UNIVERSITY OF CALIFORNIA
Santa Barbara

# Trustworthy Decentralized Publication, Search and Retrieval in Heterogeneous Networks

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

by

Isaí Michel Lombera

Committee in Charge:

Professor Louise E. Moser, Chair

Professor P. M. Melliar-Smith

Professor Tim Cheng

Professor Forrest Brewer

June 2013

The dissertation of
Isaí Michel Lombera is approved:

---

Professor P. M. Melliar-Smith

---

Professor Tim Cheng

---

Professor Forrest Brewer

---

Professor Louise E. Moser, Committee Chairperson

June 2013

Trustworthy Decentralized

Publication, Search and Retrieval

in Heterogeneous Networks



by

Isaí Michel Lombera

To Mario Lombera Torres,

who worked much harder,

than I ever did.

# Acknowledgements

First, I want to thank the National Science Foundation for sponsoring this research and, consequently, the majority of my doctoral education. Likewise, I want to thank the Department of Electrical and Computer Engineering at the University of California, Santa Barbara, for offering me a teaching assistant position which I used to pay for the remainder of my doctoral education.

On a more personal basis, I would like to thank Professors Louise Moser and Michael Melliar-Smith, who provided valuable insight and guidance in both my research and other matters. Thanks also to Professors Tim Cheng and Forrest Brewer, for serving on my Ph.D. Dissertation Committee. I would also like to thank my fellow Ph.D. student, Yung-Ting Chuang, for working on the more unseen parts of the research with me; written publications often hide the challenging and time-consuming parts of the *real* work. Also thanks to the other students, Andre, Chris, Jerry, Wei and Balu, who were involved in our research.

Thanks also to anyone related to me now or in the future; your tolerance of me will not go unnoticed. I know enough not to be specific; otherwise, someone might be left out. However, I will mention my Mom – thank you for everything.

Finally, thanks to God and country. I've had problems in my life, though others have had *much* worse, and I like to think that providence (or fate for those who believe in coincidence) has kept me around for some purpose. As for country, I was not born, raised, or educated, nor have I lived, in the land of my ancestors;

yet my culture is so in-grained in me not to be ignored, even though I have been isolated from it. In so much that God and a country can be thanked, thank you very much!

# CURRICULUM VITAE
# Isaí Michel Lombera

307 Retreat Court
Fallbrook, CA 92028

760-468-2828
imichel@ece.ucsb.edu

## FORMAL EDUCATION

**University of California, Santa Barbara**
Doctor of Philosophy, Electrical and Computer Engineering — June 2013
Dissertation Title: Trustworthy Decentralized Publication, Search and Retrieval in Heterogeneous Networks

**San Diego State University**
Bachelor of Science, Computer Engineering — 2007
Master of Science, Electrical Engineering — 2009
Thesis Title: The Just-in-Time Discrete Wavelet Transform, a Modified Lifting Algorithm for the First and Third Dimension

## PROFESSIONAL RESEARCH AND EXPERIENCE

**University of California, Santa Barbara, ECE Dept.** — 2010 - 2012
Graduate Student Researcher: Peer-to-peer networks, mobile wireless networks, social networks, trustworthy computing.

**ingobrero!** — 2009 - 2012
Programmer: Custom hardware/software embedded design, new complete custom designs, update existing systems, board layout.

**University of California, Santa Barbara, ECE Dept.** — 2009 - 2010
Teaching Assistant: Computer architecture, machine organization, circuits/systems/devices.

**San Diego State University, ECE Dept.** — 2009
Instructor: Digital circuits lab and microprocessor lab.
Created all projects and lesson plans for students.

**Space Naval Warfare Systems Command, US Navy** — 2008 - 2009
Graduate Research Assistant: Intelligent Systems, classified.

**San Diego State University, Research Foundation** — 2007
Research Assistant: High-speed multi-gigabit transceiver with differential AMPS with Xilinx VirtexIIpro FPGA fabric.

**TAGnet Communications (later Hart Research)** 2003 - 2007
Software Engineer: Worked for the start-up company, created initial multimedia framework for multi-national community based Web site (1000+ installations).

**Controltec Incorporated** Summer 2003
Software Programmer: GhostScript API (PostScript generator), reduced document processing (15min → 1min).

**Scientific Applications International Corporation** Summer 2002
Intern: Wrote software to process GeoTIFF images taken from submersible laser scanner (using Univ. of Minnesota MapServer).

# PROFESSIONAL ACTIVITIES

- Technical Program Committee for 4th International Conference on the Evolving Internet 2012.
- Reviewer for 4th International Conference on the Evolving Internet 2012.

# PROFESSIONAL MEMBERSHIPS

- IEEE

# TEACHING EXPERIENCE

- **Teaching Assistant**:

  Computer Architecture (ECE154)

  Computer Organization (ECE15B)

  Circuits/Devices/Systems (ECE 2A).

- **Lab Instructor/Manager**: Digital Circuits Lab (COMPE470L).

# PUBLICATIONS

**Journals**

1. **Isaí Michel Lombera**, L. E. Moser, P. M. Melliar-Smith, Y. T. Chuang, Trustworthy Mobile Ad-Hoc Wireless Networks over Wi-Fi Direct, in preparation.
2. Y. T. Chuang, P. M. Melliar-Smith, L. E. Moser, **Isaí Michel Lombera**. Detection and Defensive Adaptation Algorithms for Protecting against Malicious Attacks in the iTrust Information Retrieval Network, in preparation.

3. Y. T. Chuang, P. M. Melliar-Smith, L. E. Moser, **Isaí Michel Lombera**, Statistical Inference and Dynamic Adaptation for the iTrust Search and Retrieval System, submitted.

4. Y. T. Chuang, P. M. Melliar-Smith, L. E. Moser, **Isaí Michel Lombera**, Membership Management for the iTrust Information Retrieval Network, submitted.

5. **Isaí Michel Lombera**, L. E. Moser, P. M. Melliar-Smith, Y. T. Chuang. Mobile Decentralized Search and Retrieval using SMS and HTTP. ACM Mobile Networks and Applications Journal, vol. 18, no. 1, February 2013, pp. 22–41.

6. Y. T. Chuang, **Isaí Michel Lombera**, P. M. Melliar-Smith, L. E. Moser. Protecting the iTrust Information Retrieval Network against Malicious Attacks. Journal of Computing Science and Engineering, vol. 6, no. 3, September 2012, pp. 179–192.

**Conferences**

1. **Isaí Michel Lombera**, L. E. Moser, P. M. Melliar-Smith, Y. T. Chuang. Mobile Ad-Hoc Search and Retrieval in the iTrust over Wi-Fi Direct Network. Proceedings of the Ninth International Conference on Wireless and Mobile Communications, Nice, France, July 2013.

2. W. Dai, L. E. Moser, P. M. Melliar-Smith, **Isaí Michel Lombera**, Y. T. Chuang. The iTrust Local Reputation System for Mobile Ad-Hoc Networks. Proceedings of the 2013 International Conference on Wireless Networks, Las Vegas, NV, July 2013.

3. **Isaí Michel Lombera**, L. E. Moser, P. M. Melliar-Smith, Y. T. Chuang. Peer Management for iTrust over Wi-Fi Direct. Proceedings of the 2013 International Symposium on Wireless Personal Multimedia Communications, Atlantic City, NJ, June 2013.

4. B. Peng, L. E. Moser, P. M. Melliar-Smith, Y. T. Chuang, **Isaí Michel Lombera**. A Distributed Ranking Algorithm for the iTrust Information Search and Retrieval System. Proceedings of the 9th International Conference on Web Information Systems and Technologies, Aachen, Germany, May 2013, pp. 199–208.

5. Y. T. Chuang, P. M. Melliar-Smith, L. E. Moser, **Isaí Michel Lombera**. Discovering Joining Nodes and Detecting Leaving Nodes in the iTrust Membership Protocol. Proceedings of the International Multi-Conference on

Engineers and Computer Scientists, Hong Kong, China, March 2013, pp. 89–194.

6. **Isaí Michel Lombera**, L. E. Moser, P. M. Melliar-Smith, Y. T. Chuang. Decentralized Search and Retrieval for Mobile Networks using SMS. Proceedings of the IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications, Barcelona, Spain, October 2012, pp. 134–141.

7. **Isaí Michel Lombera**, L. E. Moser, P. M. Melliar-Smith, Y. T. Chuang. A Mobile Peer-to-Peer Search and Retrieval Service for Social Networks. Proceedings of the IEEE 1st International Conference on Mobile Services, Honolulu, HI, June 2012, pp. 72–79.

8. C. M. Badger, L. E. Moser, P. M. Melliar-Smith, **Isaí Michel Lombera**, Y. T. Chuang. Declustering the iTrust Search and Retrieval Network to Increase Trustworthiness. Proceedings of the 8th International Conference on Web Information Systems and Technologies, Porto, Portugal, April 2012, pp. 312–322.

9. Y. T. Chuang, **Isaí Michel Lombera**, P. M. Melliar-Smith, L. E. Moser. Detecting and Defending against Malicious Attacks in the iTrust Information Retrieval Network. Proceedings of the 26th International Conference on Information Networking, Bali, Indonesia, February 2012, pp. 263–268.

10. P. M. Melliar-Smith, L. E. Moser, **Isaí Michel Lombera**, Y. T. Chuang. iTrust: Trustworthy Information Publication, Search and Retrieval. Proceedings of the 13th International Conference on Distributed Computing and Networking, Hong Kong, China, January 2012, Lecture Notes in Computer Science, vol. 7129, Springer, pp. 351–366.

11. **Isaí Michel Lombera**, Y. T. Chuang, L. E. Moser, P. M. Melliar-Smith. Decentralized Mobile Search and Retrieval Using SMS and HTTP to Support Social Change. Proceedings of the 3rd International Conference on Mobile Computing, Applications, and Services, Los Angeles, CA, October 2011, pp. 152–171.

12. Y. T. Chuang, **Isaí Michel Lombera**, L. E. Moser, P. M. Melliar-Smith. Trustworthy Distributed Search over the Internet. Proceedings of the 2011 International Conference on Internet Computing, Las Vegas, NV, July 2011, pp. 169–175.

13. **Isaí Michel Lombera**, Y. T. Chuang, P. M. Melliar-Smith, L. E. Moser. Trustworthy Distribution and Retrieval of Information over HTTP and the

Internet. Proceedings of the Third International Conference on the Evolving Internet, Luxembourg City, Luxembourg, June 2011, pp. 7–13.

14. M. Armella, **Isaí Michel Lombera**, S. H. Rubin, S. C. Chen, G. K. Lee. Knowledge Acquisition from Corresponding Domain Knowledge Transformations. Proceedings of the International Conference on Information Reuse and Integration, Las Vegas, NV, August 2009, pp. 175–181.

15. S. H. Rubin, **Isaí Michel Lombera**, M. Armella, J. Conn, S. C. Chen, G. K. Lee. Directing Web Search Engines using a Knowledge Amplification by Structured Expert Randomization Architecture. Proceedings of the International Conference on Sensor Networks and Applications, San Francisco, CA, November 2009, pp. 97–102.

16. **Isaí Michel Lombera**, J. Patel, S. H. Rubin, S. C. Chen, G. K. Lee. A Graphics-User Interface in Support of a Cognitive Inference Architecture. Proceedings of the International Conference on Computer Applications in Industry and Engineering 2008, Honolulu, HI, November 2008, pp. 274–279.

## POSTERS

1. **Isaí Michel Lombera**, Y. T. Chuang, P. M. Melliar-Smith, L. E. Moser. Trustworthy information distribution and retrieval. Engineering Insights 2011, University of California, Santa Barbara, CA, April 2011.

## PRESENTATIONS

1. Decentralized Search and Retrieval for Mobile Networks using SMS. 8th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, Barcelona, Spain, October 2012.

2. A Mobile Peer-to-Peer Search and Retrieval Service for Social Networks. 8th IEEE World Congress on Services, Honolulu, HI, June 2012.

3. Decentralized Mobile Search and Retrieval Using SMS and HTTP to Support Social Change. 3rd International Conference on Mobile Computing, Applications and Services, Los Angeles, CA, October 2011.

4. Trustworthy Distribution and Retrieval of Information over HTTP and the Internet. 3rd International Conference on the Evolving Internet, Luxembourg City, Luxembourg, June 2011.

5. Diversity/Opportunities in Science, Technology, Engineering and Mathematics for Minorities and Underrepresented Students: Panel Member. Escondido Union School District Leadership Retreat, Escondido, CA, August 2009.

6. The Just-in-Time Discrete Wavelet Transform, a Modified Lifting Algorithm for the First and Third Dimension. Master's Thesis Defense. Electrical and Computer Engineering Department, San Diego State University, San Diego, CA, June 2009.

7. Microbead Image/Video Processing. Research/Technology Presentation. Electrical and Computer Engineering Department, San Diego State University, San Diego, CA, December 2008.

## SELECTED COURSES

Advanced x86 System Architecture, Digital Image Processing, Multimedia/Image Processing, Mobile Systems (Cellular), Digital Circuits/VLSI, ASIC Design, Pattern Recognition, Imaging Systems (XRAY, NMRI, GPR), Digital Signal Processing, Systems Programming, Operating Systems, Embedded Systems.

## TECHNICAL SKILLS

Extensive experience in C, Java, PHP, X/HTML, CSS, VHDL, ASM, JavaScript, Matlab, Perl and SQL. Familiarity with C++, C#, VB, Verilog, VRML, TCL, BASH, OpenGL and GLSL.

## LANGUAGES

Fluent in both Spanish and English.

# Abstract

## Trustworthy Decentralized Publication, Search and Retrieval in Heterogeneous Networks

Isaí Michel Lombera

As ubiquitous networked devices continue to play an increased role in the daily lives of most people, there is a growing desire to share ever more information among friends and acquaintances. Traditional centralized search engines, such as those of Google, Yahoo! and Bing, allow simple and efficient searching for publicly available information; however, there is no similar service or method that enables people to share personal information directly and easily in a distributed and decentralized manner. Furthermore, the dissemination of information in centralized networks can be subverted or restricted by governments or corporations if the information is deemed undesirable.

To address this need, we present a trustworthy decentralized publication, search and retrieval system, named iTrust. The iTrust network may be accessed using a variety of devices from traditional computer desktops with wired networking to mobile ad-hoc wireless devices, such as mobile phones and tablets. In this dissertation, we describe the functionality and architecture of the iTrust system including: the generation and distribution of metadata that describes information, the distribution and relaying of queries, the information retrieval

process and the probabilistic method of distributing messages. We analyze the performance of the iTrust network over heterogeneous networks, including HTTP, SMS and Wi-Fi Direct. Where relevant, we discuss the novel methods developed to address specific network challenges such as mobile ad-hoc message routing, peer-to-peer network management, message size constraints, *etc.* Finally, we show how our research contributes both to the fields of computer networking and distributed systems, and also censorship-free sharing of information.

---

Professor Louise E. Moser
Dissertation Committee Chair

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Social networks such as Twitter and Facebook, as well as search services such as those of Google, Yahoo! and Bing, have been used to help coordinate mass uprisings and revolutions in the world. Unfortunately, centralized systems, whether controlled by a government or a business, are reliant on one or a few nodes that can be easily subverted or censored. If a service provider does not cooperate with such censoring entities, access to the service might be denied entirely. In Egypt and Syria, the Facebook group meeting service was used to help organize protest meeting places and times. In both countries, the government disabled the Internet to hinder the organization of those meetings.

Our trust in the accessibility of information over the Internet and the Web (hereafter referred to as the Internet) depends on benign and unbiased administration of centralized search engines and centralized search indexes. Unfortunately, the experience of history, and even of today, shows that we cannot depend on such administrators to remain benign and unbiased in the future.

The focus of this research is iTrust, a novel distributed publication, search and retrieval system that does not rely on a centralized search engine, such as that of Google, Yahoo! or Bing; thus, it is resistant to censorship by central administrators. Our initial implementation of iTrust is based on the HyperText Transfer Protocol (HTTP), and is most appropriate for desktop or laptop computers on the Internet. However, many people today use mobile phones to organize their activities. In many countries of the world, mobile phones are the only computing platform generally available. Consequently, it is appropriate to utilize the cellular telephony network to provide iTrust on mobile phones using the Short Message Service (SMS), and further to transcend such infrastructure-based networks by providing iTrust on mobile ad-hoc networks using Wi-Fi Direct. By providing a truly peer-to-peer (P2P) method of sharing information over Wi-Fi Direct, we free the users to share information among themselves without reliance on any third party. Thus, we provide the functionality of iTrust over HTTP, SMS and Wi-Fi Direct, and let the user decide which network is most appropriate to satisfy his or her needs.

The broad goals of this research involve the dissemination of information freely and easily between devices on traditionally segregated networks. For example, a mobile phone or tablet might share information with a laptop or desktop computer without the user worrying about network protocols or physical location. A mobile phone user in Asia might share a file with a desktop computer user in

South America without any central authority facilitating or censoring information. Alternatively, a number of mobile phone users might congregate within the same physical location, share files and then disperse without any infrastructure beyond Wi-Fi Direct capable mobile phones.

Finally, it is important to ensure that such a trustworthy information distribution and retrieval system is available when it is needed, even though a user might normally use a conventional centralized search engine. The heterogeneous nature of iTrust over HTTP, SMS and Wi-Fi Direct augments but does not replace traditional centralized search on the Internet.

We highlight below the three main novel contributions of this research:

- A robust method of disseminating data to ensure censorship-free sharing of information. Our multi-network information distribution and searching strategy empowers individuals to share information with others easily across the globe, from different corners of the same country and within the same room and, in some cases, with little regard to centralized or authoritative systems. We exploit the benefits of each network technology, be it HTTP or SMS or Wi-Fi Direct, and use the strengths of the particular network medium where appropriate.

- We extend SMS beyond a simple text communication tool, into a bearer of general information. Instead of using SMS to simply post or send short text messages, we use iTrust over SMS to request information directly from

devices and to retrieve that information. We ensure that any mobile phone user can access iTrust by providing multiple user interfaces and, in some cases, we work with the user interfaces already provided on the handset.

- To date, there is nascent support on devices for the capabilities of Wi-Fi Direct and little development effort to exploit this technology. Our iTrust over Wi-Fi Direct implementation is among the first to use Wi-Fi Direct; so much so, that we had to develop novel methods to connect nodes automatically. We enable conventional Android devices to use mostly ignored hardware capabilities, along with the iTrust over Wi-Fi Direct software, to enable users to create mobile ad-hoc peer-to-peer networks effortlessly.

In the remainder of this dissertation, we present each implementation of iTrust along with the requisite descriptions of the iTrust network mechanisms. First, we outline the fundamental concepts behind iTrust including the strategy, protocol and theoretical foundations. iTrust over HTTP is fully described from the Web server foundation to the user interface, as well as a performance evaluation for several hundred emulated nodes. iTrust over SMS, from the components to the user interface, is also described along with a hybrid SMS-HTTP bridge node and an implementation on several mobile devices. iTrust over Wi-Fi Direct is presented next, with a description of mobile ad-hoc capabilities and implementation on several mobile devices including smart phones and tablets. Finally, related work along with the conclusion and future work close this dissertation.

# Chapter 2

# Design of iTrust

The iTrust system is a decentralized and distributed information publication, search and retrieval system, whose objective is to prevent censorship and filtering of information accessed over the Internet and the cellular telephony network. In iTrust, metadata describing information are randomly distributed to multiple participating nodes. Similarly, requests containing keywords are randomly distributed to multiple participating nodes. If a participating node receives a request and the keywords in the request match the metadata it holds, the participating node sends the URL/URI for the information to the requesting node. The requesting node then can retrieve the information from the source node.

In this chapter, we present the iTrust messaging and membership protocols which form the fundamental design of iTrust [17, 18, 55, 57, 59, 62, 65]. We establish lower bounds for the probabilities of a match if all of the participating nodes are operational and if a proportion of the participating nodes are non-operational or subverted. These results show that distribution of the metadata

and the requests to relatively few nodes suffices to achieve a high probability of a match, even if some of the nodes are non-operational or subverted.

## 2.1 The iTrust Strategy

The nodes that participate in an iTrust network are referred to as the *participating nodes* or the *membership*.

Some of the participating nodes, the *source nodes*, produce information, and make that information available to other participating nodes. The source nodes also produce metadata that describes their information. The source nodes distribute the metadata, along with the path of the information, to a subset of the participating nodes chosen at random. For example, a Uniform Resource Locator (URL) may be shared in the case of HTTP, whereas some form of Uniform Resource Identifier (URI), such as a telephone number or a Medium Access Control (MAC) address, may be used in the case of SMS or Wi-Fi Direct.

Other participating nodes, the *requesting nodes*, request and retrieve information. Such nodes generate requests that contain keywords, and distribute the requests to a subset of the participating nodes chosen at random. Nodes that receive a request compare the keywords in the request with the metadata they hold. If a node finds a match, which we call an *encounter*, the matching node returns the URL of the associated information to the requesting node. The requesting node then uses the URL to retrieve the information from the source node. A

*match* between the keywords in a request received by a node and the metadata held by a node might be an exact match or a partial match, or might correspond to synonyms.

Initially, we assume that the metadata, generated by the source nodes, are small, much smaller than the information itself. Thus, the metadata can be communicated to participating nodes that have no interest in the information. The information is potentially large, such as a video file, and is communicated only to the nodes that need it. Each participating node generates only a small proportion of the information available, and retrieves only a small proportion of that information. Nodes produce new information at unpredictable intervals, and new information is communicated quickly to the nodes that need it.

It is possible, indeed quite likely, that a single request might result in multiple responses with the same URL/URI for a given set of metadata. In that case, the duplicates are suppressed by iTrust at the requesting node. It is also possible that a request might result in multiple responses with different URLs/URIs. Although outside the scope of this dissertation and heterogeneous networks, we have performed iTrust related research into ranking algorithms to address such cases [76].

In iTrust, we do not aim for secret or anonymous communication of the metadata or information. Metadata and requests are "public," because nodes must be able to match the keywords in the requests against the metadata they hold.

Rather, we aim for information publication, distribution and retrieval that cannot be easily censored, filtered or subverted. In iTrust, we use existing public key/private key encryption mechanisms to protect the communication of metadata and information against inspection and censorship.

In iTrust, we aim for as high a probability of a match as feasible, given the available resources (communication, processing, storage). We recognize that iTrust is more costly, particularly in communication, than a centralized search engine; however, history indicates that people are willing to accept that extra cost if they suspect censorship of a topic that they regard as important. We aim to minimize the extra cost for communication, processing and storage, but are not restricted by that cost.

## 2.2 The iTrust Messaging Protocol

At one extreme, all of the *metadata* can be flooded to all of the nodes in the network. At the other extreme, all of the *requests* for information can be flooded to all of the nodes in the network. Neither of those strategies is sufficiently efficient to be practical.

Thus, for iTrust, we use a different messaging protocol for information publication, distribution and retrieval. The steps involved in the iTrust messaging protocol are given below and are illustrated in Figures 2.1, 2.2, 2.3 and 2.4.

1. Nodes with information (the *source nodes*) distribute their metadata randomly to a set of participating nodes in the network. Some of those nodes might forward the metadata they receive to other nodes in the network.

2. Nodes that need information (the *requesting nodes*) distribute their requests randomly to a set of participating nodes in the network. Again, some of those nodes might forward the requests they receive to other nodes in the network.

3. If a node receives both the metadata and a request, the node determines whether the metadata and the keywords in the request match.

4. If a node finds that its metadata matches the keywords in the request, the matching node provides, to the requesting node, the URL or URI where the requesting node can retrieve the information. If a node finds that its metadata does not match the keywords in the request, it does nothing.

5. The requesting node then retrieves the information from the source node using the URL or URI provided by the matching node.

For appropriately chosen parameters, it is probable that at least one node receives both the metadata and a request with corresponding keywords, *i.e.*, that the request encounters the metadata and a match occurs.

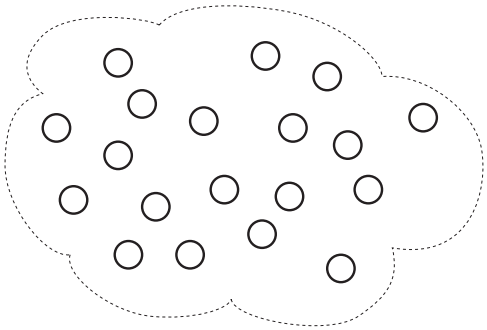


**Figure 2.1:** A network with participating nodes.

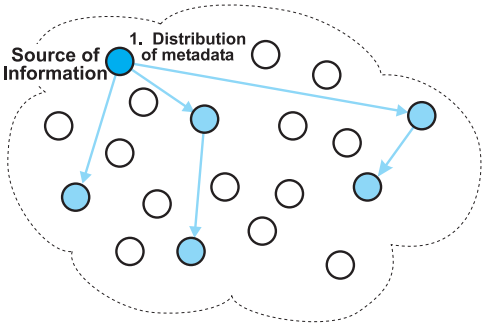



**Figure 2.2:** A source node distributes metadata, describing its information, to randomly chosen nodes in the network.

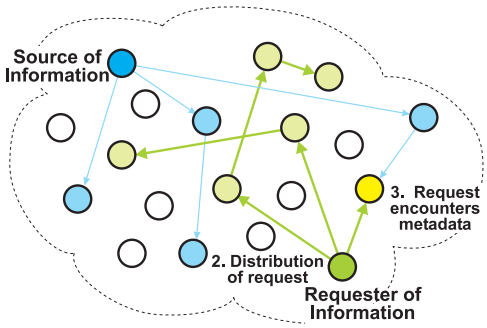



**Figure 2.3:** A requesting node distributes its request to randomly chosen nodes in the network. One of the nodes has both the metadata and the request and, thus, an encounter occurs.

