{\text{wifi}}(t) + p_{\text{cell}}(t)} \times s_i.$$ 

Therefore, the data plan cost $C_{\text{cell}}(i, t_{\text{pre}}) = D(i, t_{\text{pre}})$ and data plan benefit $B_{\text{cell}}(i, t_{\text{nav}}) = q_i \times D(i, t_{\text{nav}})$.

The viewing performance benefit is considered as the granted feasibility for offline access if the user desires to view a content when he/she doesn’t have network access. On the other hand, if the user checks the content when he/she has network access, the benefit is the hidden latency of the on-demand content downloading/buffering. Assume the download bandwidth at the time when the user requests the content is $bw$, the hidden latency can be formulated as $d(i) = \min(s_i, s_{\text{max}}) / bw$, where $s_{\text{max}}$ is the maximum playback buffer size for videos, or $s_{\text{max}} = s_i$ for non-video content. Then, taking into account the viewing probability of the content as well as the predicted network conditions, for any time slot $t_{\text{nav}}$ that the user
may actively navigate social media contents, the expected viewing performance benefit for prefetching content $i$ is:

$$B_{\text{view}}(i, t_{\text{nav}}) = p_{\text{wifi}}(t_{\text{nav}}) \times q_i \times d_{\text{wifi}}(i, t_{\text{nav}}) + p_{\text{cell}}(t_{\text{nav}}) \times q_i \times d_{\text{cell}}(i, t_{\text{nav}})$$

$$+ w_{\text{off}} \times (1 - p_{\text{wifi}}(t_{\text{nav}}) - p_{\text{cell}}(t_{\text{nav}})) \times q_i,$$

where $d_{\text{wifi}}(i, t_{\text{nav}})$ and $d_{\text{cell}}(i, t_{\text{nav}})$ are the predicted hidden latency under WiFi and 3G respectively, and $w_{\text{off}}$ is a relative benefit weight of viewing offline to viewing online, that can be customized by the user.

### O$^2$SMPS Problem

The O$^2$SMPS problem maximizes the prefetching gain as the benefit minus cost, by allocating contents for downloads in each of the next $K$ slots. We define a prefetch scheduling matrix, $Z = \{z_{i,k}\}$, where $z_{i,k} \in \{0, 1\}$, such that $z_{i,k} = 1$ if download content $i$ at slot $k$ and $z_{i,k} = 0$ otherwise. Furthermore, let $y_i(k)$ be another 0-1 variable that indicates whether a content $i$ has been downloaded before slot $k$. It is easy to see that $y_i(1) = 0$ and $y_i(k) = \sum_{l=1}^{k-1} z_{i,l}$ under the constraint that $\sum_{k=1}^{K} z_{i,k} \leq 1$, i.e., a content can not be scheduled for download more than once.

To be useful, a content must be prefetched before the user checks his/her social media streams. Taking into account $p_{\text{nav}}(t)$, i.e., the likelihood that the user may actively navigate social media contents at each of the $K$ slots, we can calculate an expected prefetch scheduling benefit for any scheduling $Z = \{z_{i,k}\}$ as:

$$\text{Benefit}(Z) = p_{\text{nav}}(1) \times \sum_{i=1}^{\left|I\right|} (B(i, 1) \cdot y_i(1))$$

$$+ (1 - p_{\text{nav}}(1)) p_{\text{nav}}(2) \times \sum_{i=1}^{\left|I\right|} (B(i, 2) \cdot y_i(2))$$

$$+ \cdots + \prod_{k=1}^{K-1} (1 - p_{\text{nav}}(k)) p_{\text{nav}}(K) \times \sum_{i=1}^{\left|I\right|} (B(i, K) \cdot y_i(K)),$$
where each term on the right side of the equation specifics the expected prefetching benefit if the next time when the user navigates his/her social media streams is at the $k^{th}$ slot. On the other hand, the expected prefetch scheduling cost for the prefetch scheduling $Z$ is:

$$\text{Cost}(Z) = \sum_{i=1}^{\mid I \mid} \sum_{k=1}^{K} C(i, k) \cdot z_{i,k}.$$ 

Now we formally define the O$^2$SMPS problem as:

\begin{align*}
\text{maximize} & \quad \text{Benefit}(Z) - \text{Cost}(Z) \\
\text{subject to} & \quad \sum_{i=1}^{\mid I \mid} s_i \cdot z_{i,k} \leq T_{\text{prefetch}} \cdot bw(k) \quad k = 1, \ldots, K; \quad (6.2b) \\
& \quad \sum_{k=1}^{K} z_{i,k} \leq 1 \quad k = 1, \ldots, K; \quad (6.2c) \\
& \quad z_{i,k} \in \{0, 1\} \quad i = 1, \ldots, \mid I \mid; k = 1, \ldots, K, \quad (6.2d) 
\end{align*}

where (6.2b) specifies that the total amount of data can be downloaded in a prefetching slot is constrained by the average download bandwidth in the slot.

The above problem can be reduced to a Generalized Assignment Problem (GAP) [118] that assigns $|I|$ items to $K$ bins and the value of each item varies for different bins it puts in. The problem is known to be NP-hard, and its efficient approximation algorithms has been studied extensively in literature. In this work, we adopt the polynomial-time algorithm by Martello and Toth [117] that provides an approximate solution to the problem under the overall time complexity of $O(K|I|\log K + |I|^2)$.

The algorithm has two phases. The first phase tries to provide a reasonably good assignment uses a measure of the desirability of assigning content $i$ to slot $t$, say $g_{i,t}$. It iteratively considers all the unassigned contents, and picks the content with the maximum difference between the largest and second largest $g_{i,t}$ to get assigned first. The intuition is such content is most critical since failing to assign it into its best slot will negatively impact the overall
performance most. We let $g_{i,t} = \frac{f_{i,t}}{s_i}$, where $f_{i,t}$ is the gain of assigning content $i$ to slot $t$ derived from $F$, and $s_i$ is the size of the content. So $g_{i,t}$ is a measure of the \textit{unit gain} of the content $i$ in slot $t$. In the second phase, once all contents have been assigned, the solution is improved through local exchanges. Readers are referred to [117] for the algorithm details.

\section*{Forecasting Network Connectivities and User Activity}

The prefetcher relies on two predictions for each of the prefetching slots to make scheduling decisions: (a) a network prediction in terms of the connectivity probability distribution vector $\vec{p}_{\text{net}}(k) = [p_{\text{wifi}}(k), p_{\text{cell}}(k), (1 - p_{\text{wifi}}(k) - p_{\text{cell}}(k))]$ as well as the bandwidth vector $\vec{bw}(k) = [bw_{\text{wifi}}(k), bw_{\text{cell}}(k), 0]$; and (b) a user activeness prediction as the likelihood $p_{\text{nav}}(k)$ that the user might navigate his/her social media streams in the slot.

Because people are creatures of habit, many existing profiling and forecasting techniques (e.g., location-based or time series prediction) can be used by both prediction problems to achieve highly accurate predictions. A notable technique is BreadCrumbs [127], which is a location-based prediction scheme. It tracks the movement of the mobile device and utilizes a simple Markov model to generate connectivity forecasts. Their evaluation results indicate a very good accuracy. The technique requires the location information either from the device’s GPS or techniques like Place Lab [95] where the device has to communicate with a remote server to extract its location information from the information of its current WiFi access point and cellular towers. To mitigate this constraint, we applied a another simple Markov model based technique when the location information is not generally available.

We use a time-dependent Markov model for forecasting. The technique is applied to both network connectivity and user activity forecast. To forecast network connectivity, the states of the Markov model are \textit{WiFi}, \textit{cellular} and \textit{offline}. To forecast user activeness, the states are \textit{active} and \textit{inactive}. The transition matrix depends on the time of the day. Since the
prefetch schedule is slotted, say $N$ slots a day, we maintain the same number of transition matrix to capture the probability of a transition from a state at slot $t$ to a state at slot $t+1$. For each state in the model and time boundary between slots, the prediction component updates the corresponding Markov transition matrix whenever the model is in the state and transitions to another or the time is moved to a new slot. The future predictions can then be made from the trained transition matrix. For example, let $A(k)$ be the transition matrix for transition from slot $k$ to slot $k+1$, given the network connectivity at slot $k$ as $\overline{p_{net}}(k)$, we can approximate the network connectivity $k+1$ and slot $k+2$ as $\overline{p_{net}}(k)A(k)$ and $\overline{p_{net}}(k)A(k)A(k+1)$ respectively. Meanwhile, the average bandwidth vector $\overline{bw}(k)$ can be easily derived for each time slot.

### 6.4 A Broker/Proxy Framework for Social Content Delivery

The mobile middleware approach provides an easily deployable solution for efficient social content delivery independent of external infrastructures. This approach, however, is still relatively expensive in terms of the consumed data plan quota and mobile device energy, which is caused primarily by the limitations in the current APIs of social network providers.