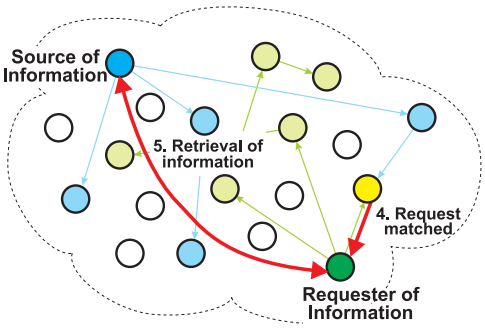



**Figure 2.4:** A node matches the metadata and the request and reports the match to the requesting node. The requesting node then retrieves the information from the source node.

## 2.3 The iTrust Membership Protocol

For iTrust to work, the nodes need to know the nodes to which the metadata and the requests are distributed, *i.e.*, the *participating nodes* or the *membership*. We use the iTrust messaging protocol itself to publish, distribute and retrieve

membership information. Each node maintains a membership table that contains, for each member, its URL or URI and its public key.

An extensive literature on membership exists (see, *e.g.*, [11, 15]), but most of that work is not relevant to iTrust. Prior work has focused on an agreed accurate membership, despite asynchrony, unreliable processors, unreliable communication, and even malice. It is impossible to achieve an agreed accurate membership [11], but good approximations are possible. Our requirements for membership present a much easier and less costly problem.

In iTrust, the nodes chosen at random for distribution of the metadata and the requests constitute only a small proportion of the participating nodes. If the membership includes nodes that are no longer participating, those nodes are equivalent to non-operational nodes. Similarly, if the membership is not yet updated to include recently joined nodes, the metadata and the requests are not distributed to those nodes. The iTrust strategy still works if a substantial proportion of the nodes are non-operational, as shown in Section 2.4.

### 2.3.1 Joining the Membership

The protocol for joining the membership exploits the iTrust messaging protocol for publication, distribution and retrieval. The steps involved in joining the membership are given below, and are illustrated in Figures 2.5 and 2.6.

1. A node wishing to join the membership contacts any current member to obtain the current membership. It does so using mechanisms that are outside the iTrust network, perhaps email, conventional Web search, twitter, Facebook or even printed publications.

2. The node then publishes its own joining the membership, using the iTrust messaging protocol for publication, distribution and retrieval.

3. The participating nodes periodically request and retrieve information about new nodes that have joined the membership.

Periodically, a participating node can compare its membership with the membership of another node chosen at random. The node can then augment its membership with the nodes known to the other node and vice versa.

Bootstrapping involves a single node or a small group of nodes that form the initial iTrust membership.

### 2.3.2 Leaving the Membership

The protocol for leaving the membership also exploits the iTrust messaging protocol for publication, distribution and retrieval. The steps involved in leaving the membership are given below and are illustrated in Figures 2.7 and 2.8.

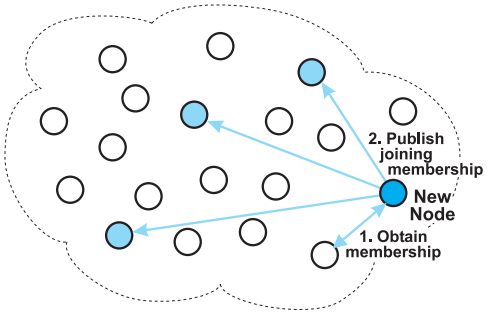



**Figure 2.5:** A node joins the membership by first obtaining the current membership from a member and then publishing its joining the membership.

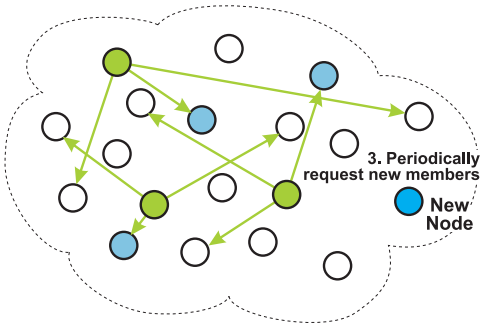



**Figure 2.6:** Other nodes periodically request information about new nodes joining the membership.

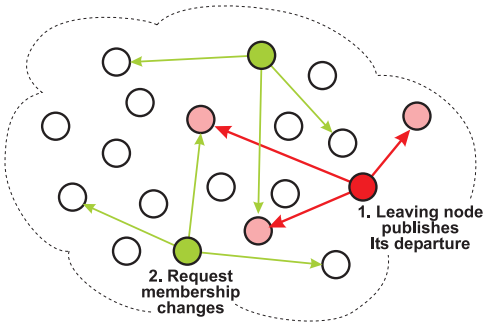



**Figure 2.7:** A node leaves the membership by first publishing its departure and then leaving. Other nodes periodically request information about membership changes.

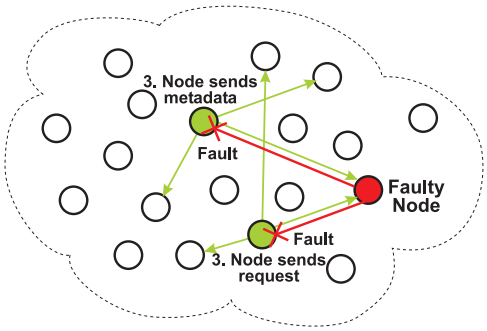



**Figure 2.8:** A faulty node does not acknowledge metadata or request messages, which alerts other nodes of its failure. Other nodes can then remove the faulty node from the membership.

1. A node that wishes to leave the membership publishes its departure and then leaves.

2. Other nodes periodically request membership change information.

3. A node might leave the membership without publishing its intention, in particular if it becomes faulty. Such an event is detected when another node

sends metadata or a request to the faulty node and does not receive an acknowledgment. The node then removes the faulty node from its membership and sends the metadata or the request to another node.

It is not appropriate to allow a node to publish the departure of another node, because doing so might enable a malicious node to remove many nodes from the membership. Rather, over time, each node individually discovers the departure of a node. When a node publishes its own departure, a digital signature (based on asymmetric encryption) is used to authenticate that publication.

## 2.4 Theoretical Foundations of iTrust

In the results presented below, we assume that all of the participating nodes have the same membership set $S$. We assume that the metadata and the requests are distributed uniformly at random to the participating nodes, without forwarding or relaying of messages. We assume that a match is an exact match between the keywords in a request and the metadata describing the information. The keywords in a request might match the metadata for two different resources with different URLs or URIs; in such a case, the matches associated with the two resources are considered separately. Initially, we assume that all of the participating nodes in the membership set $S$ are operational; later, we relax that assumption.

The primary parameters determining the performance of iTrust are the number $n$ of participating nodes (*i.e.*, the size of the membership set $S$), the number $m$

of participating nodes to which the metadata are distributed, and the number $r$ of participating nodes to which the requests are distributed.

In iTrust, all of the requests are distributed and processed concurrently; however, in the proofs below, we consider the requests as successive trials.

**Theorem 1.** If the iTrust membership set contains $n$ participating nodes, the metadata are delivered to $m$ participating nodes, a request is delivered to $r$ participating nodes, $m+r > n$, and $p$ is the probability of one or more matches, then $p = 1$.

**Proof.** Let $M$ be the subset of nodes to which the metadata are delivered, and $R$ be the subset of nodes to which the request is delivered. Because $m + r > n$, $M$ and $R$ intersect in at least one node and, thus, $p = 1$.

From Theorem 1, it follows that, if $m = r = \lceil \frac{n+1}{2} \rceil$ nodes, *i.e.*, the metadata and the requests are delivered to a majority of the nodes, then a match occurs. However, choosing $m = r = \lceil \frac{n+1}{2} \rceil$ nodes, does not scale as the number $n$ of participating nodes increases. Therefore, for larger values of $n$, we consider distributing the metadata and the requests to fewer participating nodes, specifically $\sqrt{n}$, $\sqrt{2n}$ and $2\sqrt{n}$ nodes, and investigate the probabilities of a match in these cases. Note that, for $n \geq 12$, $\lceil \sqrt{n} \rceil < \lceil \sqrt{2n} \rceil < \lceil 2\sqrt{n} \rceil \leq \lceil \frac{n+1}{2} \rceil$.

**Theorem 2.** If the iTrust membership set contains $n$ participating nodes, the metadata are delivered to $m$ participating nodes, a request is delivered to $r$ par-

ticipating nodes, $n \geq m+r$, and $p$ is the probability of one or more matches, then

$$p = 1 - \frac{n-m}{n}\frac{n-m-1}{n-1}\cdots\frac{n-r+1-m}{n-r+1}$$

**Proof.** First, we find the probability $q$ of no match on any of the $r$ trials at the $r$ nodes to which the requests are delivered. The probability of a match on the first trial is $\frac{m}{n}$ and, thus, the probability of no match on the first trial is $1-\frac{m}{n} = \frac{n-m}{n}$. Likewise, the probability of no match on the second trial is $\frac{n-1-m}{n-1}$, and so on. Finally, the probability of no match on the $r$th trial is $\frac{n-r+1-m}{n-r+1}$.

Thus, the probability $q$ of no match on any of the $r$ trials is:

$$q = \frac{n-m}{n}\frac{n-1-m}{n-1}\cdots\frac{n-r+1-m}{n-r+1}$$

Consequently, the probability $p$ of a match on one or more of the $r$ trials is $p = 1-q$, and the result follows.

**Theorem 3.** If the iTrust membership set contains $n$ participating nodes, the metadata are delivered to $m$ participating nodes, a request is delivered to $r$ participating nodes, and $p$ is the probability of one or more matches, then $p > 1-e^{-\frac{mr}{n}}$.

**Proof.** As in the proof of Theorem 2, the probability $q$ of no match on any of the $r$ trials is:

$$\begin{aligned} q &= \frac{n-m}{n}\frac{n-m-1}{n-1}\cdots\frac{n-r+1-m}{n-r+1} \\ &< \frac{n-m}{n}\frac{n-m}{n}\cdots\frac{n-m}{n} \\ &= (\frac{n-m}{n})^r = (1-\frac{m}{n})^r \end{aligned}$$

| $n$ | $\lceil\sqrt{n}\rceil$ | $p$ |
|---|---|---|
| 10 | 4 | 0.9286 |
| 100 | 10 | 0.6695 |
| 1000 | 32 | 0.6527 |
| 10000 | 100 | 0.6358 |
| 100000 | 317 | 0.6351 |
| 1000000 | 1000 | 0.6325 |
| Lower Bound | | 0.6321 |

**Table 2.1:** Probability $p$ of a match when the metadata and the requests are distributed to $\lceil\sqrt{n}\rceil$ nodes.

| $n$ | $\lceil\sqrt{2n}\rceil$ | $p$ |
|---|---|---|
| 10 | 5 | 0.9960 |
| 100 | 15 | 0.9290 |
| 1000 | 45 | 0.8800 |
| 10000 | 142 | 0.8707 |
| 100000 | 448 | 0.8668 |
| 1000000 | 1415 | 0.8653 |
| Lower Bound | | 0.8647 |

**Table 2.2:** Probability $p$ of a match when the metadata and the requests are distributed to $\lceil\sqrt{2n}\rceil$ nodes.

| $n$ | $\lceil 2\sqrt{n}\rceil$ | $p$ |
|---|---|---|
| 10 | 7 | 1.0000 |
| 100 | 20 | 0.9934 |
| 1000 | 64 | 0.9874 |
| 10000 | 200 | 0.9831 |
| 100000 | 633 | 0.9823 |
| 1000000 | 2000 | 0.9818 |
| Lower Bound | | 0.9817 |

**Table 2.3:** Probability $p$ of a match when the metadata and the requests are distributed to $\lceil 2\sqrt{n}\rceil$ nodes.

Using Maclaurin's series, $e^x = 1 + x + \frac{x^2}{2!} + \ldots$ for all $x$ and, thus, $1 + x < e^x$. Letting $x = -\frac{m}{n}$, we have $1 - \frac{m}{n} < e^{-\frac{m}{n}}$ and, thus, $(1 - \frac{m}{n})^r < e^{-\frac{mr}{n}}$. Consequently, $p = 1 - q > 1 - (1 - \frac{m}{n})^r > 1 - e^{-\frac{mr}{n}}$.

Tables 2.1, 2.2 and 2.3 show, for an iTrust membership with $n$ participating nodes, the probability $p$ of a match when the metadata and the requests are distributed to $\lceil\sqrt{n}\rceil$, $\lceil\sqrt{2n}\rceil$ and $\lceil 2\sqrt{n}\rceil$ nodes, respectively. For a given value of $n$, the number of nodes to which the metadata and the requests are delivered increases in each case, and the probability of a match increases correspondingly. These results are obtained from the formula given in Theorem 2.

Figures 2.1, 2.2 and 2.3 also show lower bounds for the probability $p$ of a match when both the metadata and the requests are distributed to $\lceil\sqrt{n}\rceil$, $\lceil\sqrt{2n}\rceil$ and

$\lceil 2\sqrt{n} \rceil$ nodes, respectively. These lower bounds are obtained from the inequality given in Theorem 3.

In the above evaluation, we have chosen specific values of $m$ and $r$, such that $m = r$, *i.e.*, the number of nodes to which the metadata are distributed is the same as the number of nodes to which the requests are distributed. However, $m$ and $r$ need not be the same.

Now, we relax the assumption that all of the nodes are operational. Thus, we assume that a proportion $x$ of the $n$ participating nodes are operational (and, thus, a proportion $1-x$ of the $n$ participating nodes are non-operational). Furthermore, we assume independence of the nodes that are non-operational.

**Theorem 4.** If the iTrust membership set contains $n$ participating nodes of which a proportion $x$ are operational, the metadata are delivered to $m$ participating nodes, a request is delivered to $r$ participating nodes, $mx + r > n$, and $p$ is the probability of one or more matches, then $p = 1$.

**Proof.** The proof is similar to that of Theorem 1.

**Theorem 5.** If the iTrust membership set contains $n$ participating nodes of which a proportion $x$ are operational, the metadata are delivered to $m$ participating nodes, a request is delivered to $r$ participating nodes, $n \geq mx + r$, and $p$ is the probability of one or more matches, then

$$p = 1 - \frac{n - mx}{n} \frac{n - 1 - mx}{n - 1} \cdots \frac{n - r + 1 - mx}{n - r + 1}$$

**Proof.** First, we find the probability $q$ of no match on any of the $r$ trials at the $r$ nodes to which the requests are delivered. Consider the first trial. The probability that the node that receives the request has the metadata is $\frac{m}{n}$, and the probability that the node has the metadata and is operational is $\frac{mx}{n}$. Thus, the probability of no match on the first trial because the node does not hold the metadata or is not operational is $1 - \frac{mx}{n} = \frac{n-mx}{n}$. Likewise, the probability of no match on the second trial because the second of the $r$ nodes does not hold the metadata or is not operational is $\frac{n-1-mx}{n-1}$, and so on. Finally, the probability of no match on the $r$th trial is $\frac{n-r+1-mx}{n-r+1}$.

Thus, the probability $q$ of no match on any of the $r$ trials is:

$$q = \frac{n-mx}{n}\frac{n-1-mx}{n-1}\cdots\frac{n-r+1-mx}{n-r+1}$$

Consequently, the probability $p$ that one or more of the $r$ nodes that receives the request has a match and is operational is $p = 1 - q$, and the result follows.

**Theorem 6.** If the iTrust membership set contains $n$ participating nodes of which a proportion $x$ are operational, the metadata are delivered to $m$ participating nodes, a request is delivered to $r$ participating nodes and $p$ is the probability of one or more matches, then $p > 1 - e^{-\frac{mrx}{n}}$.

**Proof.** The proof is similar to that of Theorem 3.

Figure 2.9 compares the probabilities of a match for an iTrust membership with $n = 1000$ nodes, obtained from the analytical formula given in Theorem 2 and

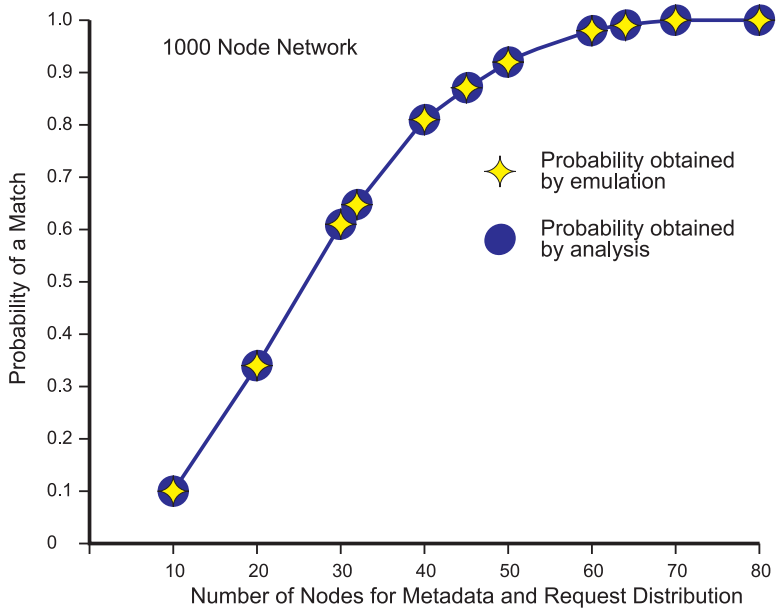



**Figure 2.9:** Probability of a match, obtained by analysis and by emulation, as the number of nodes to which the metadata and the requests are distributed increases.

from our emulation. For the emulation, each set of metadata was distributed once, and each of the search requests was performed 10,000 times and the results were averaged. The figure shows the probability of a match when the metadata and the requests are distributed to $m = r = 10, 20, 30, 40, 50, 60, 70, 80$ nodes and also to $m = r = \lceil\sqrt{1000}\rceil = 32$, $m = r = \lceil\sqrt{2000}\rceil = 45$, and $m = r = \lceil 2\sqrt{1000}\rceil = 64$ nodes. As the figure shows, the results obtained from the analytical formula and from the emulation are close.

Figure 2.10 shows the probabilities of a match for an iTrust membership with $n = 1000$ participating nodes, obtained from Theorem 5, when a proportion of the nodes are non-operational. The figure shows the probability of a match as the number of nodes to which the metadata and the requests are distributed increases

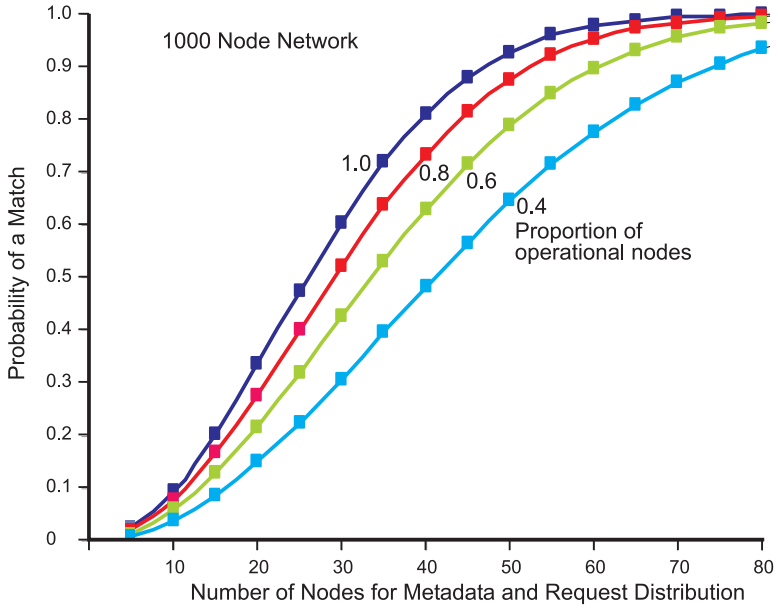



**Figure 2.10:** Probability of a match as the number of nodes to which the metadata and the requests are distributed increases, for various proportions of operational nodes.

when a proportion $x = 1.0, 0.8, 0.6, 0.4$ of the participating nodes are operational. As the figure shows, iTrust retains significant utility in circumstances in which a substantial proportion of the nodes are non-operational, which might be the circumstances in which the information is most needed.

## 2.5 Summary

In this chapter, we have presented the strategy by which iTrust distributes metadata and queries to randomly chosen nodes in the iTrust network. If the keywords in a query match the metadata held by a node, the matching node responds to the requesting node with a URL/URI, which the requesting node may

use to retrieve the desired resource. We have presented both the messaging protocol and the membership protocol that form the foundation of iTrust; the former describes how messages flow through the network and for what purpose, whereas the latter describes how individual nodes become part of the iTrust network. Finally, we have presented several theorems that govern the performance of iTrust, in particular the match probabilities.

# Chapter 3

# iTrust over HTTP

The iTrust over HTTP system [17, 18, 55, 57] allows users to share files that contain any information content and that are formatted in any way. A shared file may be a text file, an image, a video, an audio clip, or even machine code. Reading, parsing and understanding the resource is the burden of the querying node, not the network itself.

However, iTrust over HTTP does contain facilities for helping to manage files that ease users into the iTrust network. In the iTrust HTTP infrastructure, Java ARchive (JAR) files from the Apache Tika/Lucene project automatically scan shareable files and extract metadata that describes the files; thus, a user doesn't have to catalog each individual file manually.

The iTrust over HTTP infrastructure on a node consists of three distinct components that interact with each other to distribute metadata and requests and to retrieve information (resources). Figure 3.1 shows the three components: the Web server foundation, the application infrastructure and the public interface. Arrows

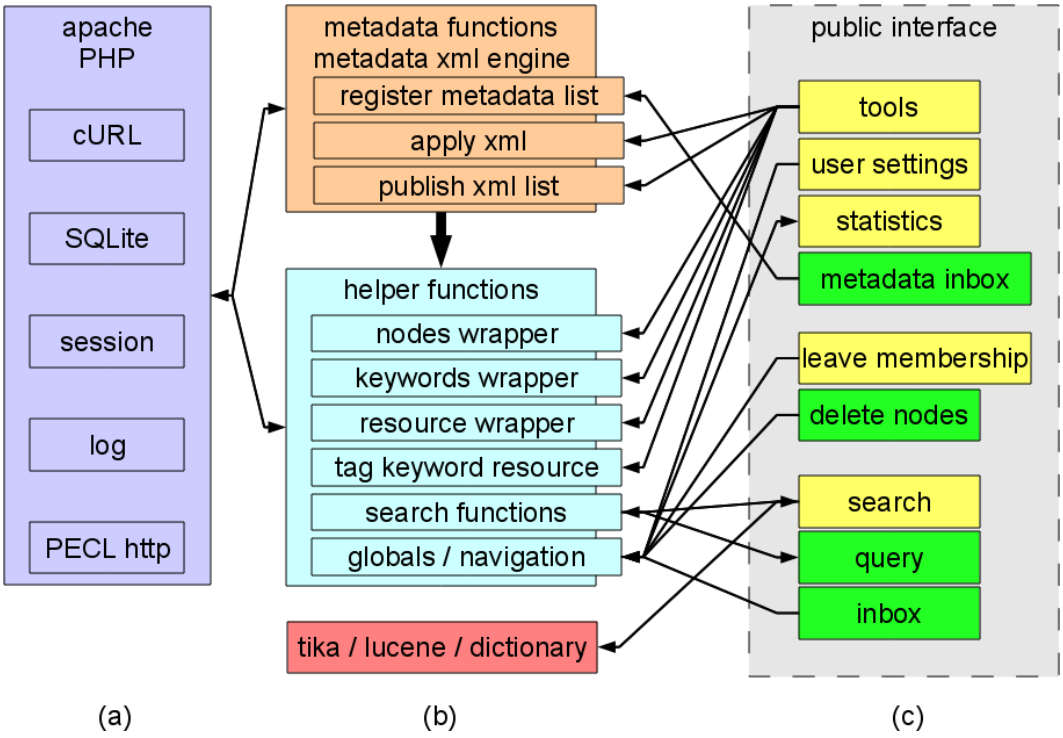



**Figure 3.1:** The iTrust system, which comprises (a) the Web server foundation, (b) the application infrastructure and (c) the public interface.

on connecting lines indicate the direction of information flow. The following paragraphs describe these three components and their interactions.

The basis of the HTTP implementation of iTrust is the Apache Web server, compiled with several PHP Hypertext Preprocessor (PHP) standard modules and library extensions, such as the session and logging modules. The session module allows tracking of users on each node, so that multiple users can interact with the same node at the same time in a convenient manner. The logging module is enabled only for debugging and simulation, and can be disabled at any time by the node administrator.

The iTrust implementation also utilizes several compiled-in modules, including cURL, SQLite and the PHP Extension Community Library (PECL) for HTTP. The cURL functions are used primarily for inter-node communication and resource-specific actions. SQLite is used for administrative information such as node, metadata and resource information. PECL HTTP is used for inter-node search and metadata queries.

The key iTrust methods reside in the application infrastructure; indeed, all of the node- and resource-related functions exist in this component. The infrastructure is divided into three parts: metadata-related functions, node- and resource-related functions and JAR files.

The creation and distribution of metadata, both internal and inter-node, are handled by the metadata-related functions. To generate metadata automatically from existing resources, the Extensible Markup Language (XML) engine scans all resources and creates an XML list that relates the metadata and the resources. Other metadata functions deal with the distribution of the XML list to other nodes, or with the receipt of XML lists from other nodes. The metadata functions scan the received XML lists, and insert the metadata into the receiving node's SQLite database.

Node- and resource-related functions, also known as helper functions, deal with bookkeeping tasks, such as functions that insert nodes into the membership, insert

keywords into the database and upload or fetch resources. The helper functions also deal with node querying and query relaying via PECL HTTP.