In order to determine whether there are new updates from social network sites, mobile device has to set up connections, and send requests for new updates. If there are no new updates, the requests are useless, and device resources (i.e., energy) consumed for sending the requests are wasteful. In some cases, mobile devices even have to pull/retrieve multiple data items (e.g., posts, comments/likes of Facebook posts), and analyze the data to detect new updates. Executing these processes on mobile devices periodically is taxing on resource-constrained mobile devices. A better approach is to employ a system of brokers and proxies in the network to help mobile devices to optimally prefetch social media contents. Fig. 6.3
illustrates the considered broker/proxy architecture where mobile users and social media providers exchange data by staging relevant content intelligently with the supports from the broker/proxy nodes. The goal of the broker/proxy architecture is to maximize the viewing likelihood and quality of experience of the prefetched social media contents while maintaining certain energy level at mobile devices for user daily activities.

In this system, the key coordination is performed at a broker – social media users on mobile devices register with the system through this brokerage service. The broker service periodically checks and tracks social media updates for each mobile user, calculates the relevance of every update to the user and generates an annotated list of updates, enhanced with the calculated relevance factors. A broker may manage one or more proxies, and each proxy in turn serves one or multiple mobile devices. The broker passes the annotated list of new contents to each mobile device’s proxy and also initiates the actual downloads of the updated contents onto the respective network proxies (usually, in close vicinity of the client). In general, proxies may be installed by mobile users, cellular carriers, content distribution networks, social networks, and third-party companies.
We assume that proxy selection is done by individual mobile users, who may have lists of trustworthy proxies in mind or prefer to use closer proxies for higher network bandwidth. More sophisticated policies for proxy selection are possible [136] - for example multiple brokers/proxies may be employed for scalability and mobility handling - the design of such policies are out of the scope of this work. We aim to design, implement, and evaluate a broker/proxy solution that schedules mobile prefetches to achieve these goals and relieve the mobile device from the associated computation and communication overhead.

Fig. 6.4 illustrates the software modules of the proposed system. The broker/proxy periodically checks for updates from social network sites and efficiently delivers the updates to mobile users. The crux of the system resides in two key modules that are described below: (i) user-content relevance evaluator, which estimates the likelihood of a content being watched by a user (based on social connections) and (ii) scheduling algorithm, which performs the actual delivery of contents to mobile users such that the benefits of prefetching contents is maximized under the resource constraints.

**User-content relevance evaluator.** This module estimates the likelihood a social con-
tent will be viewed by a user using social network information. The resulting user-content relevance provides hints to the scheduling and delivery modules in the proxy for resource-efficient delivery. We applied the same content ranking technology in our mobile middleware solution here.

**Scheduling algorithm.** The second key component is the *Scheduling Algorithm module* that executes on the proxy. The challenge here is how to utilize as few as possible knowledge from mobile devices to efficiently schedule the delivery of contents from the proxy to the mobile device. We want to minimize the periodic communications of knowledge from the mobile device to the proxy to save energy on the mobile device. Updates from social network sites for a user are downloaded to the selected proxy via the *Download and Storage module*. The mobile device connects to the proxy only when it detects a good network condition and enough battery level at the device. These system conditions are reported to the *Device Monitor* at the proxy. Decisions are made by the scheduling algorithm running on the proxy to perform the prefetch of relevant updates based on the reported network conditions and battery levels at the mobile device and the *Delivery module* implements the content transfer. The details of the scheduling algorithm are presented in the next section.

### 6.5 Broker based Content Scheduling Algorithm

In this section, we formulate and solve the scheduling problem of prefetching social contents from the proxy to the mobile device optimally.

#### 6.5.1 System Models and Problem Formulation

We divide time into slots. Every time slot, the scheduling algorithm selects which contents will be delivered to mobile device.
Content arrival. A new content $c$ generated by a friend of $u$ is fetched by the proxy and stored in the proxy’s database before it can be delivered to the user. Content $c$ includes metadata (e.g., post_id, author_id, description, source links of images and videos, and etc.) and media content (images and videos). Let us denote the size of content $c$ (including metadata and media content) as $s_c$, and the total size of all contents for the user fetched by the proxy in time slot $t$ as $\lambda(t)$. We have $\lambda(t) = \sum_{c \in a(t)} s_c$, where $a(t)$ is a set of contents arriving to the proxy in $t$. Assume that $\lambda(t)$ is finite and limited by $\lambda_{\text{max}}$, i.e., $\lambda(t) \leq \lambda_{\text{max}} \forall t$. The time averaged fetched data amount is defined as:

$$\bar{\lambda} \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\lambda(\tau)\}.$$  \hspace{1cm} (6.3)

Content departure and dropping. Every time slot, the scheduling algorithm chooses a subset of the fetched contents to deliver to the user. Let $r(t)$ be a set of contents scheduled to be delivered in time slot $t$, and $\mu(t)$ be the total delivered data amount, i.e., $\mu(t) = \sum_{c \in r(t)} s_c$. We assume $\mu(t) \leq \mu_{\text{max}} \forall t$. It is not hard to see that $\mu(t)$ is limited by the mobile device’s downlink bandwidth at $t$, defined as $\theta(t)$, i.e., $\mu(t) \leq \theta(t)$. We also assume $\theta(t) \leq \theta_{\text{max}} \forall t$. The time averaged data amount delivered to the user is defined as:

$$\bar{\mu} \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu(\tau)\}.$$  \hspace{1cm} (6.4)

Some fetched contents are dropped by the proxy due to low viewing probability or large size (hence, high energy consumption). Let $d(t)$ and $\chi(t)$ be a set of dropped contents and the total dropped data amount in time slot $t$, i.e., $\chi(t) = \sum_{c \in d(t)} s_c$. The time averaged data amount dropped from the fetched content list is:

$$\bar{\chi} \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\chi(\tau)\}.$$  \hspace{1cm} (6.5)
**Energy budget and consumption.** In our system, each user sets an energy threshold $e_t$ and nothing is prefetched if the current energy level, denoted as $e(t)$ is less than $e_t$. Moreover, we allow mobile users to set energy budget for prefetching contents, in order to preserve battery levels for other daily activities, such as making phone calls and browsing Web pages. The energy budget is set for user-specified time periods, e.g., a user may set the energy budget to 10% of the current energy for a period between 8 a.m. to 6 p.m., and to 30% between 6 p.m. to 8 a.m. In particular, we let $t_s$ and $t_e$ be the starting and ending times of a period. Assume period $[t_s, t_f]$ consists of a set of time slots $\{t_0, t_1, \ldots\}$. Let us define $e_a$ as the available energy amount at the device at $t_s$, and $e_u$ as the usable energy percentage for the system to download social contents in the period. The energy budget at the beginning of the period is $e_p = [(e_a - e_t)e_u]^+$, where $[x]^+ \triangleq \max(0, x)$. For every content download, the consumed energy is deducted from the energy budget. The energy budget may increase if the phone is charged. In time slot $t$, if the phone is charged and the current energy level is larger than $e_t$, $e_u$ percentage of the charged energy will be added to the energy budget. Let $w(t)$ be the charged energy in time slot $t$. The energy added to the budget is $\varepsilon(t) = w(t)e_u$. We write $\varepsilon(t_0) = e_p + w(t_0)e_u$ for $t_0$, and define the time averaged energy added to the budget as:

$$
\bar{\varepsilon} \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\varepsilon(\tau)\}. \quad (6.6)
$$

We next model the energy consumed by mobile device for receiving contents from the broker/proxy. Let $\rho_c(t)$ be the energy amount used for receiving content $c$ in time slot $t$. We employ the energy models from [33] to estimate $\rho_c(t)$, based on the content, network type, and network condition. The total energy consumption in time slot $t$ is $\rho(t) = \sum_{c \in r(t)} \rho_c(t)$, where $r(t)$ is the contents delivered to the mobile device in $t$. The time averaged energy
consumption amount is written as:
\[
\bar{\rho} \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\rho(\tau)\}.
\] (6.7)

**Utility model.** Let \( p_c \) be the probability a user \( u \) will view \( c \), which is between 0 and 1. The closer to 1 \( p_c \) is, the more likely \( u \) views \( c \). We consider time decay of contents, i.e., a more recent content interests users more. Let \( U_c(t) = f(p_c, t) \) be the time decay utility function that models the benefit to deliver content \( c \) to the user in time slot \( t \), which is a decreasing function on \( p_c \) and \( t \). In a time slot, some contents are delivered to the user while some are dropped, or the proxy may be totally filled up. Let’s denote \( U_r(t) \) and \( U_d(t) \) as the total utilities of contents which are delivered and dropped, respectively. Mathematically, \( U_r(t) = \sum_{c \in r(t)} U_c(t) \) and \( U_d(t) = \sum_{c \in d(t)} U_c(t) \).

**Problem formulation.** The scheduling algorithm selects a set of contents \( r(t) \) to deliver to the mobile device and a set of contents \( d(t) \) to drop from the proxy based on the following formulation.

\[
\begin{align*}
\max: \quad & \bar{U} = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\left\{U_r(t) - \alpha U_d(t)\right\}; \\
\text{st}: \quad & \bar{\lambda} \leq \bar{\mu} + \bar{\chi}; \quad (6.8a) \\
& \bar{\rho} \leq \bar{\varepsilon}; \quad (6.8b) \\
& \sum_{c \in r(t) \cap r(t) \in f(t)} s_c \leq \theta(t); \quad (6.8c) \\
& \sum_{c \in r(t) \cap r(t) \in f(t)} \rho_c(t) \leq \min(e(t) - e_t, P(t)). \quad (6.8d)
\end{align*}
\]

The objective function in Eq. (6.8a) maximizes the long term utility for content delivery and minimizes the long term utility for content dropping. The proxy drops some contents to stabilize the system as shown in constraint in Eq. (6.8b). Weight \( \alpha \) (\( \alpha \geq 1 \)) is added to
the dropping utility to force the system to consider more seriously when making a dropping decision. Constraint Eq. (6.8c) ensures that the long term energy budget is enough to perform scheduled transfers. Constraint Eq. (6.8d) ensures that bandwidth is enough for data transfer at any time slot. And the last constraint in Eq. (6.8e) makes sure that energy is enough for data receipt from the broker/proxy (note that, \( e(t) - e_t \) is the difference of the current energy level with the power threshold, and \( P(t) \) is the remaining amount in energy budget that will be modeled in the next part).

### 6.5.2 An Optimal Algorithm

We design an algorithm to solve the problem in Eqs. (6.8a)–(6.8e) using the fine-grained queue control theory – Lyapunov optimization [74], which is useful for solving long-term maximization problems. We start by presenting our design principles, and then the detailed algorithm.

**Algorithm design principles.** We use a real queue \( Q \) and a virtual queue \( P \) to represent constraints in Eqs. (6.8b) and (6.8c) for data and energy, respectively. \( Q \) stores incoming contents downloaded to the proxy. The contents in \( Q \) are taken out of the queue for either delivery or drop. The evolution of \( Q \) over time is written as:

\[
Q(t + 1) \triangleq [Q(t) - \mu(t) - \chi(t) + \lambda(t)]^+. \tag{6.9}
\]

We define the stability of the \( Q \) as:

\[
\bar{Q} \triangleq \lim_{t \to \infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{Q(\tau)\} < \infty. \tag{6.10}
\]

At \( t = 0 \), we set \( Q(t) = 0 \). The constraint in Eq. (6.8b) is satisfied, as long as \( Q \) is stable.

Virtual queue \( P \) is just a counter in memory, which indicates the energy budget for the
mobile device to prefetch contents. The evolution of $P$ over time is:

$$P(t+1) \triangleq [P(t) - \rho(t) + \epsilon(t)]^+.$$  \hfill (6.11)

At $t = 0$, we set $P(t) = 0$. In Section 6.5.3, we show that $P(t)$ is deterministically upper bounded. This therefore satisfies the constraint in Eq. (6.8c).

We define vector $\phi(t) = [Q(t); P(t)]$ and a quadratic Lyapunov function:

$$L(\phi(t)) \triangleq \frac{1}{2} \left[ Q(t)^2 + (P(t) - \kappa)^2 \right],$$  \hfill (6.12)

where $\kappa$ is a constant. Intuitively, if the system is stabilized, i.e., $L(\phi(t))$ is maintained small, $P(t)$ is accumulated close to $\kappa$. Therefore, by choosing a reasonable $\kappa$, we control the available energy amount for the mobile device to prefetch contents. A one-step Lyapunov drift is defined as below:

$$\Delta(\phi(t)) \triangleq \mathbb{E}\{L(\phi(t+1)) - L(\phi(t))|\phi(t)\}.$$  \hfill (6.13)

The intuition is if $\Delta(\phi(t))$ is maintained negatively over time, the system is stable.

**Lemma 6.1** (Lyapunov drift). The one-step conditional Lyapunov drift satisfies the following constraint at any time slot under any algorithms used to control the system:

$$\Delta(\phi(t)) \leq \beta - \mathbb{E}\{Q(t)\mu(t) + (P(t) - \kappa)\rho(t)|\phi(t)\}$$

$$- \mathbb{E}\{Q(t)\chi(t)|\phi(t)\}$$

$$+ \mathbb{E}\{Q(t)\lambda(t)|\phi(t)\}$$

$$+ \mathbb{E}\{(P(t) - \kappa)\varepsilon(t)|\phi(t)\},$$  \hfill (6.14)

where $\beta$ is a carefully chosen constant.
Our goals are to minimize the drift and maximize the system’s utility. Thereby, we obtain:

\[
\min : \Delta(\phi(t)) - V \mathbb{E}\{U_r(t) - \alpha U_d(t)|\phi(t)\}, \tag{6.15}
\]

where \( V \) is a weight that controls how important the utility is, compared to the system stability.

By applying Lemma 6.1 to Eq. (6.15), we maximize the sum \( \Phi(t) = \Phi_1(t) + \Phi_2(t) \), where:

\[
\Phi_1(t) = \mathbb{E}\{V U_r(t) + Q(t)\mu(t) + (P(t) - \kappa)\rho(t)|\phi(t)\} \\
+ \mathbb{E}\{Q(t)\chi(t) - V\alpha U_d(t)|\phi(t)\}; \tag{6.16}
\]

\[
\Phi_2(t) = \mathbb{E}\{(\kappa - P(t))\varepsilon(t)|\phi(t)\}.
\]

**The proposed algorithm.** Our proposed algorithm is to minimize Eq. (6.15) by maximizing \( \Phi_1(t) \) and \( \Phi_2(t) \). The algorithm consists of two procedures: (1) Content Selection to maximize \( \Phi_1 \), and (2) Energy Control to maximize \( \Phi_2 \). Algorithm 7 presents the algorithm in details.

**Algorithm 7: Social Content Scheduling Algorithm**

1. **Content Selection.** In time slot \( t \), from a set of social contents \( f(t) \) fetched to the proxy so far, the proxy selects a set \( r(t) \) to deliver to the mobile user and a set \( d(t) \) to drop such that the following objective is satisfied:

\[
\max : \sum_{c \in r(t): r(t) \in f(t)} \Phi^r_1(c, t) + \sum_{c \in d(t), d(t) \in f(t)} \Phi^d_1(c, t) \\
\text{st:} \sum_{c \in r(t): r(t) \in f(t)} s_c \leq \theta(t); \\
\sum_{c \in r(t): r(t) \in f(t)} \rho_c(t) \leq \min(\epsilon(t) - e_t, P(t)), \tag{6.17}
\]

\[
\Phi^r_1(c, t) = VU_c(t) + Q(t)s_c + (P(t) - \kappa)\rho_c(t); \\
\Phi^d_1(c, t) = Q(t)\chi(t) - V\alpha U_c(t). \tag{6.18}
\]

2. **Energy Control.** In time slot \( t \), the proxy adds \( \varepsilon(t) \) to \( P(t) \) if \( P(t) \leq \kappa \).
The Content Selection procedure selects \( r(t) \) and \( d(t) \) to maximize the objective function in Eq. (6.17) with two constraints. The first constraint guarantees that the device’s downlink data rate in time slot \( t \) is enough to deliver the selected contents, and the second constraint ensures the energy is sufficient for the mobile device to receive the contents. This problem is in fact a knapsack problem that can be solved by optimization tools, e.g., CPLEX [13]. The Energy Control procedure keeps track of the virtual queue \( P(t) \), and only adds \( \varepsilon(t) \) to \( P(t) \) if \( P(t) \) does not overreach \( \kappa \) yet.

### 6.5.3 Performance Analysis

We present several theory bounds of the proposed algorithm\(^1\). Let us denote \( U^* \) as the maximum long term utility of the problem in Eqs. (6.8a)–(6.8e) achieved by any stationary randomized algorithms. Notice that such an algorithm requires to know about future. Our proposed algorithm achieves a long term utility \( \bar{U} \) arbitrarily close to \( U^* \) without knowing about future.

**Theorem 6.2** (Utility lower bound). *Our algorithm achieves a lower bound of the time average utility:*

\[
\bar{U} = \lim_{T \to \infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}\{U_r(\tau) - \alpha U_d(\tau)\} \geq U^* - \frac{\beta}{V},
\]

(6.19)

where \( \beta \) is a constant defined in Lemma 6.1.

The theorem shows that by choosing an arbitrarily large \( V \), the achieved utility is arbitrarily close to the maximum value.

**Theorem 6.3** (Real queue \( Q \) upper bound). *Our algorithm restricts the storage limit for* \( 1\)The detailed proofs are omitted due to the space limitations.
real queue $Q$ as:

$$Q(t) \leq V\alpha + \lambda_{\text{max}} \; \forall t. \quad (6.20)$$

The theorem indicates the real queue is bounded by $V\alpha + \lambda_{\text{max}}$ in any time slot. The larger $V$ leads to a larger restriction on $Q$. That is, there is a trade-off $O(V, 1/V)$ for the size of the real queue and the achieved long term utility. That is, queue size increases by $V$ leads to long term utility increase by $\frac{1}{V}$.