JAR files are used to generate metadata quickly and easily, and to provide the user with many conveniences such as spell checking, synonyms, *etc.* The Apache Tika and Lucene packages are used to generate metadata from resources automatically and efficiently, if the user chooses not to generate the metadata manually. The WordNet dictionary provides spell checking and synonym suggestions.

The public interface, through which the users and the system administrator for a node interact with iTrust, is divided between human and computer interfaces. Computer interfaces, shown as dark (green) boxes on the right in Figure 3.1, handle all inter-node communication such as queries, resource distribution and metadata list distribution. The remaining interfaces, shown as clear (yellow) boxes on the right in Figure 3.1, are human-oriented and consist of PHP-driven HyperText Markup Language (HTML) Web pages.

The iTrust HTTP implementation optionally replicates the resources themselves (not only the associated metadata) across nodes. Doing so increases the probability of a match at the expense of extra network traffic during such replication, which might be desirable if the query "hit" latency must be small and the resource file sizes are small.

Administration is performed through the tools and other Web pages. Tools allow a node administrator to add nodes or metadata keywords using simple HTML

form text boxes. Adding a resource involves uploading a file (form file input) or providing a URL (form text box input). User settings and statistics Web pages provide feedback about the membership size, resource count, *etc.* The administrator may generate and distribute metadata lists or update a node's metadata lists. The administrator may also remove a node from the membership.

Searching is performed using a Web page, where the user enters a search query to request a resource. The query is sent from the current node to participating nodes using computer interfaces in a simple inbox fashion. A participating node reads its inbox for query requests, and sends back a response if there is a match.

The next section provides further details on each component of the iTrust HTTP implementation.

## 3.1 HTTP Implementation

The HTTP implementation of iTrust enables a user to distribute metadata and requests easily. Each node is implemented using PHP on an Apache Web server, thereby allowing any user with a Web browser on any platform to interact with the node. Node information is stored locally in an SQLite database. Multiple nodes can be installed on a single Web server by creating multiple virtual Web sites; multiple nodes on a single Web server have separate SQLite databases.

### 3.1.1 Membership

The membership list for a node is stored locally in an SQLite database table that contains node records whose fields are the node identifier and the node address. The node address may be either an Internet Protocol (IP) address or a URL. A node is not verified when it is added to the membership as there is no guarantee that a node is always available. For example, a cell phone or a laptop with a Wi-Fi connection may enter and leave its access point signal range multiple times a day. In practice, the only restriction on node addresses is that the Web site document root has Web server write permissions for saving uploaded resources.

### 3.1.2 Resources

Resources are files or groups of files uploaded to nodes in the membership. The list of resources on a node are stored in an SQLite database table that contains resource records whose fields are a resource handle, file path and expiration date. The resource handle is a shortened random name (typically with 32-64 characters), which can be referenced across participating nodes. The file path is the name of the file on the local node disk. Thus, a participating node retrieving data may simply request a resource by the handle to the source node, instead of using the entire file path. The expiration date specifies when a given resource and associated keywords should be deleted. It allows time-sensitive information to be removed

automatically from queries past the expiration date when the information is no longer relevant. For example, yesterday's weather forecast is not included in today's weather queries.

Resource files can be placed on a node using the Common Gateway Interface (CGI) by means of a file dialog, or through the cURL package provided by PHP. In the case of cURL, the file(s) is fetched directly by the node and written to the local disk.

When a resource is entered into the SQLite database, metadata for the resource file is generated using the Apache Tika/Lucene packages. These packages classify metadata based on content, such as text strings, and file attributes, such as data type, file size, *etc.* Also, the user may supply additional metadata keywords. The metadata keywords are stored in a SQLite database table, and the associations between a resource and a keyword are stored in a separate table, thus normalizing the database.

### 3.1.3 Metadata Distribution

Periodically, metadata keywords and other information about one (or more) resources on a node are collated and compiled into an XML file (also referred to as the metadata list) which describes the resource. The periodicity depends on the node and the platform; however, in practice, the user can update the metadata list at any time by clicking a button or running a cron job. The resource description

in the metadata list includes the resource handle, file path and expiration date for the resource. The metadata list includes all associated keywords for the resource.

After creation of the metadata list, random nodes in the membership are contacted and informed of a metadata update by means of an HTTP POST statement. The number of random nodes that are selected for metadata distribution is a tunable parameter in the iTrust node configuration file. The contact message includes the source node IP address and metadata list URL, which are stored on the receiving node. Each contacted node decides if and when to retrieve the metadata list. The retrieval period is receiving node dependent, so as not to trigger an instant download of the metadata list file by multiple nodes.

When a node retrieves the metadata list file from the source node, it processes the XML and stores the metadata list . If there are multiple resources represented by sets of metadata in the metadata list, then processing continues on the next set of metadata, until the end of the metadata list is reached. XML processing is performed using the SimpleXML PHP extension.

### 3.1.4 Query Relaying

Search queries (requests) are the main interaction between the user and an iTrust node and, as such, require the most processing. A search query originates at a single node, but the query message may be relayed among multiple nodes in

the iTrust network. A query message may take any available network path with the sole restriction that a node never relays the same query twice.

The query field is a simple HTML form text box on the current node; the command to begin the query is detection of pressing a submit button or enter key. The query text itself is URL encoded to facilitate later operations; no custom processing on the query text (*e.g.*, duplication detection, grammar checking, *etc.*) is performed.

Two additional variables are created before a node sends the query: the node IP address and the query identifier. The node IP address is read from the iTrust configuration file; it ensures that queried nodes know which node originally sent the query. The query identifier ensures that no query is relayed twice and helps manage multiple queries sent from a single node.

Once the query is ready to be sent, multiple random nodes are selected from the node database table and the query is packaged into an HTTP POST statement. The frequency of node selection is another customizable iTrust configuration variable and need not be the same at different nodes. Node selection for querying is not necessarily the same as node selection for metadata list distribution; both variables may be set by the administrator of the node.

A node sends the HTTP POST statement to random nodes, and each node (both sender and receivers) saves a copy of the query before processing. If any text in the query matches the metadata keywords, an encounter occurs. The

queried node then sends a HTTP POST response back to the originating node. The originating node is obtained from the sender node's IP address given in the query. The response includes the query identifier (again obtained from the query), the encountered node's IP address (hereafter referred to as the source node), and the resource handle of the matching resource. Additionally, the querying node saves the response in an SQLite database table for later processing.

A queried node, regardless of whether or not it has an encounter, may relay the query as if it were the originating query node. The only difference is that it does not recreate the two additional variables (source IP address and query identifier); those variables are relayed without modification. However, before relaying the query, the node checks the query identifier to ensure that the node has not already seen the query. If the node has already saved the query identifier, it does not relay the message.

Note that the current node in this context may be either the node sending the query or the node receiving the query. In case the receiving node has not yet relayed the query, it relays the query to nodes randomly selected from the membership and records the query identifier. In case the receiving node has already relayed the query, the receiving node ignores the query. Because of the decentralized random nature of iTrust, the original node that sent the query might have the same query relayed back to itself. In this case too, the current node ignores the query.

After waiting for responses a certain amount of time, the querying node displays all of the source nodes with appropriate resource handles. All encounters are thus recorded on the querying node, and the user is informed of which nodes have resources matching the query.

### 3.1.5 Retrieving Resources

All query results are recorded in the requesting node's SQLite database table, *i.e.*, the source node IP address and resource handle. When the user selects a query result, the source node is sent a HTTP GET statement with the resource handle, and the source node returns the resource file directly to the user. Alternatively, the source address and resource handle can be encoded directly into a URL; the user then accesses the file using a HTML anchor tag.

## 3.2 User Interface

The iTrust user interface is a Web-based interface where the user can both administer and query the nodes through Web pages. Query results from multiple nodes are presented in a single Web page following a query. Node administration and user queries are separated into distinct Web pages to keep tasks distinct and easily manageable.

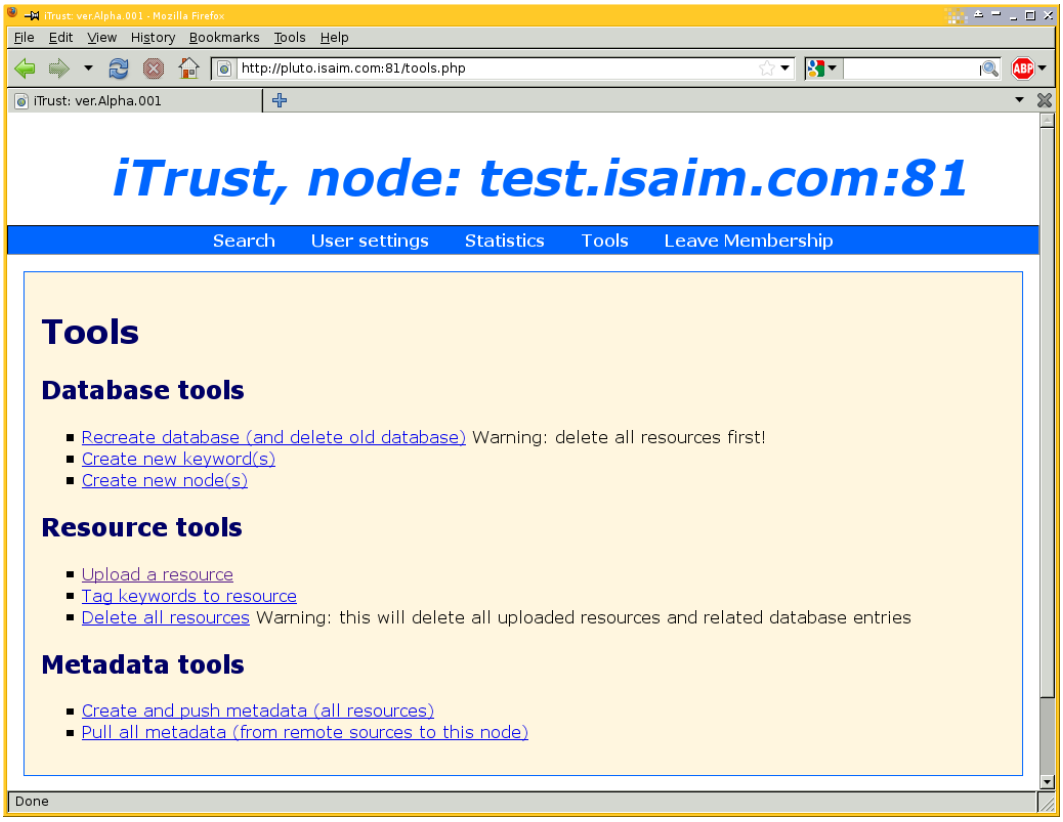



**Figure 3.2:** The administration interface.

### 3.2.1 Node Administration

The user may change the membership, add source nodes, distribute metadata and perform other administration tasks through the administration interface shown in Figure 3.2.

A node is added to the membership by entering the node IP address or URL on a comma delimited list inside a HTML form text box. Double listing is not permitted; duplicates are removed from the list. However, multiple nodes are permitted as long as the Web site document root is distinct (*e.g.*, both www.example.com/foo and www.example.com/bar are allowed).

Figure 3.3 shows the resource insertion Web page. A resource is added to a node by means of an HTML form file control; this control permits the user to upload a file from his/her local machine. Alternately, a Web site URL can be specified, and the node then fetches the contents at that URL. The uploaded contents are post-processed, using the Apache Tika/Lucene package, to generate descriptive metadata (*i.e.*, keywords) automatically. The user can customize several parameters for metadata creation, including indexing by file raw content (literal text strings) or file meta content (file size, type, *etc.*). In addition to automatic metadata creation for an uploaded resource, the user may add new keywords or remove existing keywords. Finally, the user may assign an expiration date to the resource.

Administration tasks also include file administration functions to allow the user to setup, restore, or reset iTrust nodes easily. Clearing the membership, deleting all resources and metadata associations and resetting a node to its initial setup state can all be done with a single button click. The task of pushing all metadata changes to random nodes is also accomplished with a single button click.

### 3.2.2 User Queries

The user may perform queries, view the query results and obtain resources through the user interface.

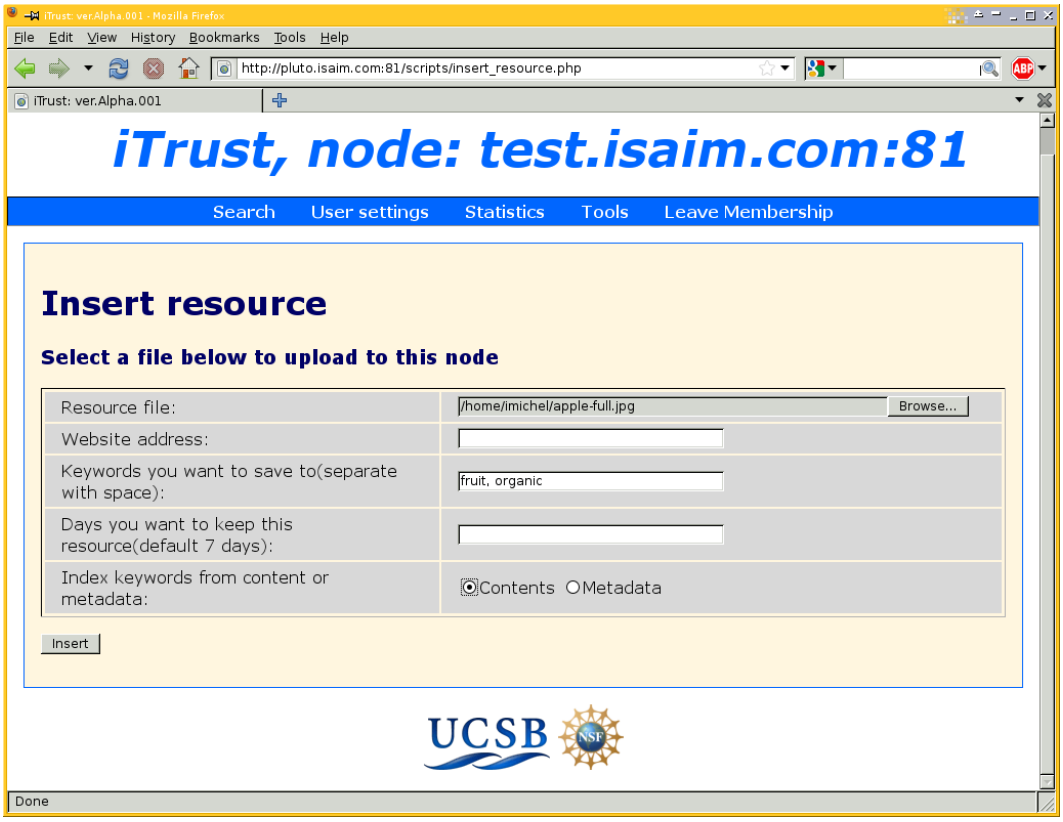



**Figure 3.3:** The insert resource Web page.

Querying is done through a single HTML form text box, whereupon the query is registered on the node and distributed throughout the iTrust network. The user is shown a status/wait Web page while the query is relayed among nodes; a result Web page is shown after a wait page timeout. The default timeout is 3 seconds and, thus, a query incurs a 3 second latency between initialization of the query and display of the query results. However, the wait page timeout is also configurable by the node administrator.

Figure 3.4 shows the query results displayed on a new Web page (the wait page automatically redirects to the new page) in a simple HTML list. Each encounter

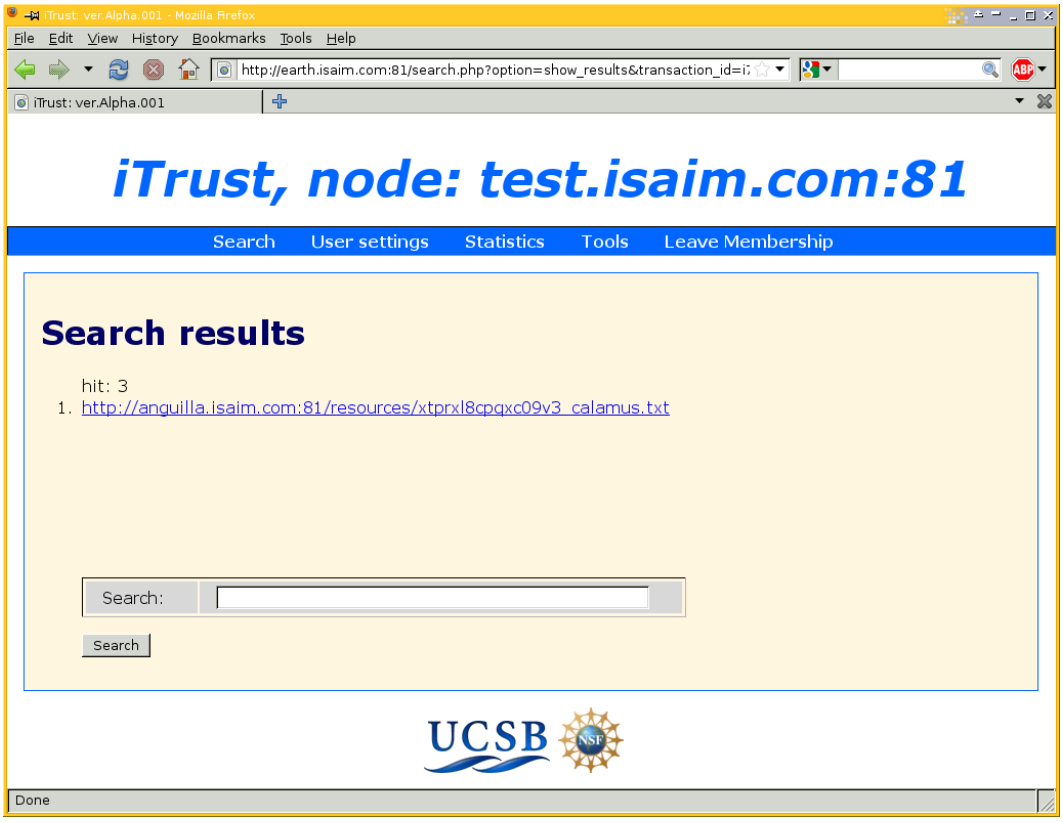



**Figure 3.4:** The query results Web page.

is shown as a list item with the source address and resource handle encoded into a single URL.

The user may click the URL to retrieve the resource file; the format of the file is the originally uploaded format (there is no MIME-type modification). If the Web browser recognizes the file type, it handles the data accordingly; otherwise, it calls the operating system to open the data file.

### 3.2.3 User Settings

For querying, the three primary user settings (which the user sets on the user settings page) are the number of nodes to which the metadata are distributed, the number of nodes to which the requests are distributed and the search duration.

The number of nodes to which the metadata are distributed and the number of nodes to which the requests are distributed must, of course, be less than the number of participating nodes in the membership.

The search duration refers to the lifetime that a search query exists. The user may specify how many days a query will be stored in the database. When a user initiates a query, the system adds its creation time to the database. Later, when the user initiates a new query, the system checks and deletes expired queries from the database.

These user settings apply to the entire duration of a search session. The search session starts when a user accesses the search Web page and ends when the user exits the browser window or tab. The PHP session functions are used to automate this process.

## 3.3 Performance Evaluation

In the performance evaluation, we consider the probability of a match, using both analysis and emulation based on our iTrust over HTTP implementation. We

assume that all of the participating nodes have the same membership set. In addition, we assume that the Internet is reliable and that all of the participating nodes have enough memory to store the source files and the metadata.

For the analysis, we use the theorems in Section 2.4 to calculate the match probabilities and represent them on a graph. For the emulation, we set up a Web server with 144 distinct named virtual hosts on a public IP address and with a registered domain name. Although all nodes were on the same IP address, the registered domain name allowed us to use realistic time delays between node communication (DNS lookups were not cached). We then plotted the emulation results against the analysis results.

### 3.3.1 Analysis

In an iTrust network with a membership of $n$ nodes, we distribute the metadata to $m$ nodes and the requests to $r$ nodes. The probability $p$ that a node has one or more matches then is:

$$p = 1 - \left(\frac{n-m}{n}\right)\left(\frac{n-m-1}{n-1}\right)\ldots\left(\frac{n-m-r+1}{n-r+1}\right) \tag{3.1}$$

Formula 3.1 holds for $n \geq m + r$. If $m + r > n$, then $p = 1$.

As above, we distribute the metadata to $m$ nodes and the requests to $r$ nodes in an iTrust network with a membership of $n$ nodes. But now we introduce another variable $x$, which represents the proportion of the $n$ nodes that are operational. Non-operational nodes might have failed, or left the network, or might be mali-

cious, refraining from matching requests against metadata. In an iTrust network with a membership of $n$ nodes, where $x$ nodes are operational, the probability p that a node has one or more matches is:

$$p = 1 - \left(\frac{n - mx}{n}\right)\left(\frac{n - mx - 1}{n - 1}\right)\ldots\left(\frac{n - mx - r + 1}{n - r + 1}\right) \tag{3.2}$$

Formula 3.2 holds for $n \geq mx + r$. If $mx + r > n$, then $p = 1$.

Figures 3.5, 3.6 and 3.7 show the probability $p$ of a match obtained from Formulas 3.1 and 3.2 with $n = 144$ nodes where $x = 100\%$, $80\%$ and $60\%$ of the participating nodes are operational, respectively, as a function of $m = r$ (the number of nodes to which the metadata and requests are distributed). As we see from the graphs, the probability $p$ of a match increases and asymptotically approaches 1, as $m = r$ increases.

### 3.3.2 Emulation

Using our iTrust over HTTP implementation described in Section 3.1, we performed experiments to validate the analytical formulas given above. In our emulation, we used libCURL (which is a free client-side URL transfer library for transferring data using various protocols) to collect the match probabilities.

Before we run our emulation program, we delete all resources and data from the SQLite databases. Next, the program adds all the nodes to the membership. Once all the nodes are added to the membership, we supply the number of

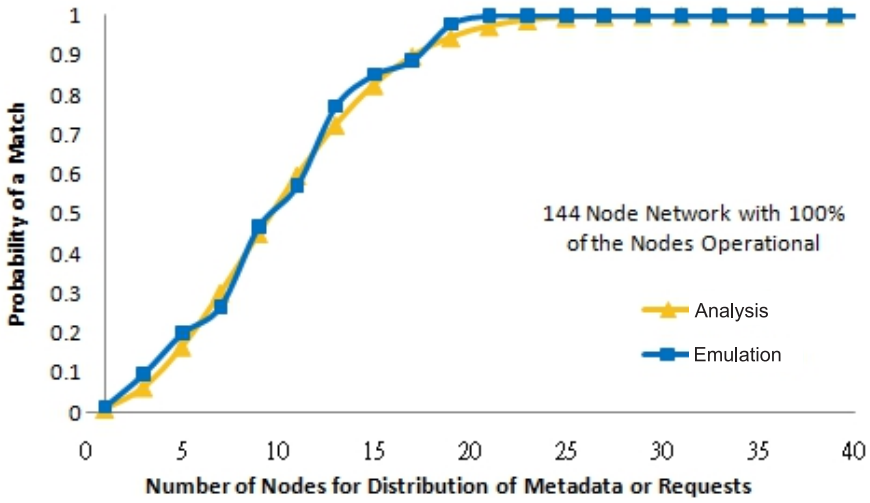



**Figure 3.5:** Match probability vs. number of nodes for distribution of metadata and requests in a network with 144 nodes where 100% of the nodes are operational.

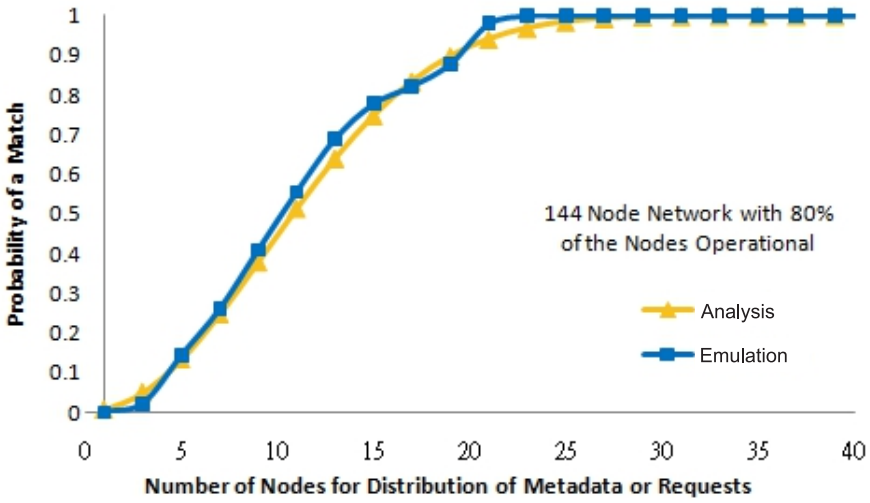



**Figure 3.6:** Match probability vs. number of nodes for distribution of metadata and requests in a network with 144 nodes where 80% of the nodes are operational.