An interesting observation from this analysis is that we can control the storage size, which is not unlimited at the broker/proxy. Another interesting observation from this theorem is that we can control the time average queue delay. Queue delay is the time difference between a content is created and it is delivered to the mobile device. We define time average queue delay as follows: $\bar{D} = \frac{\bar{Q}}{\bar{r}} \leq \frac{V\alpha + \lambda_{\text{max}}}{\bar{r}}$ where $\bar{r}$ is time average data rate from the broker/proxy to the mobile device. It is clear that queue delay can be controlled by $V$.

**Theorem 6.4 (Energy upper bound).** *Our algorithm ensures that the energy consumed in any time slot is bounded by:*

$$P(t) \leq \kappa + \varepsilon_{\text{max}} \; \forall t. \quad (6.21)$$

The theorem shows that our algorithm ensures energy budget at any time is not more than a threshold. We know that energy budget $P(t)$ shared for the prefetching system may increase when the device is charged. This theorem infers that the energy budget is controlled even when the device is charged fully for a long time.
6.5.4 A Near-Optimal Algorithm

We next present an efficient algorithm for content selection that runs in $O(n \log(n))$ with an approximation factor of 2. That is, the algorithm achieves $\Phi_{apx}^r(t) \geq \frac{1}{2} \Phi_{opt}^r(t)$, where $\Phi_{opt}^r(t)$ is the optimal solution for the problem in Eqs. (6.17)–(6.18) and $\Phi_{apx}^r(t)$ is the near-optimal solution.

**Algorithm 8: A Near-Optimal Algorithm for Content Selection**

**Step 1:**
Given a set $f(t)$ of contents fetched by the proxy so far, select $r(t)$ to maximize:

$$\max: \sum_{c \in r(t); r(t) \in f(t)} (\Phi_{r_1}^r(c, t) - [\Phi_{d_1}^d(c, t)]^+)$$

subject to:

$$\sum_{c \in r(t); r(t) \in f(t)} s_c \leq \vartheta(t).$$

This is done by:

1. For each $c \in f(t)$, calculate $w_c = \frac{\Phi_{r_1}^r(c, t) - [\Phi_{d_1}^d(c, t)]^+}{s_c}$.

2. Add $c$ in $f(t)$ to $r(t)$ in decreasing order of $w_c$ if $w_c > 0$ until $\sum_{c \in r(t)} s_c \leq \vartheta(t)$ does not hold.

**Step 2:** Select $d(t)$ from the set $f(t) - r(t)$ by iterating through each content $c$ in $f(t) - r(t)$, and add $c$ to $d(t)$ if $\Phi_{d_1}^d(c, t) > 0$.

We first convert the problem in Eqs. (6.17)–(6.18) to a new problem. The energy consumption for receiving contents is linear to data size [33], i.e., $\rho(t) = \rho_a(t)s(t) + \rho_b(t)$, where $s(t)$ is the total amount of contents transferred in time slot $t$, and $\rho_a(t)$ and $\rho_b(t)$ are constants in each time slot, which are functions of the network type and condition. Therefore, the second constraint of the problem in Eqs. (6.17)–(6.18) becomes $\sum_{c \in r(t)} s_c \leq \min_{\rho_a(t)} \frac{(e(t) - e_0P(t) - \rho_b)}{\rho_a(t)}$, which leads to the relaxation of one of the two constraints. Let us define $\vartheta(t) = \min(\theta(t), \frac{(e(t) - e_0P(t) - \rho_b)}{\rho_a(t)})$. The problem in Eqs. (6.17)–(6.18) is equivalent to the
following problem:

\[
\text{max: } \sum_{c \in r(t), r(t) \in f(t)} (\Phi^r_1(c, t) - [\Phi^d_1(c, t)]^+) + \sum_{c \in f(t)} [\Phi^d_1(c, t)]^+ ;
\]

\[
\text{st: } \sum_{c \in r(t): r(t) \in f(t)} s_c \leq \bar{d}(t).
\]

Notice that in the new objective function \(\sum_{c \in f(t)} [\Phi^d_1(c, t)]^+\) is independent to \(r(t)\). This allows us to select the contents for delivering \(r(t)\) first, and then derive the contents for dropping \(d(t)\). We solve the problem in Eq. (6.23) in two steps, as presented in Algorithm 8. The following theorems analyze the performance of the near-optimal algorithm.

**Lemma 6.5.** Using Algorithm 8 for Content Selection achieves a bound on \(\Phi_1(t)\):

\[
\Phi^{apx}_1(t) \geq \frac{1}{2} \Phi^{opt}_1(t)
\]

**Theorem 6.6** (Utility bound). By using Algorithm 8 for Content Selection, the system achieves the following bound on the long term utility:

\[
\bar{U}_{apx} \geq \frac{1}{2} U^* - \frac{\beta_{apx}}{V}.
\]

We note that using Algorithm 8 achieves the same bounds on the real queue and virtual queue as presented in Theorems 6.3 and 6.4.
6.5.5 Theoretical Proofs

Proof of Lemma 6.1

The one-step conditional Lyapunov is:

\[
\Delta(\phi(t)) = \mathbb{E}\{L(\phi(t+1)) - L(\phi(t))|\phi(t)\} \\
= \frac{1}{2} \mathbb{E}\left\{Q^2(t+1) - Q^2(t)|\phi(t)\right\} \\
+ \frac{1}{2} \mathbb{E}\left\{(P(t+1) - \kappa)^2 - (P(t) - \kappa)^2|\phi(t)\right\},
\]

(6.26)

where the evolution of \(Q(t)\) and \(P(t)\) is defined as:

\[
Q(t+1) = [Q(t) - \mu(t) - \chi(t) + \lambda(t)]^+,
\]

(6.27)

and

\[
P(t+1) = [P(t) - \rho(t) + \varepsilon(t)]^+
\]

(6.28)

Note that

\[
Q^2(t+1) - Q^2(t) = (\mu(t) + \chi(t) - \lambda(t))^2 \\
- 2Q(t)(\mu(t) + \chi(t) - \lambda(t))
\]

(6.29)

\[
(P(t+1) - \kappa)^2 - (P(t) - \kappa)^2 = \\
(\rho(t) - \varepsilon(t))^2 - 2(P(t) - \kappa)(\rho(t) - \varepsilon(t))
\]

(6.30)
Let’s define a constant $\beta$ as below:

\[
\beta(t) \triangleq \frac{1}{2} \left[ (\mu(t) + \chi(t) - \lambda(t))^2 + (\rho(t) - \varepsilon(t))^2 \right] \\
\leq \frac{1}{2} \left[ \max(\mu_{\text{max}} + \chi_{\text{max}}, \lambda_{\text{max}})^2 + \max(\rho_{\text{max}}, \varepsilon_{\text{max}})^2 \right] \\
= \beta
\]  \hspace{1cm} (6.31)

The Lyapunov drift now becomes:

\[
\Delta(\phi(t)) = \mathbb{E}\{L(\phi(t + 1)) - L(\phi(t))|\phi(t)\} \\
\leq \beta - \mathbb{E}\{Q(t)(\mu(t) + \chi(t) - \lambda(t))|\phi(t)\} \\
- \mathbb{E}\{(P(t) - \kappa)(\rho(t) - \varepsilon(t))|\phi(t)\} \\
\leq \beta - \mathbb{E}\{Q(t)\mu(t) + (P(t) - \kappa)\rho(t)|\phi(t)\} \\
- \mathbb{E}\{Q(t)\chi(t)|\phi(t)\} \\
+ \mathbb{E}\{Q(t)\lambda(t)|\phi(t)\} \\
+ \mathbb{E}\{(P(t) - \kappa)\varepsilon(t)|\phi(t)\} \\
\]  \hspace{1cm} (6.32)

**Proof of Theorem 6.2**

Let’s denote $U^*$ is the maximum utility achieved by some stationary algorithm. From our system’s goal

\[
\max : \Delta(\phi(t)) - V\mathbb{E}\{U(t)|\phi(t)\} \\
\]  \hspace{1cm} (6.33)
where \( U(t) = U_r(t) - \alpha U_d(t) \). By applying the results from Lemma 6.1, we have:

\[
\mathbb{E}\{L(\phi(t+1)) - L(\phi(t))|\phi(t)\} - V \mathbb{E}\{U(t)|\phi(t)\} \\
\leq \beta - \mathbb{E}\left\{2Q(t)(\mu(t) + \chi(t) - \lambda(t))|\phi(t)\right\} \\
- \mathbb{E}\left\{2(P(t) - \kappa)(\rho(t) - \varepsilon(t))|\phi(t)\right\} - VU^* \tag{6.34}
\]

To maintain the stability, it follows that:

\[
\mathbb{E}\{\mu(t) + \chi(t) - \lambda(t)|\phi(t)\} = \epsilon_q \\
\mathbb{E}\{\varepsilon(t) - \rho(t)|\phi(t)\} = \epsilon_p
\]

where \( \epsilon_q \leq 0 \) and \( \epsilon_p \leq 0 \). We take an expectation with respect to the distribution of \( \phi(t) \) and use the law of iterated expectations to yield:

\[
\mathbb{E}\{L(\phi(t+1)) - L(\phi(t))\} - V \mathbb{E}\{U(t)\} \\
\leq \beta - \epsilon_q \mathbb{E}\{Q(t)\} - \epsilon_p \mathbb{E}\{(\kappa - P(t))\} - VU^* \tag{6.35}
\]

Sum over all slots \( t \in (0, 1, \ldots, T-1) \) and divide by \( T \):

\[
\frac{1}{T} \mathbb{E}\{L(\phi(t+1)) - L(\phi(t))\} - \frac{V}{T} \sum_{\tau=0}^{T-1} \mathbb{E}\{U(\tau)\} \\
\leq \beta - \frac{\epsilon_q}{T} \sum_{\tau=0}^{T-1} \mathbb{E}\{Q(\tau)\} - \frac{\epsilon_p}{T} \sum_{\tau=0}^{T-1} \mathbb{E}\{(\kappa - P(\tau))\} - VU^* \tag{6.36}
\]
Divide both sides in the above by $V$, and simply rearrange the terms to achieve:

$$\frac{V}{T} \sum_{\tau=0}^{T-1} \mathbb{E}\{U(\tau)\}$$

$$\geq U^* - \frac{\beta}{V} + \frac{\epsilon_q}{VT} \sum_{\tau=0}^{T-1} \mathbb{E}\{Q(\tau)\} + \frac{\epsilon_p}{VT} \sum_{\tau=0}^{T-1} \mathbb{E}\{\kappa - P(\tau)\}$$

$$+ \frac{1}{VT} \mathbb{E}\{L(\phi(T))\} - \frac{1}{VT} \mathbb{E}\{L(\phi(0))\}$$

Taking the limits as $T$ goes to $\infty$, and notice that $\phi(0) = 0$ and $\phi(t)$ is stabilized, we obtain:

$$\bar{U} = \lim_{T \to \infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}\{U(\tau)\} \geq U^* - \frac{\beta}{V}$$

(6.38)

**Proof of Theorem 6.3**

$Q(t)$ changes when new contents are downloaded from social network sites, or existing contents are delivered to mobile device or dropped.

From the viewpoint of content delivery to the mobile device, $Q(t)$ is bounded by:

$$Q(t) \leq \kappa \rho_{\text{max}} + \lambda_{\text{max}} = Q_1^*$$

(6.39)

where $\kappa$ is a constant, $\rho_{\text{max}}$ is the maximum energy consumed in a time slot, and $\lambda_{\text{max}}$ is the maximum content rate coming in a time slot as we defined earlier. The proof for (Eq.6.39) is based on the Selection procedure in algorithm 7. Note that we select contents to deliver to the mobile device if $\Phi^*_r(c, t) = VU_c(t) + Q(t)s_c + (P(t) - \kappa)\rho_c(t)$ is not negative. Also note that, the viewing probability $U_c(t)$ is not larger than 1.
From the viewpoint of content dropping, \( Q(t) \) is bounded by:

\[
Q(t) \leq V\alpha + \lambda_{\text{max}} = Q_2^*
\]  

(6.40)

Note that we drop contents if \( \Phi_1^d(c, t) = Q(t)\chi(t) - V\alpha U_c(t) \) is not negative. The proof for (Eq.6.40) is similar to the proof for Theorem 6.4.

It is easy to see that \( Q_2^* \) is the right bound of \( Q(t) \) at any time slot. If \( Q_2^* \geq Q_1^* \), our conclusion is correct. In the other case, since content dropping does not depend on available resources (e.g., network and energy conditions), content dropping decisions will be fired when \( Q(t) \) reaches \( Q_2^* \), and thus does not reach to \( Q_1^* \). We complete our proof.

**Proof of Theorem 6.4**

Let us assume that at time slot \( t \):

\[
P(t) \leq \kappa + \varepsilon_{\text{max}}
\]

(6.41)

as the theorem presents. We now show that at time slot \( t+1 \), \( P(t+1) \leq \kappa + \varepsilon_{\text{max}} \). Then the conclusion is \( P(t) \) is less than \( \kappa + \varepsilon_{\text{max}} \) at any time slot \( t \). It is clear that at the beginning \( t_0 \), \( P(0) \leq \varepsilon_{\text{max}} \). Therefore, the theorem is true at \( t_0 \).

At time slot \( t \), there are two cases \( P(t) \leq \kappa \) and \( P(t) > \kappa \).

1. Case 1: If \( P(t) \leq \kappa \), then \( P(t+1) \leq \kappa + \varepsilon_{\text{max}} \) because the energy budget added to the energy queue is not larger than \( \varepsilon_{\text{max}} \) as we define earlier.

2. Case 2: If \( P(t) > \kappa^* \), then \( P(t+1) \leq \kappa + \varepsilon_{\text{max}} \) because the energy queue will not be added by any energy budget amount in time slot \( t+1 \) as presented in our algorithm 7.
**Proof of Lemma 6.5**

The first procedure, Content Selection, of Alg. 7 requires to solve problem (6.17-6.18) that is known hard to be solved. The equivalent problem (6.23) converted from (6.17-6.18) is in fact a knapsack problem. This reveals that we can solve the original problem using a heuristic algorithm presented in Alg. 8. Now let’s understand more about how Alg. 8 is designed. Contents are selected based on a principle that a content with higher ratio of utility over size has a higher important level than the one with lower ratio. Our design approach is similar to work presented in [77]. The reader can follow the proof in [77] to understand how to prove our lemma.

**Proof of Theorem 6.6**

From Lemma 6.5 and the definition of \( \Phi_1(t) \), we have:

\[
\Phi_{\text{sub}}(t) = \mathbb{E}\{VU_{\text{sub}}^r(t) + Q(t)\mu_{\text{sub}}(t) - (\kappa - P(t))\rho_{\text{sub}}(t)|\phi(t)\} \\
+ \mathbb{E}\{Q(t)\chi_{\text{sub}}(t) - V\alpha U_{\text{sub}}^d(t)|\phi(t)\} \\
\geq \frac{1}{2}\mathbb{E}\{VU_{\text{opt}}^r(t) + Q(t)\mu_{\text{opt}}(t) + (\kappa - P(t))\rho_{\text{opt}}(t)|\phi(t)\} \\
+ \mathbb{E}\{Q(t)\chi_{\text{opt}}(t) - V\alpha U_{\text{opt}}^d(t)|\phi(t) = \Phi_{\text{opt}}(t)\}
\]

We start the proof with Lemma 6.1 for the Lyapunov drift of the suboptimal algorithm, \( \Delta_{\text{sub}}(\phi(t)) \):

\[
\Delta_{\text{sub}}(\phi(t)) \leq \beta - \mathbb{E}\{Q(t)\mu_{\text{sub}}(t) - (\kappa - P(t))\rho_{\text{sub}}(t)|\phi(t)\} \\
- \mathbb{E}\{Q(t)\chi_{\text{sub}}(t)|\phi(t)\} \\
+ \mathbb{E}\{Q(t)\lambda(t)|\phi(t)\} \\
+ \mathbb{E}\{(P(t) - \kappa)\varepsilon(t)|\phi(t)\}
\]

182
The objectives of stabilizing the drift and maximizing the viewing probability is:

\[
\Delta^{\text{sub}}(\phi(t)) - V \mathbb{E}\{U^{\text{sub}}(t) - \alpha U^{\text{sub}}(t)|\phi(t)\}
\leq \beta - \mathbb{E}\{V U^{\text{sub}}(t) + Q(t)\mu^{\text{sub}}(t) - (\kappa - P(t))\rho^{\text{sub}}(t)|\phi(t)\}
- \mathbb{E}\{Q(t)\chi^{\text{sub}}(t) - V \alpha U^{\text{sub}}(t)|\phi(t)\}
+ \mathbb{E}\{Q(t)\lambda(t)|\phi(t)\}
+ \mathbb{E}\{(P(t) - \kappa)\varepsilon(t)|\phi(t)\}
\]  

(6.44)

Insert Eq. (6.42) into Eq. (6.44), we have:

\[
\Delta^{\text{sub}}(\phi(t)) - V \mathbb{E}\{U^{\text{sub}}(t) - \alpha U^{\text{sub}}(t)|\phi(t)\}
\leq \beta - \frac{1}{2} \mathbb{E}\{V U^{\text{opt}}(t) + Q(t)\mu^{\text{opt}}(t) - (\kappa - P(t))\rho^{\text{opt}}(t)|\phi(t)\}
- \frac{1}{2} \mathbb{E}\{Q(t)\chi^{\text{opt}}(t) - V \alpha U^{\text{opt}}(t)|\phi(t)\}
+ \mathbb{E}\{Q(t)\lambda(t)|\phi(t)\}
+ \mathbb{E}\{(P(t) - \kappa)\varepsilon(t)|\phi(t)\}
\]  

(6.45)

By following steps presented to prove Theorem 6.2, the time average viewing probability achieved by Alg. 8 is:

\[
\bar{U}_{\text{sub}} \geq \frac{1}{2} U^* - \frac{\beta_{\text{sub}}}{V}
\]  

(6.46)

### 6.6 Proof of Concept Implementation and Performance Evaluations

As a proof-of-concept, we implemented an Android app based on the proposed concepts. Our initial prototype supports prefetching of social media streams from Facebook. The app can
download updates on users’ News Feeds page at Facebook including the feed contents (texts, images, and videos), comments and likes, and provides a friendly GUI for the users to view as well as interact (e.g., comments and likes) with the downloaded contents offline without network connectivity. Examples of the GUI are presented in Figure 6.5. We implemented the prefetching middleware as a service on Android, which contains two main threads, one to collect the meta-data for Facebook News Feeds stream and the other to prefetch the embedded contents. The app can also receive updates from the broker/proxy program.