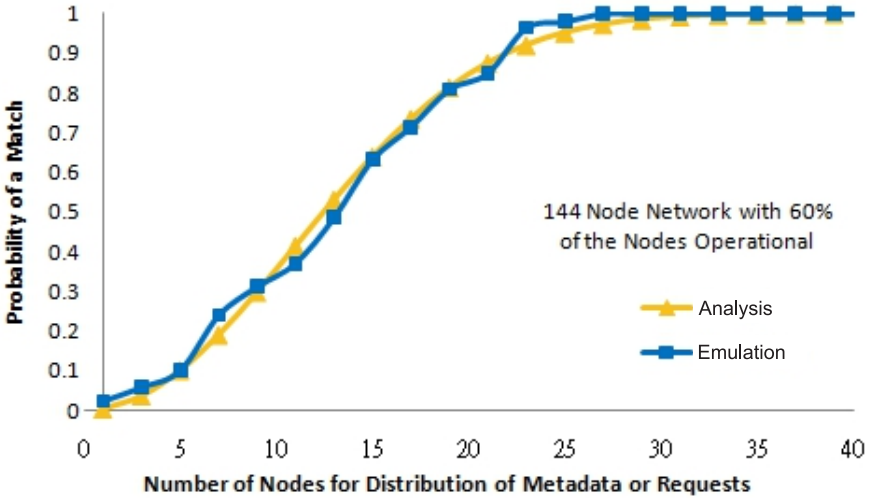



**Figure 3.7:** Match probability vs. number of nodes for distribution of metadata and requests in a network with 144 nodes where 60% of the nodes are operational.

nodes for distribution of metadata and requests, and the proportion of operational nodes, to the emulation program. Next, we call the source nodes to upload files and the program then creates the corresponding metadata. Then, the program randomly selects the nodes for metadata distribution and distributes the metadata to those nodes. Next, the program randomly selects the nodes for the requests and distributes the requests. If one or more nodes returns a response, there is a match and the emulation program returns 1; otherwise, there is no match and the emulation program returns 0.

Figures 3.5, 3.6 and 3.7 show the emulation results with 144 nodes where 100%, 80% and 60% of the participating nodes are operational, respectively. As we see from these graphs, the emulation results are very close to the analytical

results calculated from Formulas 3.1 and 3.2 where 100%, 80% and 60% of the participating nodes are operational.

The lesson we learned from this performance evaluation is that iTrust retains significant utility even in the case where a substantial proportion of the nodes are non-operational.

## 3.4 Summary

In this chapter, we have presented the iTrust over HTTP system, which is divided into three parts: the Apache/PHP Web server foundation, the metadata generation and query matching/distribution code and the Web-based user interface. The user interface is designed to be easy-to-use by the typical computer novice – queries can be easily made and resources can be easily retrieved; node administration is easily accomplished through the Web user interface; and network parameters can be effortlessly changed. We also presented a performance evaluation of iTrust over HTTP, which shows that the measured data from an emulation of iTrust over HTTP support the analytical results for the probability of a match.

# Chapter 4

# iTrust over SMS

Mobile phones have become pervasive in our daily lives; mobile applications, in addition to providing basic communication and entertainment services, have become enablers of societal transformation. Social networks such as Twitter and Facebook, as well as search services such as Google and Bing, have been used to help coordinate mass uprisings and revolutions in the world. For example, in Egypt and Syria, the Facebook mobile group meeting service was used to help organize the places and times of protest meetings. In several cases, governments disabled local access to the Internet to hinder the organization of such meetings.

Modern day users expect mobile phones to have many of the same capabilities that more traditional computers have. The modern user wants a computer that fits in his (her) pocket (purse) and that is network enabled. In many countries, mobile phones are the only computing platform generally available; thus, it is appropriate to provide the iTrust system on mobile phones.

We extend the iTrust publication, search and retrieval system based on HTTP, so that it does not only rely on the Internet but also can utilize the cellular telephony network. First, we extend the iTrust over HTTP system to allow users of mobile phones to connect to iTrust over HTTP via SMS, so that they can benefit from the decentralized publication, search and retrieval service that iTrust provides. We name this system iTrust *with* SMS [56, 57]. Our objective is not to supplant HTTP but instead to have SMS work along side it, to increase accessibility during dynamic situations where mobile phones are used. Second, we completely re-implement the iTrust over HTTP system to work only over SMS, thus creating the iTrust *over* SMS system [60, 57, 62]. Whereas iTrust with SMS allows mobile phones to send text messages to the iTrust over HTTP network, iTrust over SMS allows mobile phones to form self-contained networks that are not necessarily connected to the Internet. iTrust over SMS allows mobile phones to search for and retrieve documents entirely within the cellular telephony network; an Internet connection is not required. Figure 4.1 illustrates these different kinds of iTrust networks.

The remainder of this chapter is organized as follows. Section 4.1 briefly discusses mobile search and SMS . Next, Section 4.2 describes the implementation of iTrust with SMS, focusing on the iTrust SMS-HTTP bridge that allows any hardware-capable iTrust over HTTP node to act as a relay of queries that originate from an SMS-capable mobile phone. Section 4.3 presents the iTrust with SMS

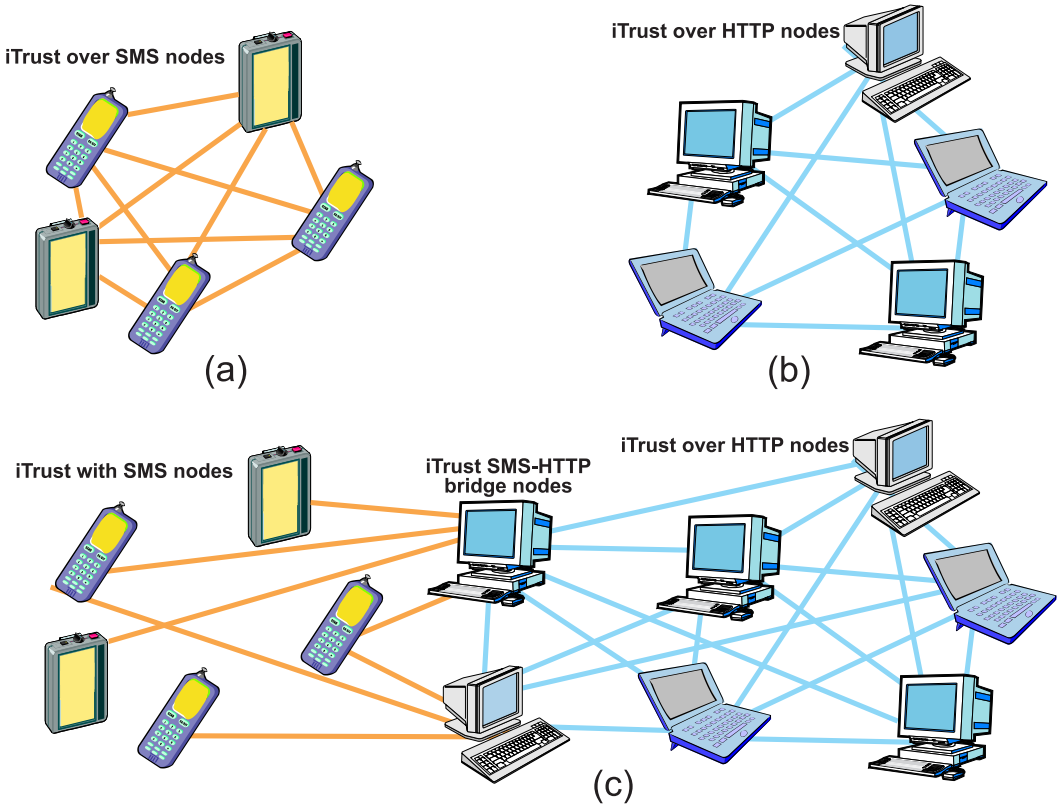



**Figure 4.1:** Different kinds of iTrust networks, showing: (a) iTrust over SMS nodes, (b) iTrust over HTTP nodes and (c) iTrust with SMS nodes communicating with iTrust SMS-HTTP bridge nodes communicating with iTrust over HTTP nodes.

user interface that allows users to make queries and receive query results. Next, Section 4.4 describes the iTrust over SMS protocol and implementation that allows mobile phones in the iTrust network to communicate directly over the cellular telephony network. Section 4.5 presents the rudimentary iTrust over SMS user interface that allows users to make queries and receive query results. In Sections 4.6 and 4.7, we extend the rudimentary iTrust over SMS user interface into a full Android application and describe several use cases. Finally, Section 4.8 presents a performance evaluation.

## 4.1 Mobile Search and SMS

The Short Message Service (SMS) works on low-end mobile phones and is available worldwide. Global SMS traffic is expected to reach 8.7 trillion messages by 2015, up from 5 trillion messages in 2010 [90]. To quote Giselle Tsirulnik, senior editor at Mobile Commerce Daily, "SMS is cheap, it is reliable, it is universal, and it has unrivaled utility as a bearer for communications, information and services." In developing countries, SMS is the most ubiquitous protocol for information exchange after human voice.

In SMS-based search, the query and the response are limited to 140 bytes each. Moreover, the user has to specify a query and obtain a response in one round of search. Significant work has been undertaken to improve mobile search using SMS text messages [82]. In iTrust, an SMS request (query) consists of a list of keywords, which are typically less than 140 bytes. An SMS response simply returns the requested information if it is small (less than 140 bytes). If the requested information or document is larger than 140 bytes, it is fragmented into multi-part SMS messages. Alternatively, the SMS response can return a URL, which is typically less than 140 bytes.

Finally, the short message size and transmission frequency of SMS messages accustoms users to utilizing the service for almost real-time momentary or fleeting communication. After an hour, or even several minutes, most SMS messages are no longer important to the user; in many cases, even important messages are

meaningless without surrounding context such as time, circumstances, or information not recorded directly by the mobile device. For this reason, the temporal integrity of an SMS query result is relevant only if a search hit is returned relatively quickly (within minutes); otherwise, without the context provided by the user, the information is meaningless.

## 4.2 Implementation of iTrust with SMS

The iTrust with SMS system enables any node (laptop, desktop, server) to act as a bridge between an SMS-capable mobile device and an iTrust over HTTP node. The only requirement for an iTrust with SMS node is having a hardware interface for receiving and transmitting SMS messages; a simple and inexpensive cellular modem suffices. Note that only a single hardware interface is required for sending and receiving SMS messages. (Not all iTrust nodes need to be SMS-capable.) Additionally, a single node may have any number of such inexpensive cellular modems connected thus, creating multiple points of communication between the SMS-enabled querying device and the iTrust with SMS node (in Section 4.2.2, we describe the open-source software utilized to service the modems). The result is that an existing iTrust network can remain unchanged; only the iTrust SMS-HTTP bridge node must be software updated.

Figure 4.2 provides a system block diagram that shows the communication path taken by SMS request and response messages. Specifically, it shows the three

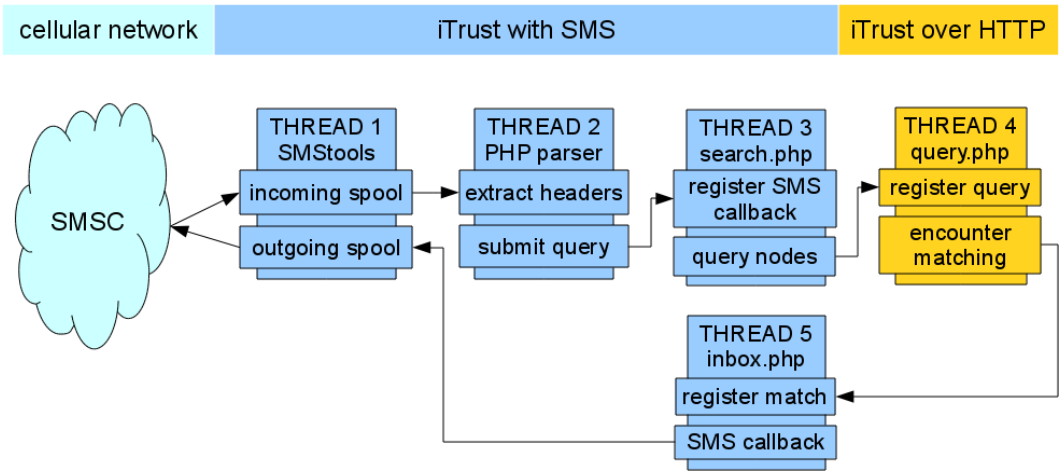



**Figure 4.2:** iTrust with SMS, showing the cellular network, the iTrust with SMS API and the iTrust over HTTP API.

main parts of iTrust with SMS, namely, the cellular network, the iTrust with SMS interface and the iTrust over HTTP interface. The blocks (numbered threads or spools) show only the application programming interfaces (APIs) relevant to the discussion of iTrust with SMS. Each block actually has many more API calls for the iTrust over HTTP implementation. Additionally, thread blocks are numbered to explain the examples. In a typical iTrust network, multiple threads can be running for each iTrust node.

### 4.2.1 Cellular Network

For the purposes of this discussion, the cellular network is modeled simply by the Short Message Service Center (SMSC), which the mobile phone service providers use to relay SMS messages. In Section 4.3, we expand the SMSC concept

slightly to include mobile phones to enable presentation of the user interface for iTrust with SMS.

Briefly, the SMSC is a store-and-forward entity in the network of the mobile phone service provider. When a user sends an SMS message, the message is stored in the SMSC and, when possible, it is forwarded to the intended destination. If the destination is unavailable, the message is spooled for later transmission.

For the iTrust network, there is no distinction between a single SMSC and multiple SMSCs that handle SMS relaying. iTrust does not require any service provider agreements or integration with existing mobile networks; it simply uses a mobile phone number like any mobile device seen by the SMSC.

### 4.2.2 iTrust with SMS

First and foremost, the iTrust with SMS implementation is an extension of the iTrust over HTTP implementation; SMS capabilities are added to the API and the iTrust over HTTP implementation remains intact and operational. Thus, an iTrust with SMS node can interact with both an Internet node and a cellular network node. The iTrust SMS-HTTP bridge node allows an SMS-enabled mobile phone in the cellular network to interact with iTrust over HTTP nodes on the Internet.

In addition to the custom code written for the iTrust SMS-HTTP bridge node, the open-source SMStools package is used to handle incoming and outgoing spool-

ing of SMS messages. SMStools offers several advanced features that are easily leveraged by iTrust, including SMS message formatting, header automation and message validation.

The iTrust SMS-HTTP bridge node requires a single hardware interface for sending and receiving SMS messages. Optionally, SMStools can be configured to handle multiple cellular modems from multiple cellular network providers and can spool the SMS messages accordingly. However, the typical iTrust configuration uses a single cellular modem to act as both the incoming and the outgoing SMS device, and SMStools to spool both incoming and outgoing SMS messages.

In iTrust with SMS, THREAD 1 consists of SMStools, which spools both incoming and outgoing SMS messages. Incoming SMS messages are registered with an event handler that triggers a command-line (*not* a Web server) PHP script in THREAD 2. Outgoing SMS messages are sent by writing a properly formatted plain text file and placing it in a specific SMStools monitored directory, so that an SMS response message is created and sent to the querying mobile device. Outgoing SMS messages are further explained below in the THREAD 5 functionality description.

The SMS message parser in THREAD 2 performs simple text processing to extract headers, such as the sender's mobile phone number and query. The extracted data are then packaged into an HTTP GET statement and submitted as a query to THREAD 3.

Particularly in THREAD 3, iTrust with SMS functionality is tightly integrated with existing iTrust over HTTP functionality; however, it remains distinct from the functionality of pure iTrust over HTTP nodes. Along with query text and timestamp information, the sender's callback phone number is registered to enable results sent to the iTrust SMS-HTTP bridge node to be relayed back to the mobile phone. The bridge node then queries the nodes in the iTrust network as if the query originated directly from the bridge node (not as an SMS-relayed query). The mobile phone number itself is not included in the query; only the iTrust SMS-HTTP bridge node is aware of the mobile phone number. Thus, the bridge node masquerades as an iTrust over HTTP node performing a routine search.

Nodes in the iTrust network execute the routines in THREAD 4 when queried for results. First, the query is registered so that duplicate relayed queries are ignored and, then, an encounter (match), if any, causes a response message to be sent back to the querying node. THREAD 4 exhibits typical iTrust over HTTP behavior; no SMS information or awareness is required from a node running this thread.

The SMS callback routine in THREAD 3 is perhaps the most extensive routine in the iTrust with SMS part. It has the dual function of pulling the source information and packaging that information appropriately before passing on the message to SMStools for spooling.

In THREAD 5, first the resource is automatically fetched from the source node and temporarily stored on the bridge node for further processing. Second, the document (if it is less than 140 bytes) is formatted for SMS, and the callback phone number of the original SMS querying user is added. Third, the message is written to an SMStools monitored directory, which further appends relevant message fields (*i.e.*, SMSC information, text formatting, *etc.*) before spooling the message for delivery (THREAD 1). Finally, the message is sent to the SMSC for delivery to the user's mobile device.

### 4.2.3 Interaction with iTrust over HTTP

The iTrust over HTTP implementation was fully described in Chapter 2. We highlight here only those API calls that involve interactions with iTrust with SMS.

When a query arrives at a node, the query is registered in THREAD 4 using the *register query* routine. If the query has been seen previously, processing stops as repeating an old query is not useful. If the query has not been seen previously, the query text is compared against a database consisting of metadata and URLs of the corresponding resources, using the *encounter matching* routine in THREAD 4. If the query keywords match locally stored metadata, the node responds to the requesting node with the URL. Note that, in this case, the requesting node is the iTrust SMS-HTTP bridge node; it is *not* the SMS mobile phone node.

### 4.2.4 A Typical SMS Request/Response Path

A typical path along which SMS request and response messages travel from the mobile phone and back again is described below (see Figure 4.2 to trace the message flow).

#### Sending the Request

A user sends an SMS request (query) message from his/her mobile phone with a simple text query. After being relayed by the SMSC, the SMS message enters the iTrust SMS-HTTP bridge node through a cellular hardware interface (such as a cellular modem) and is held in the incoming spool (THREAD 1). A new message in the incoming spool triggers an event handler (THREAD 2), which then loads a PHP script to process the spool and extract the user's mobile phone number and text query. The mobile phone number is registered for callback purposes (THREAD 3), and the query enters the iTrust network exactly as if it were originated by an iTrust over HTTP node. The query is relayed through the iTrust network until an encounter occurs (THREAD 4).

#### Receiving the Response

A response message is sent from an iTrust over HTTP node to the iTrust SMS-HTTP bridge node (THREAD 5). After normal processing by iTrust, the resource is fetched and placed in local storage. The locally stored resource (or a URL for

the locally stored resource, if the resource is large) is further processed into an SMS message, placed into the outgoing spool and relayed to the SMSC (THREAD 1). The user receives an SMS message, sent from the iTrust SMS-HTTP bridge node.

### 4.2.5 API Function Call Swapping and Race Conditions

In Figure 4.2, under THREAD 3, there are two API function calls: *register SMS callback* and *query nodes*. The iTrust over HTTP nodes (where a *register SMS callback* is simply a register query callback) have the order of these two calls swapped for performance reasons. In practice, querying a node before registering the query leads to better performance in the Apache prefork model. This model inherently prevents the occurrence of a race condition, because the query is registered long before another node responds with a result. This behavior holds particularly for threads numbering in the several thousands; however, in practice, even a self-query on a single node does not result in a race condition.

The iTrust SMS-HTTP bridge node has a stricter requirement. An iTrust with SMS node must *always* register the SMS callback phone number before querying another iTrust node. Otherwise, an iTrust node that is not SMS-capable might respond to a query before the callback phone number is registered. In this case, the particular response is not relayed to the mobile phone; however, future responses,

that arrive after the SMS callback phone number has been registered, will be relayed.

Simply swapping the order to that shown in Figure 4.2 prevents a race condition from occurring.

## 4.3 iTrust with SMS User Interface

The addition of iTrust with SMS to iTrust over HTTP requires not only an additional bridge mechanism on the iTrust nodes, but also a new interface to allow the mobile phone user to interact with the iTrust network. Whereas iTrust over HTTP requires the use of a Web browser to search and retrieve documents, iTrust with SMS needs a more user-friendly mobile phone interface that conforms to the expectations of the user for a typical Instant Messaging service. For iTrust with SMS, we compare a generic SMS Instant Messaging interface with a custom-built Android interface for iTrust with SMS.

As an example, we consider a protest demonstration scheduling service that periodically distributes meeting locations and times to iTrust nodes. For each demonstration, there exists a file that includes basic information such as meeting location and time. A query from one iTrust node begins a search among other participating nodes in the iTrust network, and an encounter returns the demonstration named file that includes the meeting information. In particular,

we consider the case in which a user searches for meeting information around *Tahrir Square* in Cairo, Egypt.

### 4.3.1 Using the Generic Instant Messaging Interface

The interface for iTrust with SMS is minimalistic in both function and use, compared to the Web interface for iTrust over HTTP. Requests (queries) are simply SMS messages that are sent to the mobile phone number of the iTrust SMS-HTTP bridge node; similarly, responses are SMS messages containing document data sent back to the user. There is no user hardware requirement apart from an SMS-capable mobile phone; the SMS message may be sent to a dumb phone or a smart phone, with the user experience remaining consistent. Because the primary focus of a user of iTrust with SMS is simply to make a query, there is no interface for modifying the membership, adding resources, or configuring user parameters, as in the iTrust over HTTP interface.

Figure 4.3 shows an image of a typical iTrust with SMS interaction between a mobile user and an iTrust node. This particular screen shot uses the standard built-in SMS application bundled in the Android platform (specifically, Android version 2.1); however, apart from aesthetics, the interaction is the same for iOS, WebOS, Symbian, *etc.* Note that the only information required to interact with an iTrust node, apart from the query, is the mobile phone number of the iTrust node (which is partially obscured). This particular Instant Messaging interface presents


**Figure 4.3:** iTrust with SMS, using the generic Instant Messaging interface.

all SMS messages between the same callers in a single scrolling conversational type format. In this example, the display shows the user query *Tahrir Square* message sent to the iTrust node. A response message is sent back from the iTrust node to the user approximately one minute later (as shown in the last message); this result (or hit) is the data that correspond to the user's search keywords.

Note that the data itself are returned to the user without reference to the URL, document file name, or address of the source node of the document. This presentation is consistent with the iTrust with SMS functionality, which requires that the iTrust SMS-HTTP bridge node itself must fetch the document, package it in an SMS-compatible format and send back the result. In contrast, the iTrust

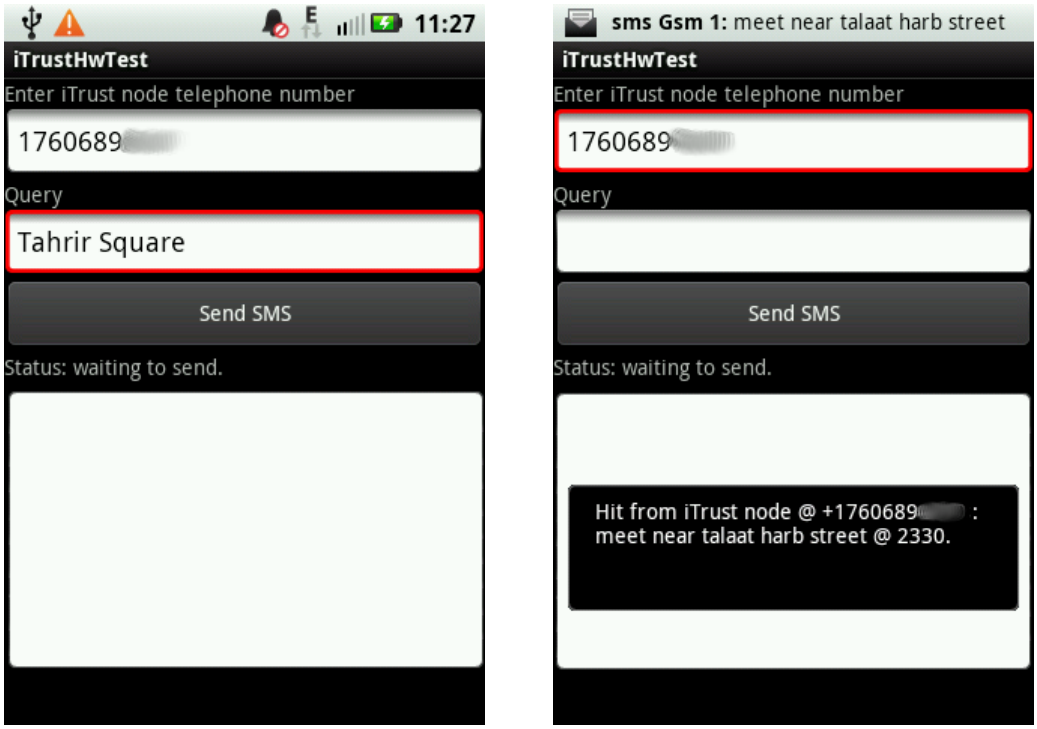
over HTTP interface simply presents a list of hits and does not fetch the document data automatically.

This simple and direct interaction makes it easy to carry on a conversation of sorts with an iTrust node by simply asking questions (submitting queries) and reading answers (hit data).

### 4.3.2 Using the Custom Android Interface

The custom Android application for interacting with an iTrust with SMS node is a hybrid of the generic SMS Instant Messaging interface and the iTrust over HTTP interface. Figure 4.4 shows the submission of a query from the SMS-capable mobile phone and the returned result from the iTrust SMS-HTTP bridge node, respectively. The custom Android interface for iTrust over SMS enhances the generic SMS Instant Messaging interface in that it provides: familiarity for users accustomed to iTrust over HTTP, preset mobile phone numbers to iTrust SMS-HTTP bridge nodes and a framework for handling non-textual result data.

Figure 4.4(a) shows the entry of a query into a text editing area that is similar to that in the iTrust over HTTP search interface. Above the query is the pre-entered mobile phone number of the iTrust SMS-HTTP bridge node. Although this interface is a minimal enhancement to the generic SMS interface, the rapid and transient nature of most SMS interactions favors features that reduce extraneous information not related to the SMS message itself. Additionally, once the query




(a) (b)

**Figure 4.4:** iTrust with SMS with the custom Android interface: (a) Searching for information and (b) Viewing a hit.

is sent, the query text area is cleared, so that the user can easily enter another search query.

Figure 4.4(b) shows the resulting data returned from the iTrust SMS-HTTP bridge node; the result is the same as the result for the generic SMS interface. The resulting data are displayed in text format; however, alternate formats can be handled by the built-in framework. For example, a Portable Document Format (PDF) file sent over SMS would be passed on to the Android platform, presumably to be opened by a PDF reader application available on the mobile phone. In this

case, the user would need a separate reader application appropriate to the file type. The iTrust system searches and retrieves all files, regardless of format (as long as the metadata are properly generated); however, the user is responsible for appropriate decoding.

## 4.4 iTrust over SMS Protocol Implementation

The iTrust over SMS protocol allows any SMS-capable mobile device to communicate with any other such device in the iTrust over SMS network, regardless of hardware or software platform. The iTrust over SMS protocol is described below in terms of: (1) message formats, (2) metadata distribution message types and examples, and (3) search and retrieval message types and examples.

Although the iTrust over SMS protocol is platform agnostic, the first implementation of the protocol was developed in conjunction with a Java-based iTrust over SMS implementation for the Android platform. Therefore, for completeness, first we briefly describe the iTrust over SMS implementation on Android, and then we describe how that implementation follows the iTrust over SMS protocol, before we describe the protocol itself.

### 4.4.1 iTrust over SMS Implementation on Android

Figure 4.5 shows the Android-based implementation of iTrust over SMS, which comprises the user interface, the iTrust over SMS API, and the mobile (cellular)

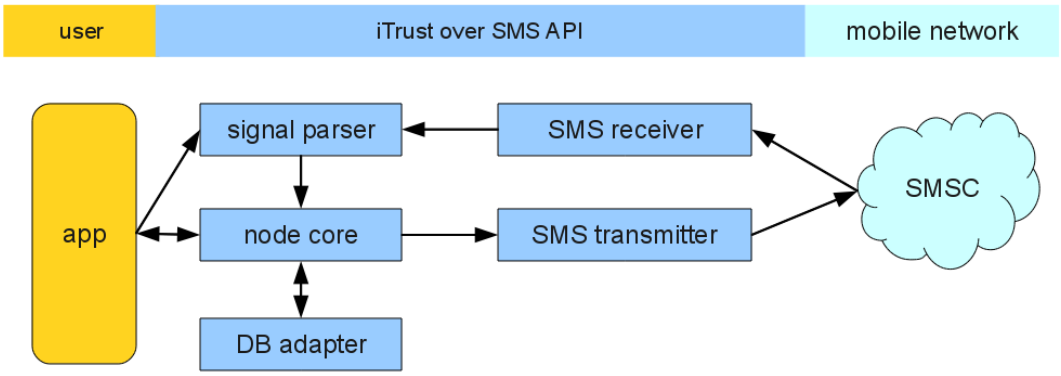



**Figure 4.5:** The iTrust over SMS API and components, along with the user interface and the mobile network.

network. The user interface might be an Instant Messaging (IM) application, document sharing application, or (in our case) a test application for experimentation with the iTrust over SMS protocol. The mobile network in Figure 4.5 is identical to that in Figure 4.2; it can be regarded as interconnected SMSC entities that transport messages between mobile peers.

The iTrust over SMS implementation consists of five components, two components handle the basic input and output of messages and three components handle the primary iTrust over SMS functionalities. The SMS receiver component handles input, and the SMS transmitter component handles output. Previously, we described the individual functions of iTrust with SMS, by tracing how a message flows between threads. Here, we describe how an incoming message is acted on, and an outgoing message is generated, in iTrust over SMS.

A message is received by the SMS receiver component (based on an Android *BroadcastReceiver* intent), which passes the message on to the signal parser component. Correspondingly, a message is created by the node core component, which passes the message on to the SMS transmitter component. The signal parser component decodes text messages, and the node core component acts on or responds to messages (making encounters, adding nodes to the iTrust membership, relaying queries, distributing metadata, *etc*). If a message exceeds the SMS limit of 140 octets, an Android utility splits the message into chunks and reassembles multi-part SMS messages on arrival (through the use of the SMS user data header). Thus, we say that the signal parser *reads* the iTrust over SMS protocol, and the node core (which might have to send a response message) *writes* the protocol. The database adapter component handles the bookkeeping tasks required of the node core component by the use of various SQLite database tables.

An important Android-related concern deals with the transmission of SMS text messages. Unfortunately, there is a long-standing bug in Android, which prevents the proper transmission of certain characters in SMS text messages. Specifically, the characters [ ] { } cannot be correctly sent in an SMS text message, because the GSM alphabet table is incorrectly set by Android. According to the 3G TS 23.038 version 3.3.0 technical specification, the SMS packing scheme (specifically, the packing of 7-bit characters) allows an extended 7-bit alphabet to be used. The GSM 7-bit alphabet extension table includes the characters [ ] { }; however,

because the table cannot be correctly enabled by Android, the characters are effectively unavailable. We produced a work-around that forces the characters *[ ]* { } to be transformed into the characters *( )* <>; the latter characters are not in the extension table, and so the default table can be used instead. Because Android supports the default table, the message is sent correctly. Thus, the JavaScript Object Notation (JSON) string representation of the metadata is transformed from valid JSON to quasi-JSON and placed in the outgoing message string ready for the SMS transmitter component to send the data. On arrival of the message, the receiving node must perform the reverse transformation (*i.e.*, replace all *( )* <> characters by *[ ]* { } characters), before processing the JSON data.

With this explanation of the Android implementation and its connection with the iTrust over SMS protocol, and the quasi-JSON work-around for Android, we now delve into the protocol itself.

### 4.4.2 Message Formats and Types

Table 4.1 illustrates the two basic message formats of the iTrust over SMS protocol; the primary distinction is that one format accommodates three parameters and the other format accommodates two parameters. The table depicts the body of a text message contained in an SMS text message, in this case, a single text identifier followed by several text parameters, each separated by the @ character.

**Three parameter SMS message**

| Identifier | Delimiter | Parameter 1 | Delimiter | Parameter 2 | Delimiter | Parameter 3 |
|---|---|---|---|---|---|---|
| | @ | | @ | | @ | |

**Two parameter SMS message**

| Identifier | Delimiter | Parameter 1 | Delimiter | Parameter 2 |
|---|---|---|---|---|
| | @ | | @ | |

**Table 4.1:** The message formats of the iTrust over SMS protocol for three parameter and two parameter SMS messages.

Table 4.2 shows the seven different message types of the iTrust over SMS protocol; the first four are used for search and retrieval and the remaining three are used for metadata distribution. The left-most column lists the seven message types; lower-case plain text represents string literals, and italicized angle-bracketed text represents placeholders for variable text. Cells labeled *unused* are reserved for future use. The search and retrieval functions are separated into two message types: messages with identifier *itq* search or query for information, and messages with identifier *itr* return or retrieve information. The remaining three metadata distribution messages have the identifier *itm*, and use only two parameters.

An important design consideration for the iTrust over SMS protocol is that a message should be relatively easy for humans to understand, even if doing so increases the complexity of the signal parser. With the three message identifiers

| Message | Identifier | Parameter 1 | Parameter 2 | Parameter 3 |
|---|---|---|---|---|
| SEND_QUERY | itq | *<caller_number>* | *<query_id>* | *<query>* |
| NOTIFY_MATCH | itr | *<source_number>* | *<query_id>* | *<resource_id>* |
| REQUEST_RESOURCE | itq | now | *<query_id>* | *<resource_id>* |
| SEND_RESOURCE | itr | data | *<query_id>* | *<data>* |
| NOTIFY_METADATA | itm | *<source_number>* | *<expiry_date>* | |
| REQUEST_METADATA | itm | pull | *unused* | |
| SEND_METADATA | itm | push | *<data>* | |

**Table 4.2:** The message types of the iTrust over SMS protocol.

*itq*, *itr*, and *itm*, a human can easily understand whether a particular iTrust over SMS protocol message is a query, reply, or metadata message.

The signal parser processes each incoming message; messages with identifiers *itq* or *itr* are parsed for three parameters, and messages with identifier *itm* are parsed for two parameters. Programmatically, each message is considered an enumeration, and processing is switched on a case-by-case basis. Parameters are parsed left to right, which creates string tokens delimited by the character @; parsing ceases after the final delimiter is found. Therefore, messages with three parameters are fully processed after the first three leftmost occurrences of @, and messages with two parameters are fully processed after the first two leftmost occurrences of @. Doing so allows the final parameter to use the character @ any number of times without breaking the parsing rules (*i.e.*, it is not necessary to escape @ in the final parameter). For example, both of the messages *itq@aaa@bbb@ccc* and *itq@aaa@bbb@ccc@ddd* are valid messages; the former has final parameter *ccc*, and the latter has final parameter *ccc@ddd*. Two parameter

messages also follow the same pattern. For example, the messages *itm@111@222* and *itm@333@444@555* are both valid messages; the former has final parameter *222*, and the latter has final parameter *444@555*. The rationale behind this particular way of parsing will become evident when the seven message types are described.

### 4.4.3 Metadata Distribution for iTrust over SMS

Although searching for and retrieving information in the iTrust network constitutes the bulk of the time spent by users or applications, first the metadata must be distributed before encounters for searched information can occur. Below, we describe the three messages involved in distributing metadata, graphically show how the messages are sent between iTrust nodes and, finally, we present an example with actual SMS text messages.

#### Metadata Distribution Message Types

#### NOTIFY_METADATA

The NOTIFY_METADATA message notifies a node in the iTrust network that metadata are ready to be read. The message can be sent in two different ways: by the source node of the metadata or relayed through another node. If the message is sent by the source node of the metadata, the source node creates the two parameter message with the source phone number and expiration date. The source

phone number is the mobile phone number of the source node that stores the resource described by the metadata, and the expiration date is a Unix timestamp (number of seconds, in the Unix epoch) after which reading the metadata is no longer useful. For example, if the source node holds data for meeting information, the meeting location and time are not useful the day after the meeting. If the message has been relayed, a node that receives the NOTIFY_METADATA message saves the source phone number and expiration date and then relays the message to some other node in the iTrust membership. There is no need for the relaying node to modify the message; all necessary information has already been placed in the message by the source node. When retrieving information, a node may prioritize retrieval based on source phone number or expiration date; the reasons for prioritization depend on the user application. In particular, the user application may ignore the expiration date and retrieve metadata whenever it is convenient; it is neither expected nor required that a receiving node immediately act on a NOTIFY_METADATA message.

**REQUEST_METADATA**

When a node wants to receive metadata, it sends a REQUEST_METADATA message to the source node that holds the desired resource information. The first parameter is filled with the string literal *pull*; the second parameter is unused. On receiving a REQUEST_METADATA message, the source node immediately creates and sends back a SEND_METADATA message.

**SEND_METADATA**

When a node receives a request for metadata, it immediately creates and populates the SEND_METADATA message. The first parameter is filled with the string literal *push*, and the second parameter is filled with the quasi-JSON encoded metadata resource/keyword pairs. The quasi-JSON string should have any characters not in the GSM default alphabet table removed to avoid conflicts (to work around the Android limitation previously discussed). As explained earlier, parsing the two parameter message type stops after the first two leftmost @ symbols are found; therefore, the metadata itself does not need to be re-evaluated for @ string escapes. On receiving a SEND_METADATA message, a node decodes the second parameter and inserts the resource/keyword pairs into its database.

If there is a third node (or other previous node in the relay chain), the participation of the relay node ends immediately after the NOTIFY_METADATA message is relayed.

#### 4.4.3.1 Examples of Metadata Distribution

Figure 4.6 shows example diagrams that depict the flow of messages between nodes during the distribution of metadata in the iTrust over SMS network. Parts A and B represent independent interactions; messages in the two parts do *not* chronologically precede or follow one another. Node $S$ is the source node that has resources locally stored; nodes $Z$ and $Y$ are other nodes that receive metadata.

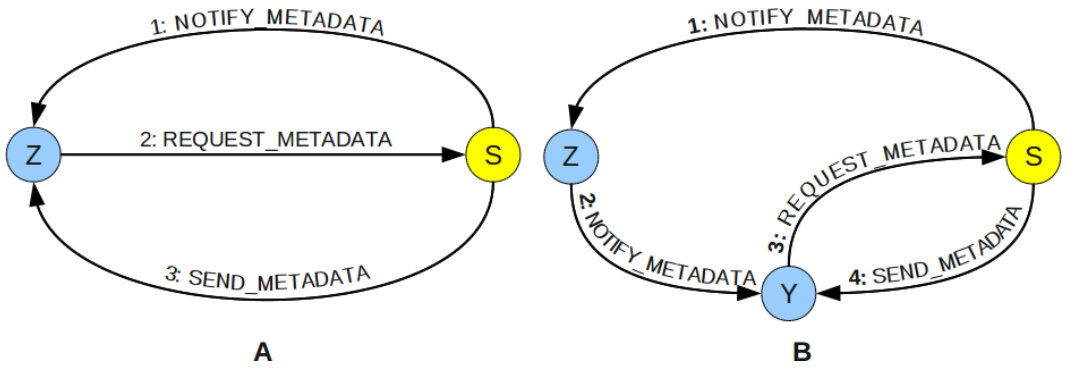



**Figure 4.6:** Metadata distribution message flow for iTrust over SMS.

The lines and arrows show the directions and the destinations of the messages; each line is labeled with the message sent. The numbers preceding the messages signify the order in which the messages are sent; the numbers are *not* part of the messages sent.

*Part A: Metadata Distribution between Two Nodes.* Node $S$ sends a NOTIFY_METADATA message to node $Z$ informing $Z$ that metadata are ready for $Z$ to read at $Z$'s convenience. $Z$ sends a REQUEST_METADATA message to $S$ requesting metadata to be sent immediately. $S$ creates the metadata and sends it to $Z$ in the SEND_METADATA message.

*Part B: Metadata Distribution between Three Nodes.* Node $S$ sends a NOTIFY_METADATA message to node $Z$ as indicated by message *1*. $Z$ relays the message to node $Y$ as indicated by message *2*. At $Y$'s convenience, $Y$ sends a REQUEST_METADATA message to $S$ requesting metadata to be sent immediately. $S$ creates the metadata, and sends it to $Y$ in the SEND_METADATA message.

#### 4.4.3.2 Example of SMS Text Messages for Metadata Distribution

Figure 4.7 shows an example of how SMS text is transmitted between nodes in the iTrust membership during metadata distribution. The nodes and other descriptions are the same as those in Figure 4.6; messages are numbered in the order in which the messages are sent.

Node $S$ sends message *1* to node $Z$; the first parameter is filled with $S$'s mobile phone number. $Z$ sends message *2* to $S$; the first parameter is filled with the string literal *pull*. $S$ sends message *3* to $Z$; the first parameter is filled with *push*, and the second parameter is filled with quasi-JSON metadata.

Note that, if message *1* had been relayed by an intermediate node, $Z$ would know to call $S$ at *15559988776*, because that phone number was included in message *1*.

### 4.4.4 Search and Retrieval for iTrust over SMS

Search and retrieval of resources involves four types of iTrust over SMS messages: two query messages and two response/retrieval messages. Below, we describe these four types of messages, give an example of message passing between nodes, and finally present an example of actual SMS text messages sent for search and retrieval of resources.

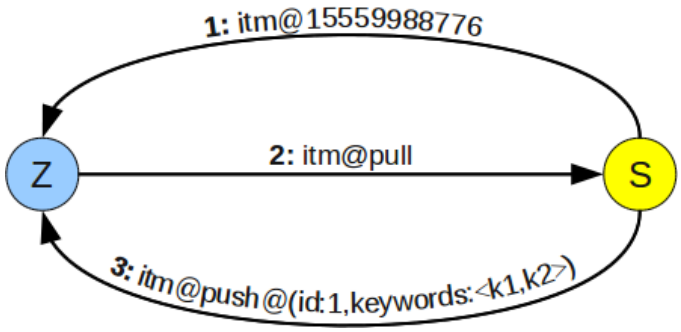



**Figure 4.7:** SMS text message example of metadata distribution for iTrust over SMS.

#### 4.4.4.1 Search and Retrieval Message Types

**SEND_QUERY**

The SEND_QUERY message is used to query a node in the iTrust network for resources. The message contains three parameters: call number, query id, and query text. The call number is the mobile phone number of the node that is issuing the query. The query id is any text string that nodes in the iTrust membership use to track the query. The same query id is used for all four iTrust over SMS messages pertaining to the search request; in effect, it is a global identifier. The query text should be checked, by the user application, to ensure that it is within the GSM default alphabet table.

If a node is originating the query, it creates these three parameters and then sends the message. If a node is relaying the query, it relays the message without modification; it can use the query id to prevent relaying the same message more than once (to prevent network flooding). On receiving a SEND_QUERY message,

a node immediately checks whether an encounter has occurred by comparing the query text against its available resources. If an encounter has indeed occurred, it sends a NOTIFY_MATCH message; otherwise, it takes no further action.

**NOTIFY_MATCH**

When a node has an encounter, it responds to the original querying node with a NOTIFY_MATCH message. The message is sent directly to the call number in the first parameter of the SEND_QUERY message; it is not sent to the node that relayed the query. The NOTIFY_MATCH message contains three parameters: the source phone number, the query id, and the resource id. The query id is the same as that in the SEND_QUERY message; again, it is a global identifier for the query and may be used by the application for various purposes. For example, an application might ignore a query that it did not originate, to protect against rogue nodes that send spurious NOTIFY_MATCH messages.

If the resource is stored locally, the source phone number is the mobile phone number of the node at which the encounter occurred (*i.e.*, the node that received the SEND_QUERY message and is about to send the NOTIFY_MATCH message), and the resource id is from the local resource table. If the resource is *not* stored locally, the source phone number is the mobile phone number of the node where the resource *is* stored and the resource id is from the resource table of that node.

Receiving the NOTIFY_MATCH message requires relatively little processing. Because the node that sent the NOTIFY_MATCH message did the processing

required to find the node on which the resource is located and set the message parameters accordingly, the only required action is to save the values for further processing before discarding the message. The user or application can decide when to retrieve the message at the source phone number; retrieval of the document is not mandatory and is done at the convenience of the user or application, using the REQUEST_RESOURCE message.

**REQUEST_RESOURCE**

When a node wants to retrieve a particular resource, it directly contacts the node that stores the resource, using the REQUEST_ RESOURCE message. The message contains three parameters: the string literal *now*, the query id, and the resource id of the stored resource on the receiving node. Although the query id is not strictly needed in this case (it's possible that the associated SEND_QUERY message was never relayed to the receiving node), it is still sent for possible use by the application. On receiving a REQUEST_RESOURCE message, a node immediately looks up the resource using the resource id and sends it using the SEND_RESOURCE message. If the resource id does not exist in its table, the node ignores the message and stops processing.

**SEND_RESOURCE**

When a node receives a REQUEST_RESOURCE message, it immediately gets the resource data and packages it for transmission in the SEND_RESOURCE message. The SEND_RESOURCE message has three parameters: the string literal *data*, the

query id, and the data itself. Again, the query id is sent only for optional tracking by the user application that interfaces with iTrust over SMS. Transmitted data in the third parameter of the SEND_RESOURCE message can be in any format suitable for the application as long as it fits within the GSM default alphabet table (again due to the Android bug). The iTrust over SMS API provides several convenience functions for inserting (extracting) plain text data into (from) the message, which make sending (receiving) plain text trivially simple. To send (receive) custom data apart from plain text, the user application simply needs to escape (unescape) that custom data.

Note that only the original querying node is involved with each of the four message types. Intermediate nodes may relay queries or send notifications of a match, but their involvement ends immediately thereafter.

#### 4.4.4.2 Examples of Search and Retrieval

Figure 4.8 shows example diagrams that depict the flow of messages between nodes during the search and retrieval of resources in the iTrust over SMS network. Parts A, B, and C represent independent interactions; messages in the three parts do *not* chronologically follow or precede one another. Node $S$ is the source node that has the resources locally stored; node $Q$ is the querying node that sends the original search query; and nodes $Z$ and $Y$ are other nodes in the iTrust network. Again, the lines and arrows show the directions and destinations of the messages;

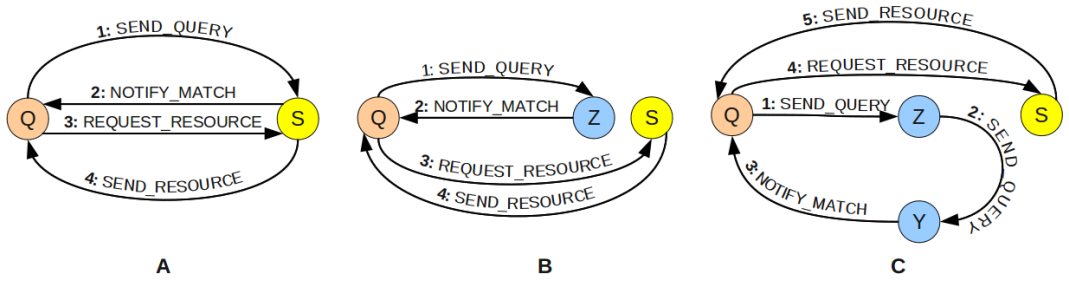



**Figure 4.8:** Search and retrieval message flow for iTrust over SMS.

the numbers preceding the messages signify the order in which the messages are sent.

*Part A: Search and Retrieval between Two Nodes.* Node $Q$ sends a SEND_QUERY message to node $S$. $S$ has an encounter, and responds to $Q$ with a NOTIFY_MATCH message. When it is convenient, $Q$ sends a REQUEST_RESOURCE message to $S$. On receiving the REQUEST_RESOURCE message, $S$ sends the resource to $Q$ in the SEND_RESOURCE message.

*Part B: An Intermediate Node Has an Encounter.* At some prior time, node $S$ distributed metadata to node $Z$. Node $Q$ sends a SEND_QUERY message to $Z$; $Z$ immediately has an encounter as a result of $Q$'s query and the metadata distributed by $S$. $Z$ sends a NOTIFY_MATCH message to $Q$. When it decides, $Q$ sends a REQUEST_RESOURCE message to $S$. $S$ sends the resource to $Q$ in the SEND_RESOURCE message.

*Part C: A Search Query Is Relayed.* At some prior time, node $S$ distributed metadata to node $Y$. Node $Q$ sends a SEND_QUERY message to node $Z$ as shown

by message *1*. *Z* does not have a match but relays the SEND_QUERY message to *Y* as shown by message *2*. *Y* immediately has an encounter between *Q*'s query and the metadata distributed by *S*, and sends a NOTIFY_MATCH message to *Q*. At its convenience, *Q* sends a message REQUEST_RESOURCE TO *S*. *S* sends the resource to *Q* in a SEND_RESOURCE message.

#### 4.4.4.3 Example of SMS Text Messages for Search and Retrieval

Figure 4.9 shows an example of how SMS text is transmitted between nodes in the iTrust membership during a typical search and retrieval interaction. The nodes and other descriptions are the same as those for Figure 4.8; messages are identified by the order in which they are sent.

Node *Q* sends message *1* to node *S* with three parameters: caller phone number *15551234567*, query id *r4nd0m1d*, and query *tahrir square*. Immediately, *S* has an encounter and knows to call back node *Q* at *15551234567* with message *2*, which contains the source phone number *15550011223*, query id, and resource id *456*. When convenient, *Q* responds to *S* by sending message 3, which contains the query id and resource id *456* of the resource that it wants to retrieve. *S* immediately responds to *Q* by sending message 4, which contains the query id and the resource data *meet near talaat harb street*.

Note that, if message *1* had been relayed, any node that had an encounter would know to contact node *Q* at *15551234567*, because the phone number is

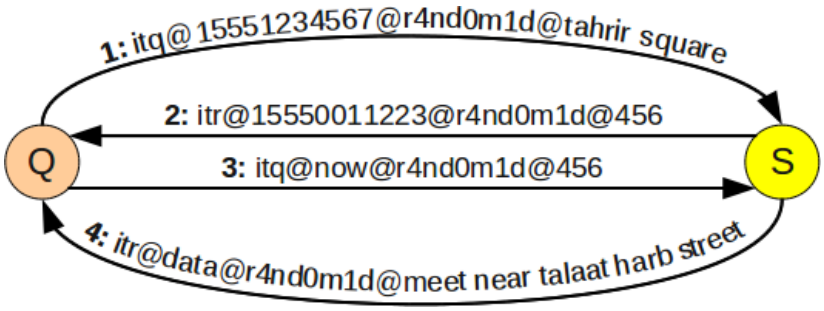


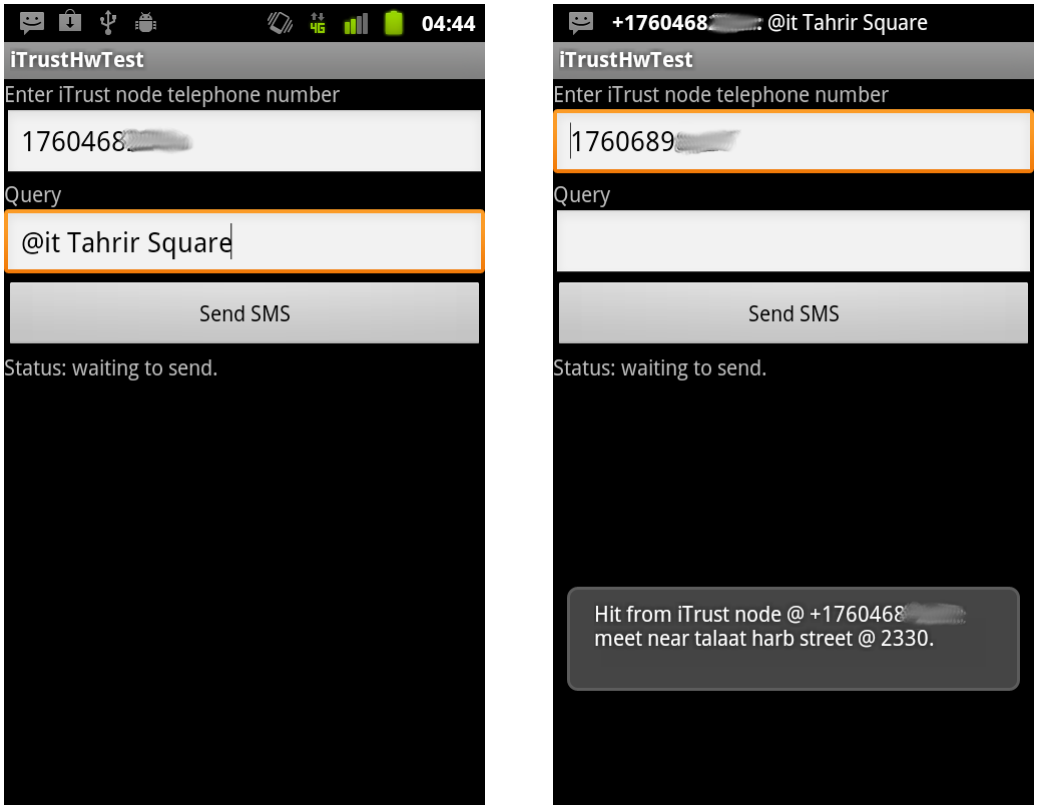

**Figure 4.9:** SMS text message example of search and retrieval for iTrust over SMS.

included in the message. Furthermore, if the encounter had occurred on metadata held by another node, the source phone number *15550011223* in message *2* would tell $Q$ which node to contact to retrieve the resource.