To gain better understanding of the performance of the proposed system and algorithms, we conducted a trace-driven evaluation. We distributed the oFacebook app to 10 participants located in America, Asia and Europe to run for 10 days from May 15th to 24th, 2013, and collected trace data to drive simulations to test the different algorithms. The users were asked to accept the terms of services agreeing to let us store and send logs for evaluation purposes. To secure the participants’ data, we did not log any private content or person such
as user names, message content, post descriptions and photo/video URLs. We represent Facebook users by hashed user ids and represent Facebook contents and multimedia objects (i.e. photos and videos) by hashed object ids using an one-way hashcode function to prevent the trace back to the original information. The participants use our system similarly to the standard mobile Facebook app: they install the system through an Android executable file (.apk), login to Facebook with their Facebook account, and enjoy the downloaded Facebook newsfeed stream through a Facebook-similar navigation GUI. In the measurement version distributed in the user study, the ranker predicts a viewing probability of 1 for all posted content; the prefetching thread simply downloads all media contents - this allows us to measure user-content interaction behavior without bias. Some information collected in the collected log traces is listed:

- **Content**: information on social media contents such as created time, author id, number of likes and comments, multimedia file size. (There are 12,596 contents collected in 10 days for all users. The maximum number of contents for a single user is 3,919 while the minimum number is 130.)

- **user Activity**: the time and content id when users click to view contents. (A single participant clicks 693 times on average to view the downloaded contents. The maximum ratio of the number of views to the number of contents for a single user is 95% while the minimum ratio is 17%.)

- **Friend**: the ids of friends of users. (A single user has 310 friends on average. The maximum number of friends a user has is 503 while the minimum number is 75.)

- **System**: information on the current network (e.g., connected or not, WiFi or cellular networks, signal strength, and etc) and battery (e.g., charging or not, battery level, and etc).
6.6.1 Evaluations on the Content Ranker

We evaluate our ranking approach with Facebook data collected from the user study by employing 5-fold cross validation. Here, we randomly partition our data sets into 5 equal size subsets. Among the 5 subsets, 4 are used for training and the last subset is used for testing. This process is repeated 5 times, each time choosing one different set for testing. We report the ranking component’s performance with two metrics that are widely used in designing recommendation systems: (i) the Receiver Operator Characteristic (ROC) curve and (ii) the area under the ROC curve (AUC). The ROC curve compares the number of contents that are viewed by user (i.e., positives) and correctly predicted to be viewed (i.e., true) with the number of contents that are not actually viewed (i.e., negatives) but incorrectly predicted (i.e., false). The ROC graph is plotted on two axes where the Y axis depicts true positive rate, which is equal to the number of correctly predicted positives divided by the number of positives, and the X axis shows false positive rate, which is the number of wrongly predicted negatives divided by the number of negatives. Intuitively, points in the upper left in the graph indicate better performance. The second metric, AUC, is a measure for the effectiveness of diagnostic tests; it is interpreted as the expected true positive rate, averaged over all false positive rates. An AUC that is closer to 1 indicates a higher accuracy. AUC is known to be a good metric to indicate accuracy for data set with skewed distributions.

![Figure 6.6: Evaluations on the content ranker: (a) AUC for all users, (b) the ROC curve for three users.](image)

```plaintext
Figure 6.6: Evaluations on the content ranker: (a) AUC for all users, (b) the ROC curve for three users.
```
Fig. 6.6(a) shows the AUCs for 10 participants. Our ranking component achieves high performance with the average accuracy of 71.7% (ranging from 81% to 62%). A random ranking in which contents are predicted randomly to be viewed or not be viewed (i.e., flip a coin) would yield 50% AUC. In Fig. 6.6(b), we plot the ROC curves for three users 4, 5, and 7, whose AUC is the best, good and worst among the set of the participants respectively, to illustrate further the AUC results. The line of the circles in Fig. 6.6(b) represents the random ranking’s performance. The ROC curve results for the users are consistent to the AUCs reported in Fig. 6.6(a). For example, it is shown that the curve of user 4 with the highest AUC indeed dominates in Fig. 6.6(b), and is much higher than the random ranking’s line.

We further show the running time of the training process. Since the whole data set is from our 10-day experiment, the data for the training process includes 8-day contents (we use the 5-fold cross validation). We use a DELL laptop with a dual core 1.8 Ghz CPU and 4 GB RAM running on Windows 7, and employ MATLAB (its logistic regression libraries are glmfit and glmval) to train and extract the learning model. The average running time to come up with a model is only 0.006 seconds while the maximum running time measured is 0.138 seconds. These results indicate that our ranking component is very efficient to mobile devices. Note that we do not need to train the model frequently, but run it once per day with a data set of the most recent 10 days to update the model with the current behaviors of the user.

### 6.6.2 Mobile Middleware Evaluations

We have implemented a trace driven simulator in Java to drive the evaluation of the proposed mobile middleware. The simulator implements a time slotted system, and runs simulations for each of the ten users for a simulated time of 10 days using their own trace data. The
simulator reads (a) the newsfeed stream trace to generate content items that arrive into the system; (b) the network trace for the network condition at the simulated time; and (c) the user activity trace for the viewing activity at the simulated time. Moreover, the PowerTutor energy model is implemented in the simulator to evaluate energy consumption for content downloads. The contents derived from the trace data are ranked by the proposed ranking technique.

Since the middleware is intended to support multiple social media applications at the same time, the content/data load is expected to be higher than that from the trace where only one social media source is considered. To gain a better understanding of the performance of each prefetching algorithm under a much higher data load, we have implemented a synthetic stream generator to provide synthetic social media stream input to the simulator. The synthetic stream generator uses a Poisson model to generate social media contents that arrives to the system. Each social media content has a type and size that are drawn randomly from the collected trace data. We consider a parameter \( r \), “video ratio”, to control the probability of drawing a video over an image in each synthetic content. Moreover, to emulate the ranking on synthetic contents, the generator assigns a viewing probability to each item using a uniform-random distribution. To emulate actual user viewing behavior on the ranked and downloaded content, we label content as viewed, again, using a uniform random distribution. Consequently, about 50% of the content will be considered as viewed, this yields an accuracy of 75% for the emulated ranker (closely matches results from trace data).

Besides the synthetic stream generator, we also implemented a synthetic network connectivity simulator to produce synthetic network connectivity. The purpose the synthetic simulation is to evaluate the system under different network environments that are not covered by the trace data. The network connectivity simulator is implemented using a Markov model with three states: “WiFi connectivity”, “Cellular connectivity” and “no connectivity”. State transitions occur every 15 minutes following the specified transition probability. The band-
width of each state follows a Gaussian distribution, except for the “no connectivity” state whose bandwidth is always zero.

We consider the following performance metrics in our evaluation:

- **energy consumption**: which has 2 aspects: (a) prefetch energy consumption for contents that are prefetched and (b) on-demand fetch energy consumption for contents not prefetched, but requested.

- **prefetch energy per hit**: which is evaluated as the total prefetch energy over the number of prefetched contents being viewed by the user; since not all the prefetched contents are viewed, this metrics serves as an indication of both prefetch accuracy and prefetch energy efficiency.

- **on-demand fetch delay**: which is the downloading delay from on-demand fetching contents if they are not prefetched by the time they are viewed; since this metric aims to examine the latency that the user will experience for viewing unprefetched contents, we also considered the difference for fetching photos and videos in the simulator; For photo fetching, the delay is the latency for downloading the entire content. For video fetching, the delay is the latency for downloading the first 1 MegaByte considered as the playback buffer size.

- **cellular data consumption**: which is the amount of data downloaded through the 3G/4G interface.

We compared our proposed prefetch scheduling algorithm with two baseline approaches. The “Aggressive” scheme periodically reads all feeds that arrive to the system but have not been downloaded, ordered from the newest to oldest, and sequentially downloads them whenever the network is available. The “Aggressive(Rank)” algorithm takes into account the content rank derived from the ranking algorithm. It periodically reads the most recent
100 feeds that have not been downloaded, but only downloads contents whose predicted viewing probability is larger than 50% (i.e. considered more likely to be viewed). Besides the baseline prefetching approaches, we also consider the conventional scenario of social media access where no prefetching is used in order to demonstrate the prefetching benefit.