## 4.5 iTrust over SMS User Interface

The custom Android user interface shown in Figure 4.4 for iTrust with SMS was re-purposed for iTrust over SMS. The custom Android user interface for iTrust over SMS is shown in Figure 4.10.

From the end user's perspective of searching for and retrieving information, the interfaces are the same. The underlying difference is that instead of communication between a mobile phone and an iTrust SMS-HTTP bridge node, communication occurs directly between mobile phones in the iTrust over SMS network. Appropriate defaults were chosen for functions inherent to iTrust over SMS but not configurable in the custom user interface originally developed for iTrust with




(a) (b)

**Figure 4.10:** iTrust over SMS with the custom Android interface: (a) Searching for information and (b) Viewing a hit.

SMS. For example, metadata distribution is done automatically on application startup for iTrust over SMS (because iTrust with SMS has no need for this function). Additionally, because Android uses the *BroadcastReceiver* intent to act as an event handler for incoming messages, as explained earlier, the custom Android interface does not need to register another event handler. The event handler written for the iTrust over SMS API suffices.

In addition to the above simple user interface for searching and retrieving resources, several use cases were considered when developing an Android application specifically targeted for ease of use for the average non technical user. The complete use case details and usage of the application are found in Section 4.7; however, we briefly review them here, because the aforementioned iTrust over SMS user interface may also be used by at least one type of user (*i.e.*, sporadic). We identify four main types of users from a sporadic document searcher to avid document searcher and detail, with use case scenarios and application descriptions, how the application accommodates each usage pattern. In particular, the application is designed as simple as possible and resembles the built-in search functionality present on the Android platform; other functions such as query managing and document managing are included for those users wanting greater control of the document retrieval process. Technical details, such as message identifiers, are completely hidden from the smart phone user. Critically, the ease of use for smart phone users does not inhibit iTrust over SMS functionality for low-cost *dumb* phone users who are already accustomed to texting messages using a thumbpad. While the average smart phone user demands the convenience of an app, the average dumb phone user willingly thumb types SMS messages for enhanced mobile device functionality such as P2P document searching.

## 4.6 iTrust over SMS Android User Interface

To make it easier to understand the common use cases of a typical iTrust over SMS user, in Section 4.7, first we describe the Android user interface for iTrust over SMS. In Section 4.5, we showed a minimal user interface (itself a re-purposed user interface for iTrust with SMS), which is adequate for dumb phone users or mobile devices with limited processing resources. In contrast, the Android iTrust over SMS interface described below is designed specifically for smart phone users with modern graphical user interfaces and touch gestures; in Section 4.7, we will show that this more powerful interface better serves a wide range of use cases.

We present example screen shots for the Android application (colors are inverted for legibility). Implementers of other Instant Messaging applications may choose to provide similar functionality, in addition to their existing features, by studying these examples as they fully illustrate the features of iTrust over SMS.

The iTrust over SMS user interface for Android comprises five distinct Java classes, each of which consists of both a layout file written in XML and an activity file containing event handlers written in Java. The layout file specifies the location and style (color, font, *etc.*) of widgets placed on the mobile device screen, as well as attribute identifiers for Android. The Android identifiers can be used for various purposes, such as binding Java resources to program subroutines during run-time, or even simple string value replacement (*e.g.*, internationalization). An event handler is triggered when a user interacts with a widget in the layout (*e.g.*,

a button tap triggers an event handler for the *onClick* method). From the user's perspective, an activity is simply the layout of widgets on the screen that allows interaction with iTrust over SMS; for this reason, we use the terms activity and screen interchangeably in the rest of this section.

Below, we briefly describe and illustrate each of the screens that a user can use to distribute, search for and retrieve information in the iTrust over SMS network.

### 4.6.1 Existing Search

Figure 4.11 shows the default screen when a user starts the iTrust over SMS Android application. This screen lists the searches that the user has made from the mobile device. If the number of searches exceeds the space on the screen, the screen automatically allows vertical scrolling to accommodate the display of more searches.

The screen in Figure 4.11 lists all searches that are explicitly *initiated* on the mobile device; searches from other nodes that passed through this node by way of query relaying are *not* shown here. This design choice fits the expectation of the typical user, who is concerned with the searches that he/she started and not the searches that other users started.

Tapping on any search list item immediately displays detailed information about that search, as shown in Figure 4.13. For example, tapping on the search

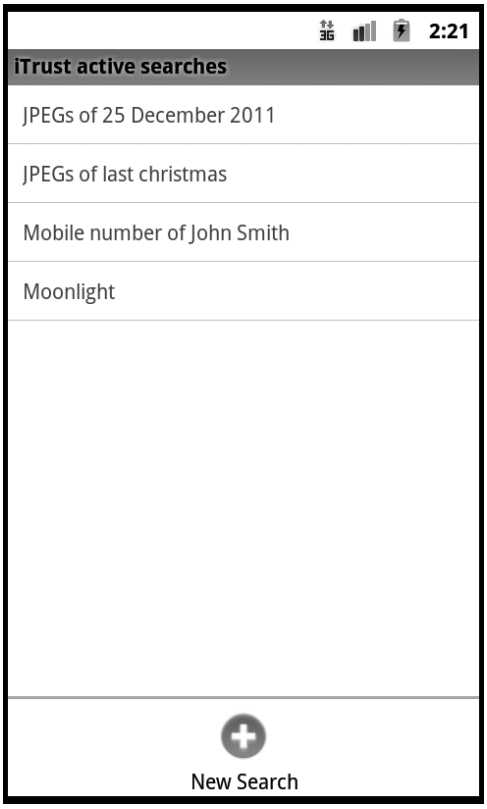



**Figure 4.11:** Screen that lists all searches sent from the local iTrust node.

item *Mobile number of John Smith* immediately switches to detailed information about that particular search query.

At the bottom of Figure 4.11 is a pop-up menu that is enabled by pressing the *menu* hard/soft button present on the Android mobile device. By default, this menu and the *New Search* menu item are not visible on the screen. However, a user can press the *menu* button, which causes the menu to pop-up (pressing the *menu* button again causes the menu to disappear). When the *New Search* button is pressed, the user is taken to Figure 4.12.

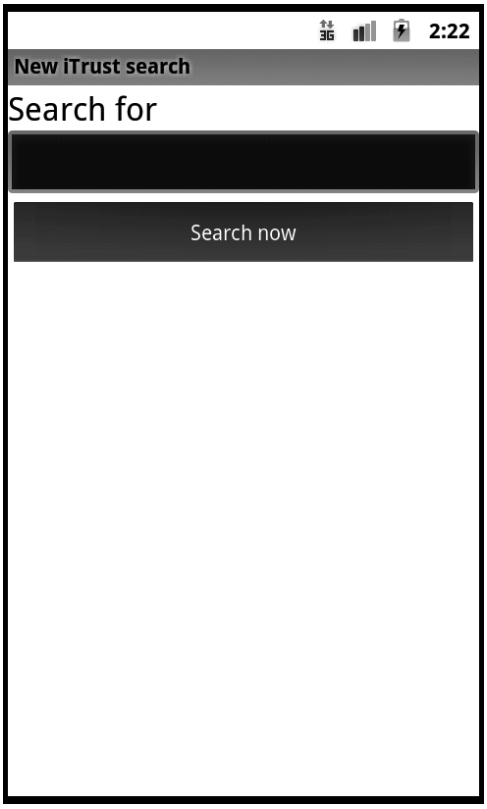



**Figure 4.12:** Screen to initiate a new search query from the local iTrust node.

### 4.6.2 New Search

Figure 4.12 shows the screen used to initiate a search across the iTrust over SMS network. A user is brought to this screen by tapping the *New Search* menu item found on the screen in Figure 4.11. The design is purposely simple and feature limited. Just as a typical Web user prefers a single text box to enter a query, the typical mobile phone user prefers a simple interface to enter search queries.

Tapping the *Search now* button takes the user back to the list of active searches sent from the device, as shown in Figure 4.11. Meanwhile, the search is automatically serviced by the iTrust over SMS library and relayed by the iTrust over SMS search service.

### 4.6.3 Search and Retrieval Details

When the user taps on a search item on the screen in Figure 4.11, that particular search request (query) is displayed in detail on the screen in Figure 4.13.

Near the top of the screen in Figure 4.13 is the text of the search request followed by two important fields: the date/timestamp when the search was initiated and the number of nodes to which the search request was relayed. The date/timestamp enables the user to recall how old the search is; the date is *not* used for priority ranking. The number of nodes displayed is the number of nodes to which the request is directly sent by this node (although, because SMS is used, only a best-effort service is provided). This number is the *minimum* number of nodes to which the search request is distributed; each such node may relay the query to yet another node.

Below the data/timestamp and the number of nodes, and separated by a thin line, is the space reserved for showing a list of matches reported back to the node. When another node has an encounter or match and reports back to the node originating the search, this space displays a tappable list of items along with the

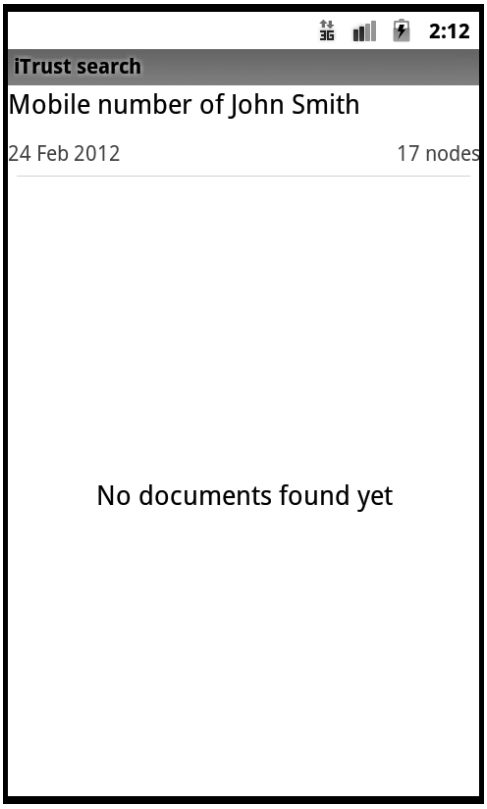



**Figure 4.13:** Screen showing the detailed information for a particular iTrust search.

node address of the node where the match occurred. Simply tapping on the list item triggers an automatic fetch of the document by the iTrust over SMS retrieval service; the resource is then displayed on the screen or optionally saved for later viewing. When no information is found, the screen displays the *No documents found yet* notice to the user.

### 4.6.4 Nodes

Figure 4.14 is similar to Figure 4.11 except that, instead of displaying the list of searches, it displays the list of nodes (or membership) of the local node. The pop-up menu near the bottom of the screen allows the user to enter a new node address in a pop-up dialog text box (not shown here). Explicit addition of node addresses by the user is not a common occurrence, but addition of node addresses is often performed automatically by the iTrust over SMS library; therefore, a new screen is not required for this task, as it is for adding new searches in Figure 4.12. A simple entry dialog box suffices.

As searches are relayed through the membership of the iTrust over SMS network, the originating query node address is saved by the iTrust library on each node that receives the relayed query. Thus, node addresses may be added to a node's membership without the node's making direct contact with those nodes.

### 4.6.5 Preferences (top)

The preferences screen shows the configurable settings that a user may modify to change the behavior of the iTrust over SMS service running on his/her mobile device. Because the preferences activity is longer than some of the other activities, it must be vertically scrolled on the mobile device, as shown in Figure 4.15 (top of the activity) and Figure 4.16 (bottom of the activity). The first two preferences

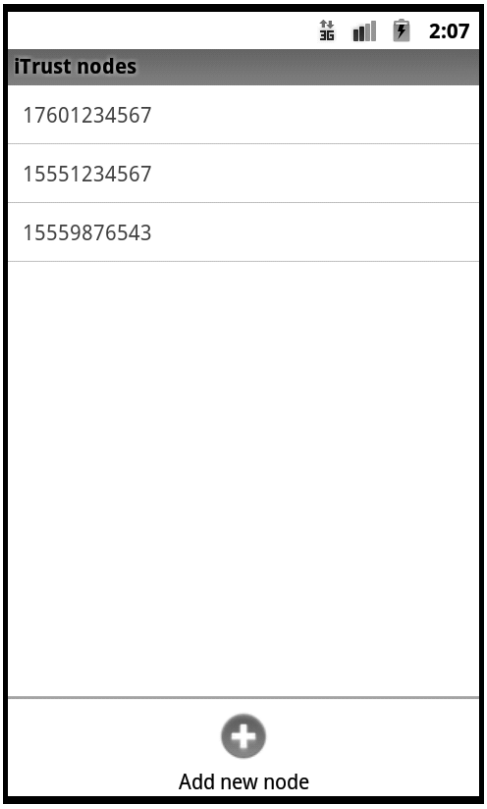



**Figure 4.14:** Screen to add new nodes to the local iTrust membership.

categories, General Settings and SMS Settings, are shown in Figure 4.15, and are discussed below.

#### 4.6.5.1 General Settings

This preference category enables a user to configure various options that directly affect the iTrust over SMS search and retrieval service on the mobile device or local node.

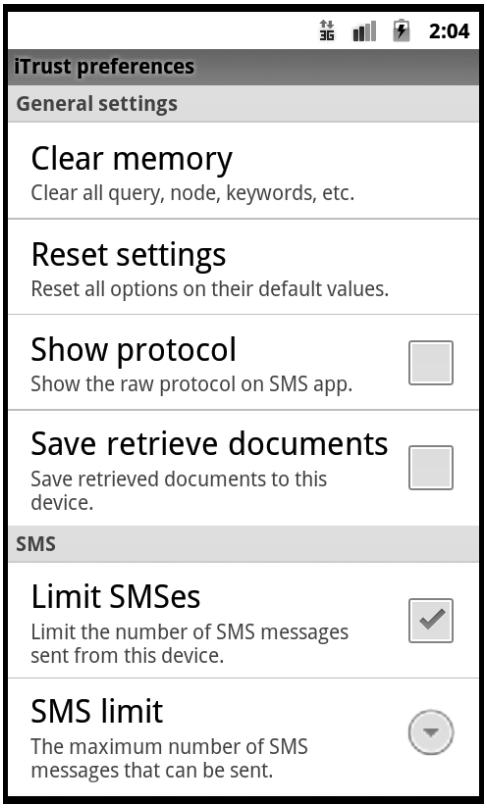



**Figure 4.15:** Screen that configures local iTrust preferences (top half).

The *Clear memory* preference deletes all information on the local node including node addresses, saved documents, saved searches, metadata and any other information generated by iTrust over SMS and stored on the local device. Note that this action is applicable only to documents stored on the local node; if another node already fetched a document from the local node, the fetched copy is not deleted.

Tapping the *Reset settings* preference restores all preferences to their default state to what they were when the application was first installed. No searches,

fetched documents, node lists, *etc.* are deleted or altered in any way. The *Reset settings* preference is a subset of the *Clear memory* preference that does *not* alter any shareable or iTrust over SMS information.

The *Show protocol* check box toggles the ability to show the underlying iTrust over SMS messaging protocol inside the Android Instant Messaging application. By default, the protocol is not shown to the Instant Messaging application, but the user may enable this option (*e.g.*, for debugging).

Also, by default, the *Save retrieve documents* check box disables the option of saving, to local storage, each document retrieved by the iTrust over SMS retrieval service. Toggling the preference saves each retrieved document into a predefined location; the user may then review the document offline or if the source node is no longer available.

#### 4.6.5.2 SMS Settings

This preference category restricts the iTrust over SMS service on the SMS telephony service. Because many mobile service providers charge a fee for each SMS message sent or received by a mobile device, this category allows a user to control his/her data usage fees more effectively.

The *Limit SMSes* check box allows a user to enable or disable the SMS messages transmitted from the mobile device (there is no realistic user control for restricting incoming SMS messages that does not rely on the mobile service provider

to some degree). If this preference is enabled, the preference *SMS limit* can be tapped and a dialog box pops up requesting the maximum number of SMS messages that may be sent by the iTrust over SMS service on this node. If the *Limit SMSes* preference is disabled, the *SMS limit* preference is likewise disabled and the maximum number of SMS messages sent is ignored.

### 4.6.6 Preferences (bottom)

Figure 4.16 displays the *Documents* category in the preferences activity. This category deals with the metadata distribution service of iTrust over SMS; configuration of metadata on the local device is managed in this category.

#### 4.6.6.1 Document Settings

The *Share contacts* check box disables the creation of metadata (to be shared with other nodes) from information stored in the local device's contact list. For example, if a user in the iTrust over SMS network had *John Smith* in his/her contact list and if this preference is enabled, then another user searching for information about John Smith (as in Figure 4.13) would have an encounter or match. By default, this preference is enabled.

Likewise, the *Share documents* check box allows a user to share metadata and documents with any node that sends a query to the local node. This preference has dual functionality. Enabling the option shares both metadata about a

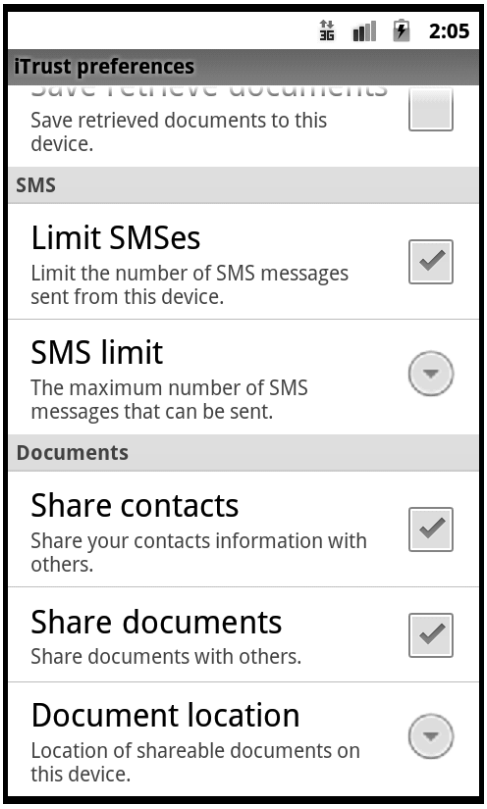



**Figure 4.16:** Screen that configures local iTrust preferences (bottom half).

document (during distribution) and the document itself (during retrieval). Similarly, disabling this option disables both the sharing of metadata and the related document.

The *Document location* preference, when tapped, pops up a dialog box asking for the location where the shareable documents are kept in local storage. Disabling the *Share documents* preference also disables the *Document location* preference.

## 4.7 Use Cases

The use cases for iTrust over SMS extend those for iTrust over HTTP in Section 3.2 and [55], but are adapted for typical mobile phone users. Although mobile phones are increasing in computational power and the ability to display more information on the screen, they are far smaller than laptops or desktops and, as such, necessitate a smaller simpler interface. For example, even though mobile users can perform Google searches on their mobile phones, they rarely venture past the first match, whereas desktop users commonly view second, third, or more matches. In the use cases below, we explain the use case context, and analyze how the iTrust over SMS service responds or adapts to requests.

### 4.7.1 Sporadic Searcher

We define a sporadic searcher to be a user who only occasionally uses the iTrust over SMS search and retrieval service; searches are relatively infrequent and retrieved documents are typically small. Such searchers are not likely to have many documents to distribute to other nodes, and the documents are not likely to be large. An example of a sporadic searcher might be someone who has no data service plan or who primarily makes only telephone calls on his/her mobile device.

The sporadic searcher mainly interacts with the screen in Figure 4.11 to view active searches, and occasionally interacts with the screen in Figure 4.13 to retrieve

documents. Searches (Figure 4.12) are infrequent, and other activities (Figures 4.14, 4.15, 4.16) are rarely used. The iTrust over SMS service accommodates the sporadic searcher, and defaults to Figure 4.11 when the application starts but, otherwise, does not adapt to the user. Specifically, it does not attempt to increase the node's membership by sending messages with node addresses. Because of the distributed nature of iTrust over SMS, it is difficult to decide, from a single node's perspective, whether its membership is sufficiently large.

The sporadic searcher is differentiated mostly by the need to address the *bootstrapping* problem when there are relatively few nodes in the local membership, but also by the relative lack of information or documents held by the sporadic searcher. Early social networking services also suffered from the bootstrapping problem. Social networks have limited value if only a few of the user's friends participate in the network; most centralized social networks require manual addition of friends, or suggest friends based on personal information.

In iTrust over SMS, the user can manually add nodes to the local membership via the user interface; however, more likely, the iTrust over SMS library automatically adds nodes to the local membership (it does not merely suggest that they be added), if those nodes are not already in the local membership. A common way of building a node's membership is that the iTrust library adds a matching node and a source node to the membership of a requesting node (searcher), it adds a requesting node to the membership of a node that receives the request and

it adds a retrieving node to the membership of a source node. This design choice increases a node's membership, by adding nodes that hold documents that match the user's search criteria and that provide interesting information from the user's perspective. Moreover, it allows sporadic searchers to *auto* promote themselves to casual searchers by simply searching more often and, thus, increasing their memberships.

### 4.7.2 Casual Searcher

A casual searcher is a user who uses the iTrust over SMS search and retrieval service to share information at a moderate frequency, size, and variety of shared documents. The casual searcher has a moderate number of documents stored on his/her mobile device, such as e-mail messages, contact information, personal photographs or videos, music and other documents. The amount of personal information stored correlates well with the usage of the device by typical smart phone users. For example, most smart phones have a basic built-in digital camera, which the smart phone user uses to take personal photographs when convenient; in contrast, a photographic enthusiast takes many more pictures but with a better-quality, stand-alone digital camera. Likewise, the typical smart phone user might store text documents or e-books but not literature manuscripts, home or amateur videos but not professional videos, e-mail messages but not work documents, *etc.*

The casual searcher mainly interacts with the screens in Figures 4.11, 4.12 and 4.13 to search for and retrieve documents. Sharing documents is handled automatically by iTrust over SMS, but the casual user may configure node settings using the screens in Figures 4.15 and 4.16. Like the sporadic searcher, the casual searcher is accommodated by iTrust over SMS by first showing the default activity in Figure 4.11 when the application starts.

Because the casual searcher sends queries frequently, the membership can be moderately large due to adding the matching node and the source node to the searcher's membership, adding the searcher and the relaying node to the matching node's membership and adding the searcher to the source node's membership. Frequent searches make the casual searcher relatively well-known among other nodes in the iTrust over SMS network.

Increasing a node's membership requires an increase in the number of nodes to which the metadata and the requests are distributed in order to maintain the same number of responses to a search query. An adaptive method [16] that we have developed for iTrust over HTTP can also be used for iTrust over SMS. It increases dynamically, and strategically, the proportion of *queried* nodes in the node's membership (rather than the total number of nodes in the node's membership). It uses an algorithm that detects whether the number of matches corresponds to an analytically expected number of matches.

Increasing one's own membership and increasing one's presence in other nodes' memberships can improve access to information as well as the speed with which matches are made. Doing both provides a kind of "instant gratification," which is desirable for the mobile user demographic. Thus, by making more searches and increasing their memberships, casual searchers can *auto* promote themselves to become avid searchers.

### 4.7.3 Avid Searcher

An avid searcher has a plethora or abundance of shareable (and likely very desirable) information, and has or seeks hours of music or video and entire collections of shareable documents. At present, this behavior transcends the typical smart phone user; therefore, the avid searcher population is smaller than the casual searcher population.

However, a crucial difference between the avid searcher and the casual searcher is that the avid searcher typically retrieves not only the document for the first match but also the documents for the second, third or more matches. Because an avid searcher is likely to retrieve all documents for which the metadata match, the order of the match responses is less important than that for the casual searcher for which the first match response is the most important.

As for the previous types of searchers, the screen in Figure 4.11 serves as the default activity when the application starts. However, for the Avid searcher, the

*Search details* activity shown in Figure 4.13 is typically used more often than the *New search* activity shown in Figure 4.12. The remaining activities shown in Figures 4.15, 4.16 and 4.14 are still seldom used.

Importantly, the avid searcher becomes more and more *instantly gratified* as the match responses return ever faster; however, there is a physical limit to the speed of SMS (which is determined by the specific mobile service provider). Repeatedly reaching this limit might have the effect of pushing the avid searcher behavior back down to that of the casual searcher and, indeed, might create a churn of casual searchers entering and leaving the avid searcher status.

### 4.7.4 Pure Searcher

The pure searcher is any searcher who searches for and retrieves documents but, unlike the other searchers previously discussed, does not contribute (distribute) documents to other nodes in the iTrust over SMS network.

The pure searcher does not distribute documents by not storing documents locally, ignoring search queries, or ignoring retrieval requests. Not storing documents or ignoring search queries is made possible using the preferences shown in the screens in Figures 4.15 and 4.16. Such preferences are optional, because there might be legitimate reasons not to share local documents with others (political oppression, copyright laws, *etc.*).

The pure searcher can still distribute metadata on shareable information and, thus, send its node address to other nodes for inclusion in their memberships. Consequently, iTrust over SMS works as intended, until the final step when a searcher attempts to retrieve the document, at which point the source node simply ignores the retrieval request. There are no preferences to enable this behavior and, indeed, iTrust over SMS does *not* support this option. To achieve this behavior, one would have to modify the iTrust over SMS source code and create a mimic iTrust over SMS service.

By *not* sharing documents, the pure searcher is not sending its node address to other nodes during query relaying, match reporting, or document sharing and, thus, it pays the price of having a smaller chance of being included in other nodes' memberships. This membership penalty might encourage the pure searcher to distribute shareable documents and become a sporadic or casual searcher.

The pure searcher in iTrust over SMS is similar to leechers in peer-to-peer networks, such as Gnutella, that provide little or no benefit to the community. Leechers are discouraged but are sometimes unavoidable, particularly when there are new nodes with small memberships or nodes that hold only a few documents locally (such as sporadic searchers).

### 4.7.5 Other Use Cases

Some nodes might freely distribute local documents and never search for documents; such behavior mostly occurs because of an abundance of resources or general good-will. Other nodes might simply relay queries, allowing the built-up membership and search queries to be used for other purposes either benign, nefarious, or somewhere in between. Lastly, malicious nodes might actively or passively attack other nodes, again not necessarily by any direct user action.

## 4.8 Performance Evaluation

To evaluate iTrust over SMS, we consider the probability of a match, and also the number of messages required to achieve a match, using both analysis and emulation based on our iTrust over SMS implementation. We assume that all of the participating nodes in the iTrust over SMS network have the same membership. Moreover, we assume that communication is reliable and timely, and that all of the participating nodes have enough memory to store the source files and the metadata that the nodes generate and receive. Furthermore, we assume that the metadata and requests are sent directly to the nodes without relaying, and that the nodes do not delay in requesting metadata or reporting matches.

The parameters determining the performance of the iTrust over SMS system are:

- $n$: The number of participating nodes (*i.e.*, the size of the membership set)
- $x$: The proportion of the $n$ participating nodes that are operational (*i.e.*, $1 - x$ is the proportion of non-operational nodes)
- $m$: The number of participating nodes to which the metadata are distributed
- $r$: The number of participating nodes to which the requests are distributed
- $k$: The number of participating nodes that report matches to a requesting node.

Note the similarity to the performance evaluation of iTrust over HTTP discussed in Section 3.3. The addition of $k$ to keep track of the number of participating nodes that report matches becomes more important for iTrust over SMS, compared to iTrust over HTTP, because message cost is much more important for the SMS network than for the HTTP network. A network-constrained user will more selectively chose which resource to fetch for the conservative SMS network; so we take into account $k$ for the iTrust with/over SMS performance evaluation.

### 4.8.1 Probability of a Match

First, we consider the probability that, for a given request, a match (encounter) occurs, *i.e.*, that one or more nodes have a match for that request. The results for

the probability of a match hold for iTrust over HTTP, iTrust with SMS (including the SMS-HTTP bridge node) and iTrust over SMS. However, we must redo the analysis of Section 3.3 to include $k$.

#### 4.8.1.1 Analysis

The probability of *exactly* $k$ matches follows the hypergeometric distribution with parameters $n$, $x$, $m$ and $r$, and is given by:

$$\begin{aligned} P(k) &= \frac{\binom{mx}{k}\binom{n-mx}{r-k}}{\binom{n}{r}} \\ &= \frac{(\frac{mx}{k}\frac{mx-1}{k-1}\cdots\frac{mx-k+1}{1})(\frac{n-mx}{r-k}\frac{n-mx-1}{r-k-1}\cdots\frac{n-mx-r+k+1}{1})}{(\frac{n}{r}\frac{n-1}{r-1}\cdots\frac{n-r+1}{1})} \end{aligned} \tag{4.1}$$

for $mx + r \leq n$ and $k \leq \min\{mx, r\}$.

In particular, the probability of $k = 0$ matches is given by:

$$P(0) = \frac{(\frac{n-mx}{r}\frac{n-mx-1}{r-1}\cdots\frac{n-mx-r+1}{1})}{(\frac{n}{r}\frac{n-1}{r-1}\cdots\frac{n-r+1}{1})} \tag{4.2}$$

for $mx + r \leq n$.

Consequently, the probability of a match (*i.e.*, *one or more* matches) is given by:

$$P(k \geq 1) = 1 - \frac{(\frac{n-mx}{r}\frac{n-mx-1}{r-1}\cdots\frac{n-mx-r+1}{1})}{(\frac{n}{r}\frac{n-1}{r-1}\cdots\frac{n-r+1}{1})} \tag{4.3}$$

for $mx + r \leq n$. If $mx + r > n$, then $P(k \geq 1) = 1$.

Figures 4.17, 4.18 and 4.19 show the probability of a match obtained from Equation (4.3) with $n = 250$ nodes where $x = 100\%$, $80\%$ and $60\%$ of the participating nodes are operational, respectively, as a function of $m = r$. As we see

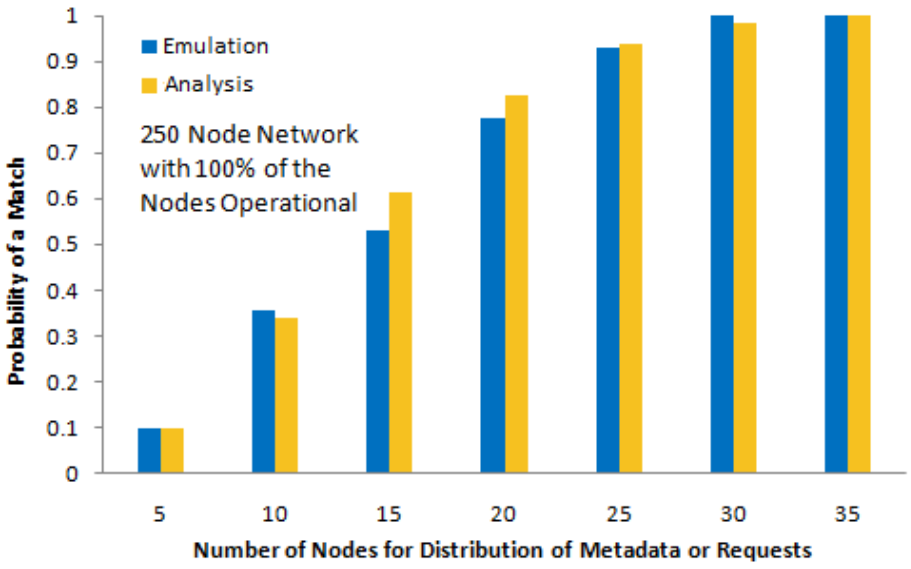



**Figure 4.17:** Match probability vs. number of nodes for distribution of metadata or requests in an iTrust network with 250 nodes where 100% of the nodes are operational.

from the graphs, the probability of a match increases and approaches 1, as $m = r$ increases.

#### 4.8.1.2 Emulation

Using our implementation of iTrust, we performed experiments to validate the analytical results for the probability of a match obtained from Equation (4.3).

Before we ran our emulation program, we deleted all resources and data from the node. Next, the program adds the nodes to the membership. Then, we supply the number $n$ of nodes for distribution of metadata and requests, and the proportion $x$ of operational nodes, to the emulation program. Next, we call the source nodes to upload the source files and the program then creates the corresponding

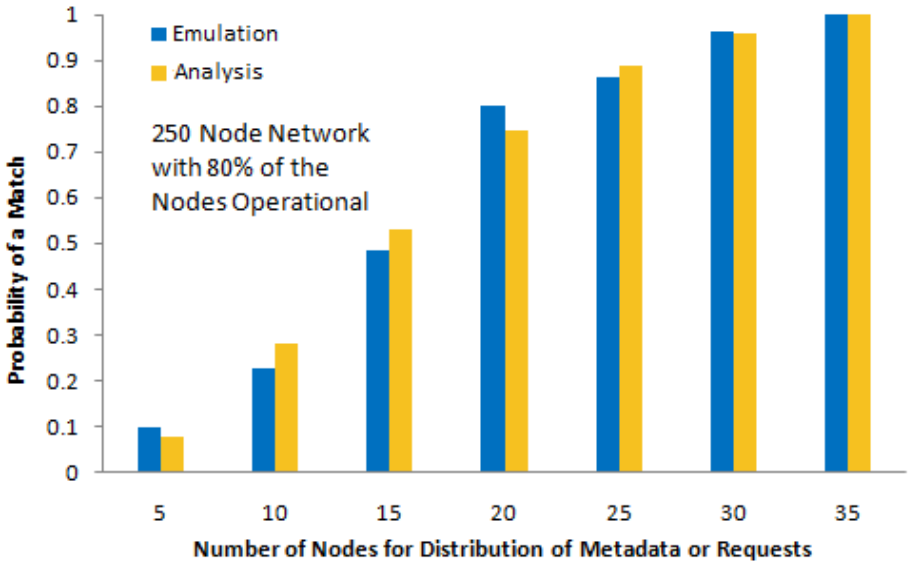



**Figure 4.18:** Match probability vs. number of nodes for distribution of metadata or requests in an iTrust network with 250 nodes where 80% of the nodes are operational.

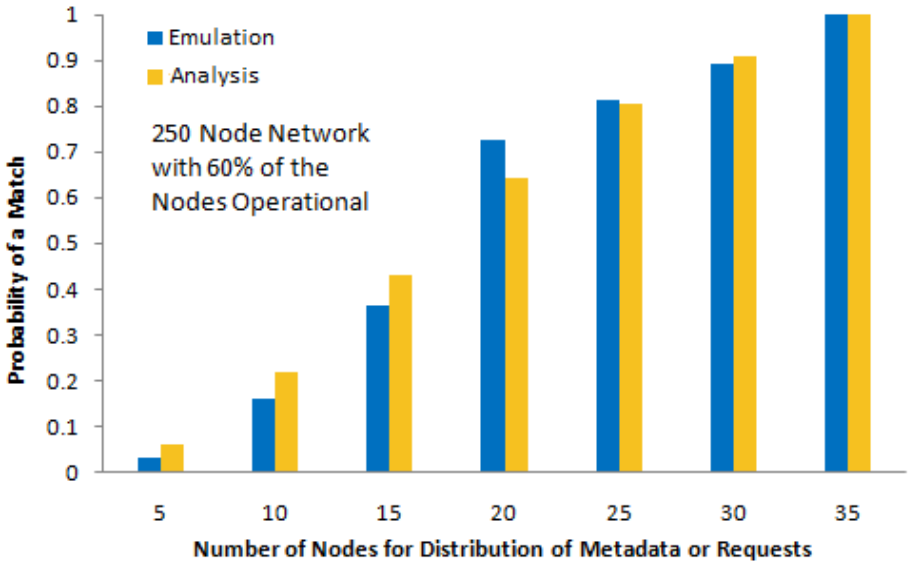



**Figure 4.19:** Match probability vs. number of nodes for distribution of metadata or requests in an iTrust network with 250 nodes where 60% of the nodes are operational.

metadata. Then, the program randomly selects $m$ nodes for metadata distribution and distributes the metadata to those nodes. Next, the program randomly selects $r$ nodes for request distribution and distributes the requests to those nodes. If one or more nodes returns a response, there is a match and the emulation program returns 1; otherwise, there is no match and the emulation program returns 0.

We repeated the same process 100 times for the source nodes and correspondingly for the requesting nodes, and plot the mean results in our graphs for the emulation.

Figures 4.17, 4.18 and 4.19 show the emulation results with 250 nodes where 100%, 80% and 60% of the participating nodes are operational, respectively, as a function of $m = r$. As we see from these graphs, the emulation results are very close to the analytical results calculated from Equation (4.3). As these results indicate, iTrust retains significant utility even in the case where a substantial proportion of the nodes are non-operational.

### 4.8.2 Number of Messages to Achieve a Match

Next, we consider the number of messages required to achieve a match for a given request.

#### 4.8.2.1 Analysis

For iTrust over HTTP and iTrust over SMS, the mean number $Y$ of messages required to achieve a match is given by:

$$Y = r + \sum_{k=1}^{\min\{mx,r\}} kP(k) \tag{4.4}$$

The term $r$ on the right side of Equation (4.4) represents $r$ requests from the requesting node to other participating nodes. The sum represents the number $k$ of matches (response messages), weighted by the probability $P(k)$ of $k$ matches obtained from Equation (4.1).

For iTrust with SMS (including the SMS-HTTP bridge node), the mean number $Y$ of messages required to achieve a match is given by:

$$Y = 2 + r + \sum_{k=1}^{\min\{mx,r\}} kP(k) \tag{4.5}$$

The term 2 on the right side of Equation (4.5) represents: 1 request message from the mobile phone to the iTrust SMS-HTTP bridge node and 1 match response message from the iTrust SMS-HTTP bridge node to the mobile phone. The term $r$ on the right side of the equation represents $r$ requests from the iTrust SMS-HTTP bridge node to iTrust over HTTP nodes. The sum is the same as that in Equation (4.4) and represents messages sent from the matching iTrust over HTTP nodes to the iTrust SMS-HTTP bridge node. Note that only the two messages involve the more expensive communication over the cellular network.

Figures 4.20, 4.21 and 4.22 show the number of messages obtained from Equations (4.1) and (4.5) with $n = 250$ nodes where $x = 100\%$, 80% and 60% of the participating nodes are operational, respectively, as a function of $m = r$. As we see from the graphs, the number of required messages increases as the probability of a match increases (and as $m = r$ increases), but is bounded by $2 + 2r$ because, in Equation (4.5), $\sum_{k=1}^{\min\{mx,r\}} kP(k) \leq \sum_{k=1}^{r} kP(k) \leq r \sum_{k=1}^{r} P(k) \leq r$.

#### 4.8.2.2 Emulation

Using our implementation of iTrust over SMS, we performed experiments to validate the analytical results for the number of messages to achieve a match obtained from Equations (4.1) and (4.5). The experiments were performed as described previously in Section 4.8.1.2.

Figures 4.20, 4.21 and 4.22 show the emulation results with 250 nodes where 100%, 80% and 60% of the participating nodes are operational, respectively, as a function of $m = r$. As we see from these graphs, the emulation results are very close to the analytical results calculated from Equations (4.1) and (4.5).

Figures 4.17, 4.18 and 4.19 and Figures 4.20, 4.21 and 4.22 show the benefit-cost trade-offs between the probability of achieving a match and the number of messages required to achieve a match. Note that the number of messages required to achieve a match is much greater than for centralized search engines, but is much less than for flooding strategies.

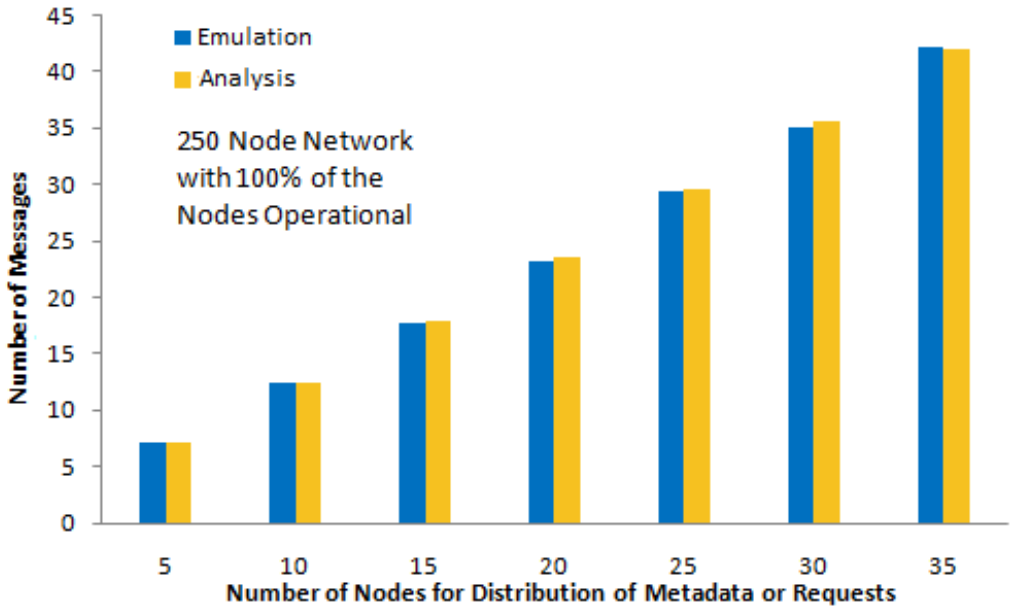



**Figure 4.20:** Number of messages vs. number of nodes for distribution of metadata or requests in an iTrust with SMS network (including the SMS-HTTP bridge node) with 250 nodes where 100% of the nodes are operational.

### 4.8.3 Analysis of a Medium-Size Membership

The membership size used in our analysis must correspond to real-world characteristics. Notably, from [79], we observe that the average Twitter follower size (roughly the same as the iTrust membership size) is slightly more than 100 nodes. Micro-blogging is a good application for iTrust over SMS, as micro-blogging is ideally suited to the small text-based messages of SMS. For our analysis, we chose a membership size of $n = 144$ nodes, with $m = r = 24$ metadata/request messages, for reasons that will become clear in Section 4.8.5.

For an iTrust network with $n = 144$ nodes and $x = 1.0, 0.7, 0.4, 0.2$ available nodes, Figure 4.23 shows the probabilities of one or more matches, obtained from

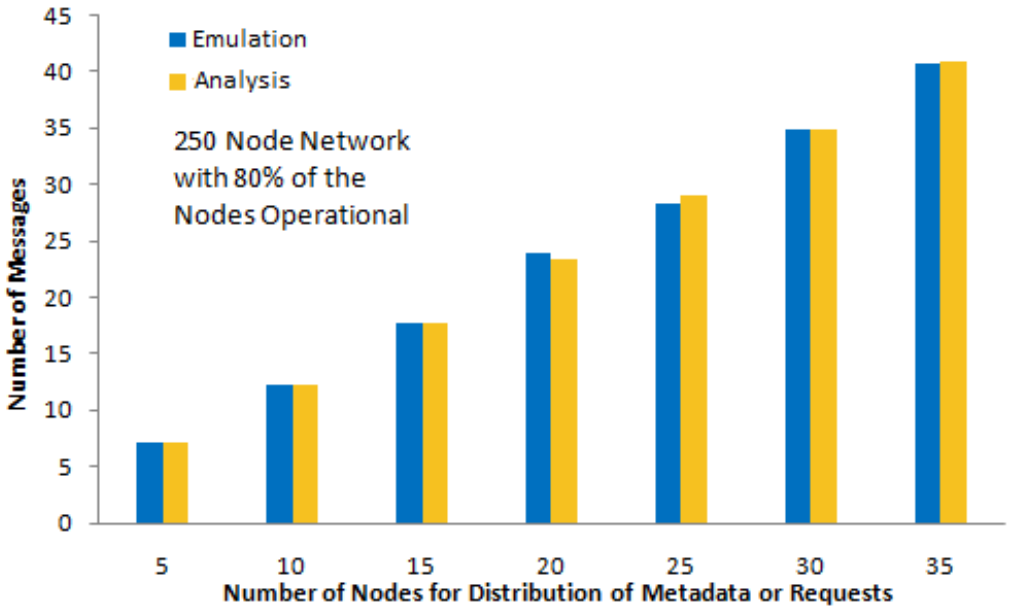



**Figure 4.21:** Number of messages vs. number of nodes for distribution of metadata or requests in an iTrust with SMS network (including the SMS-HTTP bridge node) with 250 nodes where 80% of the nodes are operational.

Equation (4.3), as the number of nodes to which the metadata and the requests are distributed increases.

### 4.8.4 Emulation of a Small-Size Membership

The iTrust over SMS system was developed and deployed on a small number of Android mobile phones, and was tested for fitness and robustness on those mobile phones. Unfortunately, it is not economically feasible to purchase enough physical mobile phones and accompanying data/service plans to enable a real-world test of 144 nodes. The next best choice is the deployment of iTrust over SMS on an emulated system of networked physical devices. We ran multiple instances of the

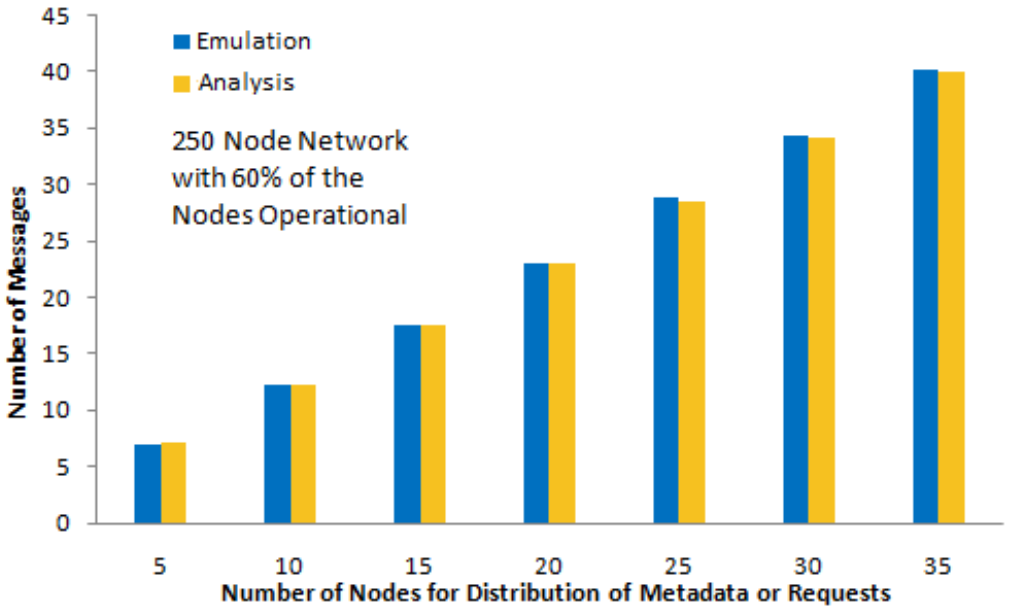



**Figure 4.22:** Number of messages vs. number of nodes for distribution of metadata or requests in an iTrust with SMS network (including the SMS-HTTP bridge node) with 250 nodes where 60% of the nodes are operational.

Android operating system on an emulated ARM to x86 environment using UNIX sockets for SMS communication. Multiple instances of the standard Android emulator with Android 2.3 (Gingerbread) images were run up to the limit of allowable ports; specifically, the maximum number of Android emulators is locked at 16 (32 unidirectional ports with each emulator requiring two ports for sending and receiving SMS messages). Furthermore, the standard Android operating system has a built-in SMS sending rate maximum, which limits the number of messages that may be sent in a given time period; those limits were disabled for testing purposes.

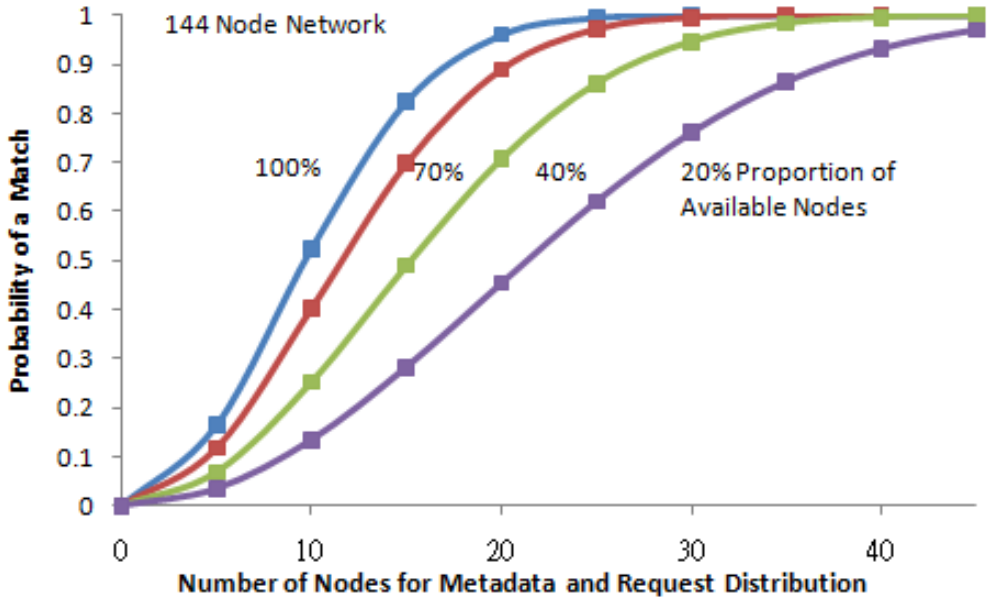



**Figure 4.23:** Probabilities $P(k \geq 1)$ of a match as the number $m = r$ of nodes to which the metadata and the requests are distributed increases, for different proportions $x$ of available nodes.

The emulation experiment was run on an AMD Phenom II 3.4GHz quad core hyper-threaded testbed; each trial (involving 16 nodes) took 16GB of RAM and required 65 seconds to complete. The experiments involved an iTrust network with $n = 16$ nodes, $x = 1.0$ proportion of available nodes and $m = r = 8$ metadata/request messages.