**Experimental Results**

Figure 6.7 reports the comparative performance of the O$^2$SM system with baseline schemes under purely trace based evaluations. We make several observations. All prefetching approaches are able to significantly improve a user’s viewing experience by reducing the on-demand fetch delay for content viewing (Fig. 6.7(a)). The “‘Aggressive’” approach improves the user’s viewing experience to the most extent, by downloading every content possible. However, its prefetch energy consumption is significantly high (Fig. 6.7(b)). On the other hand, by taking into account the viewing prediction from the ranking technique, the “‘Aggressive(Rank)’” and O$^2$SM algorithms provide much better energy efficiency by selectively downloading contents. Comparing the two we can see that while both algorithms provide a similar viewing experience improvement (Fig. 6.7(a)), the O$^2$SM algorithm uses only around 1/4 of the prefetch energy of the “‘Aggressive(Rank)’” algorithm (Fig. 6.7(b) and Fig.6.7(c)) by intelligently scheduling the download when there are good connectivity.

We also evaluated the O$^2$SM algorithm under different values of the $w_e$ parameter, which indicates the significance of the energy evaluation in the cost/benefit analysis. We can see that with larger $w_e$, the algorithm is more conservative on prefetching. It achieves less energy use but results in less improved viewing performance because fewer contents are downloaded. One way to take advantage of the effect of the parameter setting is to let the system adaptively adjust the parameter value base on the current battery level. For example, we can adapt $w_e$ to a lower (higher) value when the battery level is high(low)to enable more(less) aggressive prefetching and tradeoff viewing performance for energy conservation.

190
We also tested the impact of the forecast range of network conditions and user activities on the performance of the O²SM algorithm. We observe that when the forecast range is larger, the algorithm performs marginally better and achieves lower on-demand fetch delay and lower prefetch energy for a useful prefetch.

To gain a better understanding of the performance of each algorithm under high content/data loads especially when the network resources are not enough to ensure the prefetch of all contents, we evaluated them through synthetic simulation using the synthetic stream generator. The generator injects feeds into the system with Poisson mean rate of 1 feed per 5 minute, and the video ratio is 0.33, i.e. 1/3 of all the feeds generated are video feeds. We still use the trace data for network conditions.

Results in Fig. 6.8 show that under high content/data load, the “Aggressive” algorithm performs poorly; the high energy cost incurred (Fig. 6.8(b) and 6.9(b)) does not lead to a
better viewing performance (Fig. 6.8(a) and 6.9(a)). This is because the aggressive scheme wastes network and energy resources on content that will not be viewed. On the other hand, $O^2$SM exhibits the best energy efficiency of all techniques: it also has the lowest total energy consumption and “prefetch energy per hit”. We also evaluated the scenario where all contents will be viewed by the user. In this case, all content must be prefetched, and selective downloads based on rank will not help prefetch performance. The results are shown in Fig. 6.9. We observe that $O^2$SM can still improve prefetch energy performance by scheduling contents downloads when the mobile device has good network connectivity.

Since none of the users in our trace data have a 3G/4G data plan, to validate the system under cellular network connectivity we further evaluated the algorithms under synthetic network connectivity generated by the network connectivity simulator. The transition matrix for the network connectivity is randomly created, however, with the probability from any state to “cellular connectivity” larger than 50%. We still used the trace data for contents generated in the simulation. Fig. 6.10 shows the simulation results using synthetic network connectivity. We observe that by adjusting the $w_d$ parameter, $O^2$SM adjusts the cellular data plan use to improve prefetch performance, while keeping a low prefetch energy cost.
6.6.3 Broker/Proxy Framework Evaluations

To quantify the benefit of the broker/proxy solution, we conducted the following experiments: We install the broker/proxy on an external Ubuntu machine, and the app on two Nexus One phones running Android 2.3.6. We configure the apps on the two phones to run with and without the broker/proxy, respectively. Without the broker/proxy, the phone checks with Facebook servers for the latest updates once every 5 minutes; in the other case, the broker/proxy performs the checking of updates and signals the phone using Google Cloud Messaging, if any relevant updates appear. Upon being signaled the phone establishes a connection to the broker/proxy and receives the updates. We consider two access networks: WiFi and cellular (T-Mobile), and use PowerTutor [164] to measure the power consumption of our app. We run 30-hr experiments using the same Facebook account, and present the results in Table 6.2. This table shows that using the broker/proxy saves significant energy—we notice a 6.9-fold reduction with WiFi and a 9.1-fold reduction with cellular networks. The experimental results demonstrate the value of even a simplistic broker/proxy solution.

Table 6.2: Energy consumption (J) at devices for offline access to Facebook in 30 hours.

<table>
<thead>
<tr>
<th>Approach</th>
<th>WiFi</th>
<th>Cellular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Broker/Proxy</td>
<td>970.9 (6.9X)</td>
<td>16,783.4 (9.1X)</td>
</tr>
<tr>
<td>With Broker/Proxy</td>
<td>140.2</td>
<td>1,843.1</td>
</tr>
</tbody>
</table>
Trace-driven Simulations

Settings: We built a Java based simulator that is driven by the collected traces: the feed stream trace was used to model Facebook’s content arrival while the network trace to model network conditions for the simulation. Energy consumption was simulated using PowerTutor’s models [164]. Optimization solver [13] was used to solve the optimization problem in the Content Selection procedure in Algorithm 7. We run the simulations on a Windows 7 machine with an Intel Dual Core 1.8 GHz CPU with 4 GB RAM. We compare our algorithms, OPT (the optimal solution presented in Section 6.5.2) and NOPT (the near optimal solution presented in Section 6.5.4) with 3 baselines algorithms:

- **Rand**: This algorithm selects contents randomly to deliver to user.
- **Size**: Contents with smaller size are given higher priority for delivery selection.
- **Prob**: Contents with higher viewing probability are given higher priority for delivery selection.

In all algorithms, we do not download contents when the current energy level on the mobile device is lower than the energy threshold. The energy threshold in the simulation is set to the average energy level per user by default. For our algorithms, we set parameter \( \alpha = 10 \) to place a serious consideration on content dropping. \( \kappa \) is set to 20% of the maximum battery capacity. All users had been set to have the same energy budget periods in a day; more specially, a day time was divided into 3 periods: working period from 7 A.M. to 6 P.M, hangout period from 6 P.M. to 11 P.M., and home period 11 P.M. to 7 A.M. The corresponding usage energy percentage for these periods are 20%, 30% and 40%.

We consider several metrics to evaluate the algorithms.

- **Utility**: Accumulative viewing probability of all social contents received by a user.
• **Queue delay**: Average delay from the time a content is downloaded by the broker/proxy to the time it is delivered to a user.

• **Delivery ratio**: A ratio of the number of contents delivered to a user to the number of contents downloaded by the broker/proxy from Facebook for the user.

• **Hit late ratio**: The ratio of number of contents scheduled to be delivered to the user later than their actually viewing time to the total number of contents delivered to the user.

• **Effective energy**: Total energy consumed by the mobile device to download contents divided by the number of contents actually viewed by a user.

![Graphs showing performance metrics](image)

**Figure 6.11**: Performance of the algorithms by varying $V$.

**Simulation Results**: We evaluate our system by showing the performance of our proposed algorithms under varying values of parameter $V$. This helps us understand the proposed
algorithms’ behaviors. Later, we compare our algorithms with other baseline algorithms.

**Performance of the algorithms.** We start with comparing OPT and NOPTs’ performance under different $V$ values. Fig. 6.11(a) presents the utility achieved by our algorithms. We observe that the increase of $V$ leads to the increase of the utility in both algorithms. For example, at a very small value, $V = 10$, the utility is 51.52 for NOPT while at $V = 10^6$, the utility is 393.88. It is because at a larger $V$, the system becomes more hesitating to drop contents, and pushes content transfer to the users more strongly. A higher number of contents received by mobile device leads to a higher total viewing probability. This is supported by Fig. 6.11(b), which presents delivery ratio of contents delivered to the users. At $V = 10$, the delivery ratio is only 9%, but at $V = 10^6$, mobile device receives approximately 70%. This result proves the correctness of our performance analysis in Section 6.5.3.

An interesting result in Figs. 6.11(a) and 6.11(b) is the two algorithms, OPT and NOPT, achieves very close utility and delivery ratio (our experiments also indicate they have very similar results for the other metrics). For example, at $V = 100$, NOPT achieves the utility of 79.14 while OPT has a slightly higher value, 80.94. The largest gap of utility between the two algorithms is only 4.36 in any $V$. The key difference between these algorithms is at the running time. Using optimization solver CPLEX to solve the OPT problems leads to the worst case running time of 2.4 secs in our experiments. The worst case running
times for NOPT is only 78 ms. This indicates that NOPT runs much faster than OPT while achieving a very close performance. We thus suggest to use NOPT for the practical deployment purpose, and from now on, we do not report experiment results for OPT.

Although larger $V$ leads to higher utility, it also leads to larger queue delay. For example, in Fig. 6.11(c) at $V = 10$, the queue delay for NOPT is 0.5 hours while at $V = 10^6$, the delay is more than 8 hours. A reason for the high delay is we set energy threshold to be high (average energy level). But we see a trend here. With larger $V$, more contents in queue are considered for being delivered to mobile devices. This observation is inline with our analysis: queue size and queueing delay are strongly dependent on $V$. Transferring a larger number of contents causes high queue delay. Note that in the social network context, queue delay in some sense indicates the satisfaction of user: how soon they can view social content updates from their friends.

We report effective energy according to $V$ in Fig. 6.11(d). The lower the effective energy, the better the system works, because more energy is used for delivering right contents. Fig. 6.11(d) demonstrates that when $V$ increases, the effective energy decreases and then increases. For example, at $V = 10^1$, the effective energy is 18 J per content; at $V = 10^4$, it decreases to 11 J per content; it increases back to 30 J per content when $V = 10^6$. This trend can be explained as follows. With a small $V$, queue size is dominated in making content selection for delivery and dropping, and viewing probability does not have much impact on decisions. Thus, the effective energy is low at small $V$. With a medium $V$, the viewing probability plays a larger role, and a higher number of contents clicked to view by the user is delivered, which contributes to achieving low effective energy. With a large $V$, the system delivers more contents than necessary if network and energy conditions permit. Therefore, there are more contents not clicked to view by the user delivered to the user. This leads to higher effective energy.
Compared with other strategies. Fig. 6.12(a) shows the effective energy to download contents viewed by users using each algorithm under two energy thresholds. For fairness, we set parameter $\alpha = 10000$ to prevent our algorithm from dropping contents so that all algorithms download all contents (with different schedules) under the same energy and bandwidth constraints in the simulations. The proposed algorithms exhibit better energy efficiency than the baseline algorithms: 15%–30% difference is observed.

Fig. 6.12(b) presents the hit late ratio of the downloaded contents, which indicates the timeliness of content downloads that are viewed by users. We observe that our proposed algorithm achieves the best hit late ratio. Counter intuitively, the Prob algorithm performs worst in hit late ratio. This is because the algorithm fails to consider content size and resource conditions to download, although it prioritizes the contents on viewing probability. Downloading a long video may significantly delay the downloads of other smaller contents with similar viewing probability. In contrast, our algorithm jointly considers the viewing probability, content size, and resource conditions.

In summary, through real-world experiments and simulation studies, we have shown that:

- The broker-proxy approach is meaningful and yields significant performance benefits over a non-proxy based solution.

- The near-optimal solution is remarkably close in performance to the optimal solution that takes a long time to complete/converge.

- The near-optimal solution outperforms other state-of-the-art approaches.

- Carefully tuning the control parameter $V$ significantly improves the performance.
6.7 Concluding Remarks

Today, mobile access to current social media information is based on the connectivity of the end-user; our aim is to exploit the delay-tolerant nature of user access to social media streams to prefetch social media contents at a user’s mobile device when connectivity is good and cheap. We develop and evaluate algorithms and an end-to-end system to make the mobile social media experience seamless, personalized, and cost-effective and efficient. Towards this end, we propose a mobile middleware and a broker-proxy architecture that can efficiently sort the social media contents based on their importance to the user; and schedule the content downloads to the mobile device based on content relevance and current system conditions. The feature is implemented via a mobile app on Android platform and used to collect social network traces. We then conducted experiments on our testbed and extensive trace-driven simulations to evaluate our proposed approach and system.
In this chapter, we first conclude our contributions in developing middleware techniques for efficient societal scale information sharing. We then identify and present a few open areas that have not been touched or completely solved in this dissertation.

7.1 Summary of Dissertation Research

In this dissertation, we systematically study efficient societal scale information sharing and take a middleware approach that is resilient to the heterogeneity of the communication environments (i.e., networks and devices) and offers adaptive services to a variety of applications. We exploit the knowledge of geography and social relationships from users, and design a divide and conquer framework to address the challenges in practice. We land our focus on two major problems: 1) the information layer - who to share: in various information sharing applications, what are the mechanisms to accurately target information consumers and providers of given information; 2) the dissemination layer - how to share: given information source and targeted receivers (explicit or implicit), what are the data dissemination
mechanisms to meet the performance goals (efficiency, reliability and timeliness) of sharing applications. Furthermore, to be concrete, we categorize societal scale information sharing applications into two classes: 1) instant information sharing, highlighted by emergency and disaster alerting, and 2) delay-tolerant information sharing, exampled by online social media and question and answer.

For the class of instant information sharing applications, at the information layer, we studied mechanisms to efficiently track users’ dynamic information interests and notify relevant users when information is available. We conjectured that a topic-based publish/subscribe system can form the basis of an efficient architecture for this notification service. We designed a dynamic pub/sub broker middleware that constructs pub/sub brokers into a structured overlay and provide mechanisms to moderately reposition pub/sub users on brokers and reposition brokers on the overlay to adapt to users’ dynamic subscriptions. We implemented our solution and conducted extensive experiments showing that under highly dynamic subscriptions our solution can still maintain an efficient notification structure that provides 30% less delay and overhead, and a reconfiguration cost reduction of 80% as compared to other state-of-the-art pub/sub architecture.

At the dissemination layer of instant information sharing, we studied reliable and timely event dissemination in extreme situations (e.g., disasters). The challenge is geographically correlated regional failures caused by catastrophic disasters hinder the ability to reach recipients inside the corresponding failed region. We designed a reliable geo-social notification middleware forming a P2P overlay that is aware of (a) the geographies in which the message needs to be disseminated and (b) the social network characteristics of the intended recipient, in order to maximize/increase the coverage and reliability. It combines efficient location-based message delivery and opportunistic social diffusion through out-of band channels to achieve high reliability of information dissemination. We implemented the middleware and conducted extensive evaluations showing that our solution reaches up to 99.9% of desired
recipients even under massive regional failures of infrastructures.

For the class of delay-tolerant information sharing applications, at the information layer, we studied the mobile question and answer paradigm where information consumers explicitly request for information by submitting queries. Potential providers by taking advantage of their smartphones have the flexibility to move, collect information and response with the requested information anytime anyplace. Key challenge is to identify providers for queries so that they are not burdened inappropriately and requesters can obtain trustworthy and timely information. We designed a broker middleware that provides efficient worker selection for mobile Q&A applications taking advantage of both static and dynamic context and semantics from mobile users (e.g., geolocation, social network connections, expertise and interests, profiling of device sensors, and device battery level). We developed a prototype system consisting of an Android client app and a Q&A broker. We conducted extensive experiments with real world datasets. The results indicate that our solution is efficient and provides superior worker selection. Furthermore, our middleware is general, scalable and flexible to be leveraged by multiple mobile Q&A systems.

At the dissemination layer of delay-tolerant information sharing, we studied the problem of efficient dissemination of social media contents to mobile users to enable them to access to social media anytime anyplace without requiring to be online all the time. We designed middleware solutions that help mobile devices selectively prefetch or downloads contents that have high likelihood of being viewed, using knowledge of users social network and current network/system conditions. We developed an Android app providing offline access to Facebook. We used data traces gathered from our app to drive extensive evaluations which show that our proposed solution exhibits superior viewing performance and is energy efficiency.
7.2 Future Research Directions

We identify a few open research directions listed below. The directions either extend our research in this dissertation or study new problems for societal scale information sharing.

- A study of the privacy/security aspect of applying users’ geographical and social network information is a topic of further work. Both geographical and social network information is considered private. If acquired by malicious sources, they may be taken advantage of to harm the benefits of benign users. Since our middleware may be accessed by many applications, how to ensure the security of the information against malicious applications and users is an important research topic.

- In the dissertation we propose broker/proxy network based architecture for scalability. An interesting approach is the utilization of the upcoming cloud computing infrastructure (public, private, and edge clouds) to implement brokers/proxies. Resource intensive techniques can execute in public clouds (data centers) or in regional proxies. A structured approach to deploy our middleware to cloud infrastructure with meaningful practical solutions are key topics to be addressed in our future research.

- The research on each concerned problem can be extended as well: 1) An extension to our topic based publish/subscribe middleware is to take into account hierarchical topics for meaningful topic clustering. Also, the current middleware does not address reliability in notification. This is also a key topic to be addressed in future work; 2) Our middleware for instant information dissemination is primarily designed for wired networking environments. A nature extension is to analyze and study its application in mobile and wireless environments. Moreover, providing reliability under moving failure scenarios, e.g. tornados, is a topic of further work; 3) Our middleware for mobile Q&A addresses the challenge of the worker selection problem. A nature extension is to incorporate the publish/subscribe middleware for efficient information notification in
Q&A applications; 4) Recent research on online social media and social networks has proposed distributed social networks or federated social networks as an alternative to the traditional centralized online social network sites. While some of the systems have commercialized and start gaining popularity, an extension of our research is to take into account the distributed nature of the information source for efficient content dissemination to mobile devices.
Bibliography