Figure 4.24 shows the observed results for 1000 trial runs. Each trial run consists of generating random resource and keyword pairs, relaying the resulting metadata, searching for at least one of the keywords and finally counting the number of encounters (matches). In the figure, we see that the observed data

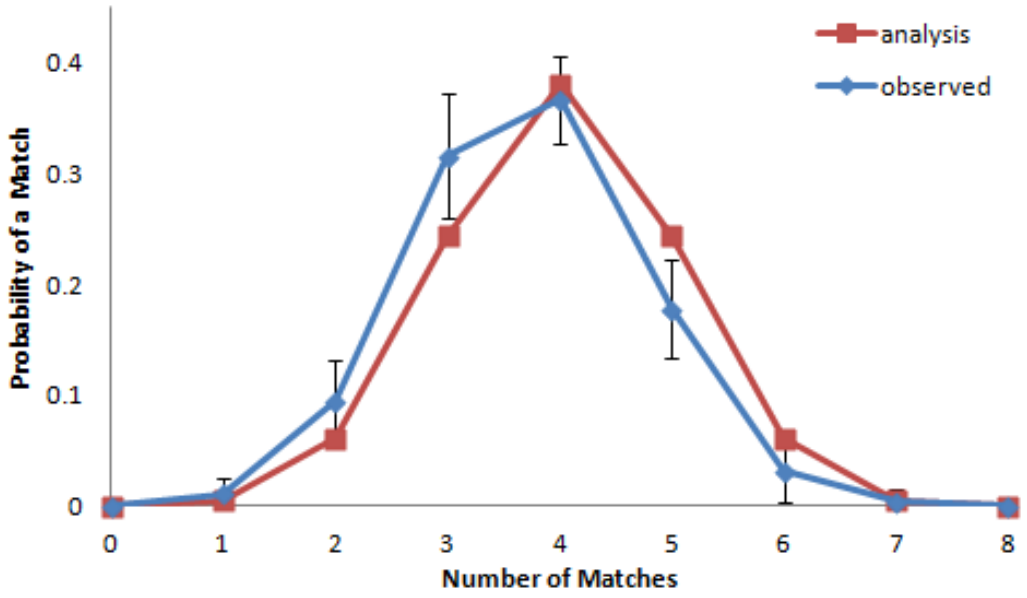



**Figure 4.24:** The number $k$ of matches vs. the mean probabilities $P_{observed}(k)$ with error bars, for a small Android emulator testbed. The probabilities $P_{analysis}(k)$ are also shown.

from the Android application closely follows the analytical data obtained from Equation (4.1) for the probability $P(k)$ of $k$ matches.

Table 4.3 lists the analysis and the observed probabilities for the various numbers of matches. For example, for $k = 6$ matches $P_{analysis}(6) = 0.060926$ and $P_{observed}(6) = 0.031020$. The right-most column lists the cumulative observed probabilities for the various numbers of matches. For example, the cumulative observed probability for 1 to 6 matches is $P_{cumulative}(1 \leq k \leq 6) = 0.996000$.

| Matches | Analysis | Observed | Cumulative Observed |
|---|---|---|---|
| 0 | 0.000155 | 0.000000 | 0.000000 |
| 1 | 0.004974 | 0.011000 | 0.011000 |
| 2 | 0.060926 | 0.094143 | 0.105143 |
| 3 | 0.243705 | 0.315714 | 0.420857 |
| 4 | 0.380790 | 0.366796 | 0.787653 |
| 5 | 0.243705 | 0.177327 | 0.964980 |
| 6 | 0.060926 | 0.031020 | 0.996000 |
| 7 | 0.004974 | 0.004000 | 1.000000 |
| 8 | 0.000000 | 0.000000 | 1.000000 |

**Table 4.3:** The analysis, observed and cumulative observed probabilities.

### 4.8.5 The Importance of $2 * \sqrt{n}$

In the above experiments, we used $n = 16$ nodes with $m = r = 8 = 2 * \sqrt{16}$ nodes to which the metadata and the requests are distributed. This choice was deliberate.

In Section 2.4 and [65], we showed that, if the iTrust membership set contains $n$ participating nodes of which a proportion $x$ are operational, the metadata are delivered to $m$ participating nodes, a request is delivered to $r$ participating nodes, then the probability of a match satisfies:

$$P(k \geq 1) \quad > \quad 1 - e^{-\frac{mrx}{n}} \tag{4.6}$$

In particular, if $m = r = \lceil 2 * \sqrt{n} \rceil$ and $x = 1.0$, then

$$\begin{aligned} P(k \geq 1) &> 1 - e^{-\frac{\lceil 2*\sqrt{n} \rceil \lceil 2*\sqrt{n} \rceil}{n}} \\ &> 1 - e^{-\frac{2*\sqrt{n}\ 2*\sqrt{n}}{n}} \\ &= 1 - e^{-4} \\ &> 0.9817 \end{aligned} \tag{4.7}$$

Thus, to obtain a high probability of one or more matches, we choose $m = r = \lceil 2 * \sqrt{n} \rceil$ nodes to which to distribute the metadata and the requests.

### 4.8.6 Mean Search Latency

For the mobile phones on which iTrust over SMS is deployed, first we make a basic observation about the *mean search latency*, *i.e.*, the duration in time from sending a query to receiving the first match response at the querying node. The mean search latency is *twice* the SMS *delivery latency*, *i.e.*, the latency for sending the query and the latency for sending back the response. We ignore the insignificant time required to process an encounter on the matching node (in practice, this time is less than 100 milliseconds on a modern mobile handset). We also ignore any time required to retrieve the resource information because, in practice, the information varies greatly in size (from a short text snippet to

a video file that uses multi-part messages) and because the user may choose to retrieve the information for only a few of the matches or even none at all.

Because a large-scale quantitative study of SMS latency has already been performed multiple times throughout the almost 20 year history of SMS (see a relatively recent study in [66]), we focus here on a smaller qualitative study related specifically to iTrust over SMS.

First, the delivery latency (and consequently the mean search latency) is mostly constant among the service providers of each mobile phone, so long as the nodes use the same major service provider. Specifically, mobile devices within the same major service provider communicate relatively quickly (usually less than 10 seconds, but sometimes as little as 4 seconds). In particular, two T-Mobile phones had an mean search latency of under 10 seconds, two Sprint Nextel phones had similar results, as did two Verizon phones and two AT & T phones, which comprise the four major national service providers in the United States.

Second, the mean search latency between devices that use two *different* U.S. service providers is considerably *worse* than devices that use the same U.S. service provider. For example, one T-Mobile phone communicating with a non-T-Mobile phone had an mean search latency of seven minutes (about three and one-half minutes per SMS delivery latency); however, after repeated experiments between the same two nodes, the mean search latency reduced to slightly less than three minutes. Presumably, the SMSC communication between service providers is

not optimized or prioritized to handle out-of-network messages (although a more extensive study is required to determine the exact cause). Likewise, the out-of-network SMS messages seem to be delivered faster after the SMSCs establish some form of adaptive or smart routing.

Third, brand licensees (*i.e.*, secondary companies that partner with the four major U.S. service providers) have the *worst* mean search latency for out-of-network communication. For example, Virgin Mobile (the brand licensee) uses the Sprint Nextel network (one of the four U.S. networks) and consistently have the worst latency when a Virgin Mobile device is used with any non-Virgin Mobile device, at more than 15 minutes per search to receive a response to a match. The exact reason for this behavior is unknown.

In conclusion, the mean search latency varied widely across different service providers. However, the mean search latency is consistently less if all of the participating nodes use the same service provider and consistently more if the nodes use different service providers.

## 4.9 Summary

In this chapter, first we described iTrust with SMS, which enables a mobile phone user to utilize the cellular network to access the iTrust over HTTP network via the iTrust with SMS bridge node. The iTrust with SMS network can be accessed by any mobile device with any generic SMS chat application; in addition,

a rudimentary iTrust with SMS Android application is provided to aid in making queries and retrieving data.

Next, we described iTrust over SMS, which is a fully developed SMS implementation of iTrust, that enables users to contact each other via cell phones and to share information. A rudimentary iTrust over SMS Android application was presented as a starting point for an easy-to-use interface; later, the graphical user interface was enhanced with a fully developed Android application. The smart phone graphical user interface enabled exploration of several use cases, including sporadic, casual, avid and pure searchers. Finally, we presented a performance evaluation of iTrust over SMS including: match probability, number of messages required for a match, different network sizes, optimum message distribution and mean search latency.

# Chapter 5

# iTrust over Wi-Fi Direct

In our increasingly wireless and mobile world, the typical cell phone user has come to expect ubiquitous access to public and private information. The traditional information search engines, such as Google, Yahoo! and Bing, have transitioned from mouse pointer desktop computers to touch screen mobile devices; thus, searching for public information is relatively easy, and continues to improve. To a certain extent, private information can also be easily indexed and searched; *e.g.*, searching for pictures taken from a camera phone, from that *same* phone, is relatively straightforward even if the graphical user interface is difficult to use. However, in the same way that public Web sites enrich the user's search experience (*i.e.*, increase the number and quality of relevant search hits), private stores of information from which to search also enhance the user's search experience. The gap between searching numerous public Web sites and searching only an individual device has, until recently, not been addressed.

The recent resurgence of personal search includes examples such as Facebook using facial recognition to tag friends in uploaded pictures, or Google+ using *circles* to suggest products/advertisements to the user. In both cases, the approach of those companies has been to have all users or participants upload personal information to a central information repository and then have each user access the centralized repository to search for personal information of their friends (or circles in the case of Google+). As long as each user has no objection to submitting personal information to a third party, the centralized search approach works; the individual users (both first and second parties) benefit by having a third party perform the search functions for them, and the third party benefits from extracting and selling the personal information of the other two parties (*e.g.*, advertising and data mining).

The centralized search approach does not work well when the third party has no incentive to enable sharing of information, or even worse when the third party has an incentive to restrict or censor the sharing of information. Consider the two cases of politics and economics. We have seen, in the recent past and in the present, political upheavals in Egypt, Tunisia and Syria, where the government has no incentive to enable the sharing of information (such as pictures, video, protest information, *etc.*) among its citizens. In fact, it is in the best interest of the government to censor and restrict the dissemination of non-government sanctioned information. In the economic case, consider the example of many

buyers and sellers in a concentrated area, such as weekend shoppers in a shopping mall, swap meet or bazaar. It is in the best interest of each buyer to share information with other buyers by comparing the products and the prices of the sellers (perhaps through pictures or short text messages); it is in the best interest of each seller not to allow buyers to compare such information. In both cases, individuals benefit from sharing personal information; and the third party either has no incentive to enable sharing or has an incentive to restrict sharing.

To address the need for sharing of personal information and simultaneously to prevent a third party from censoring information or preventing the dissemination of information, we created iTrust over Wi-Fi Direct [58, 61]. The iTrust over Wi-Fi Direct system enables users with Wi-Fi Direct enabled mobile devices to publish, search for and retrieve information among themselves. Wi-Fi Direct is a relatively new wireless technology, based on IEEE 802.11, that enables devices to form a peer-to-peer network without the need for a third intermediary device, such as an access point. In the rest of this chapter, we describe the iTrust over Wi-Fi Direct API and components as implemented on the Android platform, the associated networking model, and the peer management strategies.

## 5.1 iTrust over Wi-Fi Direct API Components

The iTrust over SMS system enables decentralized publication, search and retrieval of information between mobile devices, as long as those devices are SMS

capable (data are transmitted over SMS). However, there are circumstances in which the centralized SMS store-and-forward model can be shut down by a third party; therefore, we developed iTrust over Wi-Fi Direct as the natural technological progression that gives iTrust a completely decentralized and robust method of information transfer.

The iTrust over Wi-Fi Direct system is implemented in Android, and is compatible with version 4.1 (and above) of the mobile platform. The choice of Android was made for a variety of reasons including previous experience and hardware compatibility. Moreover, at the time of this writing, Android is the only mobile platform that has hardware support for Wi-Fi Direct; neither Apple's iOS nor Microsoft's Windows Phone supports Wi-Fi Direct. It remains to be seen whether the new Firefox operating system (also known as B2G) will include support for Wi-Fi Direct.

Figure 5.1 illustrates the iTrust over Wi-Fi Direct API and components, their relationship to both user/programmer applications and the underlying Android/ Linux mobile platform. Below we discuss in detail each block in the diagram and their relationships and interactions; the center iTrust over Wi-Fi Direct blocks are the most pertinent, but the two surrounding user and operating system blocks are also discussed. The previous name of Wi-Fi Direct is Wi-Fi P2P; we use both terms interchangeably in the rest of this dissertation.

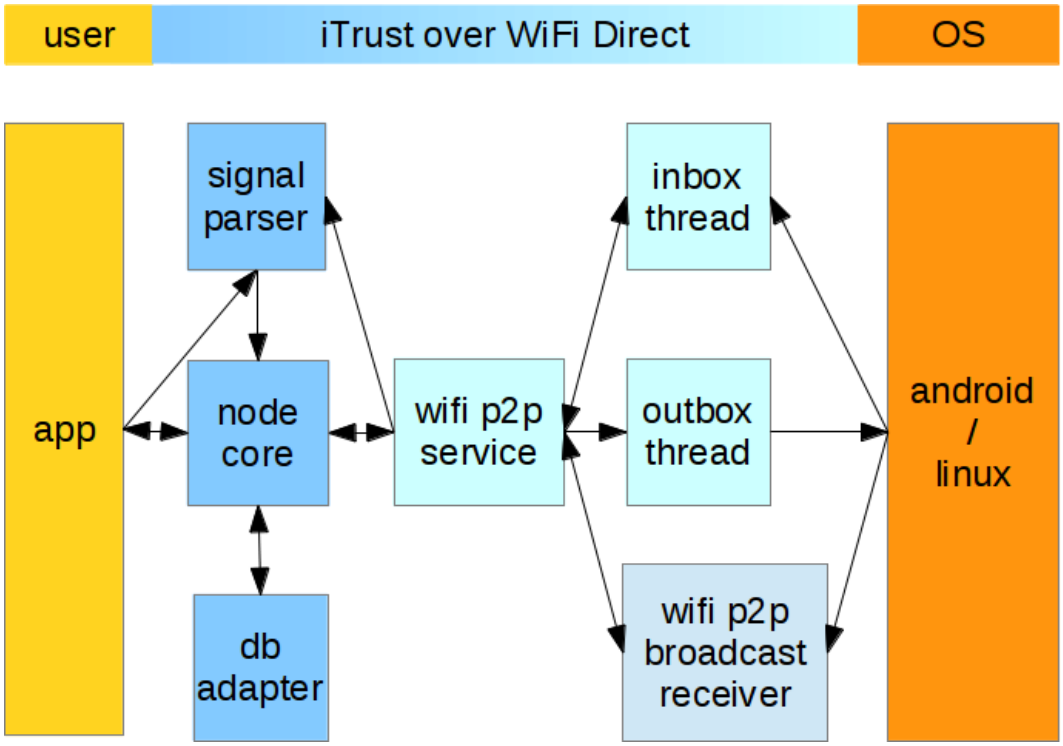



**Figure 5.1:** The iTrust over Wi-Fi Direct API and components.

### 5.1.1 Application

The app block at the left of Figure 5.1 is not strictly part of iTrust over Wi-Fi Direct but is, instead, a placeholder for the user and program code that interfaces with iTrust over Wi-Fi Direct. In the previously implemented versions of iTrust, namely iTrust over SMS and to a much lesser extent iTrust over HTTP, the application had only minimal interaction with the components of iTrust, which remains true in iTrust over Wi-Fi Direct. The application has access to only the signal parser and the node core, in order to decouple the logic between the application and the components; doing so creates a clear separation of tasks and

ensures that any application can easily add iTrust network functionality to existing modes of communication.

For example, a simple instant message text chat application was written for iTrust over SMS, described in Section 4.5, in order to demonstrate the easy way in which any application can add the decentralized publication, search and retrieval functionality of iTrust. Any application written for iTrust over SMS can use iTrust over Wi-Fi Direct without any additional function calls. The increased transmission rate available to Wi-Fi Direct, compared to SMS, enables a broader range of applications, including: file transfer, picture or media gallery sharing with nearby devices, music sharing, text document collaboration, *etc.*

### 5.1.2 Signal Parser

The signal parser in iTrust over Wi-Fi Direct is similar to the signal parser in iTrust over SMS, described previously in Section 4.4.1, but with several enhancements. The most extensive enhancement is the peer management protocol required to enable mobile ad-hoc functionality in the mobile device. Peer management is essential to ensuring that a moving device, or peer, can remain connected to other nearby devices and maintain the network connection required for publishing, searching for and retrieving information. The peer management code is quite extensive and is described in Section 5.3; however, it is important to note that the signal parser is responsible for reading the peer management message types and

properly informing the node core of which actions to take regarding peer connection states. Apart from these additions, the signal parser retains the iTrust over SMS tasks for decoding incoming messages and appropriately informing the node core of which actions to take.

### 5.1.3 Node Core

The node core is an integral part of iTrust over Wi-Fi Direct, and handles all program accounting and system state information. It retains the functionality found in the iTrust over SMS node core, described previously in Section 4.4.1, but adds a substantial amount of Wi-Fi P2P adapter information that is required for proper operation such as: self peer state, Wi-Fi P2P adapter states and nearby peer states. For example, a peer must extract its own MAC address from the Android platform to self identify to other peers when joining the iTrust membership (this functionality is explained in Section 5.2). Also, the Wi-Fi P2P hardware adapter state is *separate* from the normal Wi-Fi hardware adapter state, *i.e.*, Android makes a distinction between the normal Wi-Fi connectivity to an access point and P2P Wi-Fi connectivity directly to another peer. The two adapters must be managed separately, and the node core performs these functions, with the help of the Wi-Fi P2P service component described in Section 5.1.5.

The node core serves the fundamental iTrust functions of: node management (not P2P related), metadata generation (keyword creation) and distribution

(JSON import/export), query distribution and matching (encounters), message relaying and message formatting (protocol finite state machine and logic control). Furthermore, the node core is the only component that interacts with the database.

### 5.1.4 Database Adapter

The database (DB) adapter in iTrust over Wi-Fi Direct is structurally similar to that in iTrust over SMS, described previously in Section 4.4.1, and is not discussed here in detail. The singular enhancement is the enlargement of the node table (the database table that holds information on other nodes/peers in the iTrust membership). Specifically, whereas a particular peer is assumed to be always connected over SMS (because it is a store-and-forward bearer of information), the mobile ad-hoc nature of Wi-Fi Direct does not allow this same assumption to be made. The node table has three vital additions: a name identifier to identify the peer, a Boolean field to specify whether the peer is *in range* (physically within radio distance) and a Boolean field to store the connection state information of the peer. The database adapter simply stores this information, and has no logic to process the information. The node core and the Wi-Fi P2P service component act on the peer information.

### 5.1.5 Wi-Fi P2P Service Component

The Wi-Fi P2P service component is the centerpiece of iTrust over Wi-Fi Direct and effectively merges the fundamental iTrust network logic with the twin goals of: sending and receiving messages, and handling Wi-Fi Direct network connections.

This component is a daemon that exists separately from the application (and the other components) and that services incoming and outgoing messages through separate threads. Specifically, in Android terminology, the Wi-Fi P2P service component is a *started* Service object that is invoked with an Intent object near the beginning of application execution; once created, it remains active indefinitely (until the device is powered off). Because of the relatively aggressive memory management in Android, the Wi-Fi P2P service component may be torn down by the Android memory manager if another application requires more temporary memory; in this case, it is automatically restarted when more memory becomes available (in practice, a delay of one to two seconds).

At start-up time, the Wi-Fi P2P service component creates an Inbox thread to listen for incoming messages (described in Section 5.1.7) and acts as an intermediary between the Inbox thread and the signal parser (all of the incoming messages must be parsed by the signal parser). When the node core needs to send a message, the Wi-Fi P2P service component creates an Outbox thread (described in Section 5.1.8).

The handling of network connections is the other important function of the Wi-Fi P2P service component; indeed, it is responsible for starting and maintaining all Wi-Fi Direct functionality. It is necessary to outline the primary steps required to transmit information between peers.

First, the Wi-Fi P2P service component must check the device system settings and request permission to control the Wi-Fi P2P hardware adapter; assuming that permission is granted, the device immediately announces itself to the network and begins searching for peers. Second, new peers are detected and a connection is attempted, assuming that the peer is deemed available for connectivity (in Android parlance, the `onPeersAvailable` interface is implemented). At this point, the node core is informed of the peer changes, and all peer database entries are updated to reflect the current in-range status. Third, assuming that the invited peer is successfully connected, the node core is informed of the new connection details, and the connection status of the newly connected peer is stored in the database (in Android parlance, the `onConnectionInfoAvailable` interface is implemented). Normally, Android requires the user to accept each connection invitation between nodes manually by tapping a consent dialog window on the screen; the device that initiates the connection request must wait until the invited device explicitly agrees to connect. However, iTrust over Wi-Fi Direct simplifies this task by automatically accepting (in addition to automatically making) any device invitations; thus, the application and the individual using the device do not have

to confirm every connection request (including re-connections after accidental disconnects). Fourth, the device begins device negotiation to manage peers within the iTrust membership.

Although the Wi-Fi P2P service component is responsible for the majority of all Wi-Fi Direct related functionality, it cannot by itself service all incoming requests from Android. To handle all communication, it off-loads a majority of the event handling to the Wi-Fi P2P broadcast receiver.

### 5.1.6 Wi-Fi P2P Broadcast Receiver

The Wi-Fi P2P service component operates in the background to service the primary Wi-Fi Direct functions; however, Android requires that a specific component monitors the system for state changes. The Wi-Fi P2P broadcast receiver is similar in function to an interrupt service handler or even a handler manager that continuously receives messages broadcast by Android.

In most cases, the Wi-Fi P2P broadcast receiver simply passes on the messages to the Wi-Fi P2P service component. The Wi-Fi P2P broadcast receiver listens for four main actions emitted by Android: the state change, the peer change, the connection change and the device change actions.

A state change action is simply a Wi-Fi P2P hardware adapter power settings status; listening to this state enables the Wi-Fi P2P broadcast receiver to know whether the Wi-Fi P2P adapter is functioning correctly (and, indirectly, whether

the Wi-Fi P2P broadcast receiver has permission to access the device). The peer change action occurs when a peer is in range of the device, or a peer has left the range of the device; this event is passed on to the Wi-Fi P2P service component for further processing. A connection change action occurs when the device attempts to establish a connection to a particular peer, which simply means that *something* happened in relation to a connection attempt – there is no guarantee that the connection actually succeeded. This event is also passed on to the Wi-Fi P2P service component for further processing; *e.g.*, a successful connection triggers device negotiation to manage the peers. Finally, a device change action signals that the current state of the device has been altered in some meaningful way. This event is primarily useful for the peer management algorithms of iTrust over Wi-Fi Direct and is explained in Section 5.3; however, it does have an important role for the other components detailed here. When a device change action is triggered, it means that the Wi-Fi P2P hardware adapter has changed state and thus can (and should) be read. Reading the state of the Wi-Fi P2P adapter at this point *guarantees* that a valid self MAC address can be extracted from Android.

### 5.1.7 Inbox Thread

The Inbox thread is responsible for reading all incoming messages; it is a common Java network server socket that simply listens/waits for an incoming client socket connection. When a client connection is made, the message is buffered

and passed on (through the common Java *handler* object) to the Wi-Fi P2P service component. The same thread is maintained throughout the life of the Wi-Fi P2P service component; if the Wi-Fi P2P service component is killed and restarted, the Inbox thread is restarted.

### 5.1.8 Outbox Thread

The Outbox thread is responsible for sending all outgoing messages; it is a common Java network client socket that connects to the destination peer's Inbox thread to send the message. Unlike the Inbox thread, the Outbox thread is created to send a specific message; the Outbox thread is created on demand by the Wi-Fi P2P service component, sends its message and then dies. Doing so conserves resources; also, the probabilistic distribution of messages in the iTrust network means that the *next* message sent probably has a different destination peer. Thus, there is little reason to keep the connection open for later immediate use.

### 5.1.9 Android/Linux

The Android/Linux block in Figure 5.1 represents the Android mobile platform, and is combined because the Linux kernel provides the functionality for the Android user space. Importantly, interaction with Linux is required to implement all parts of iTrust over Wi-Fi Direct; for example, the Inbox and Outbox threads use Linux and not Android to set up sockets and transfer information. However,

unlike traditional Linux systems where network functionality is dominated by the kernel, Android plays an important role in controlling the Wi-Fi P2P network adapter (in addition to fundamental iTrust logic non-specific to network-related functions). This interaction between the Android user space and the Linux kernel in creating and using the Wi-Fi Direct connection is elaborated in the next section.

## 5.2 iTrust over Wi-Fi Direct Networking Model

To understand iTrust over Wi-Fi Direct as implemented on the Android platform more fully, it is necessary to understand not only the API and components but also the associated networking model. Figure 5.2 illustrates the iTrust over Wi-Fi Direct networking model by rearranging the components of Figure 5.1 along two axes. Horizontally, the components are separated into the portion of the Android platform to which they pertain: the Android user space on the left and the non-Android portions (mostly the Linux kernel) on the right. The Android user space and the Linux operating system are actually tightly intertwined within the Android mobile platform but, for simplicity, we say that the Android and Linux network *stacks* exist in parallel. Vertically, components are arranged from top to bottom according to the Internet protocol stack in the following layers: Application, Transport, Internet and Link layers. The Physical layer is not shown. The remainder of this section starts with some general observations, and then explains

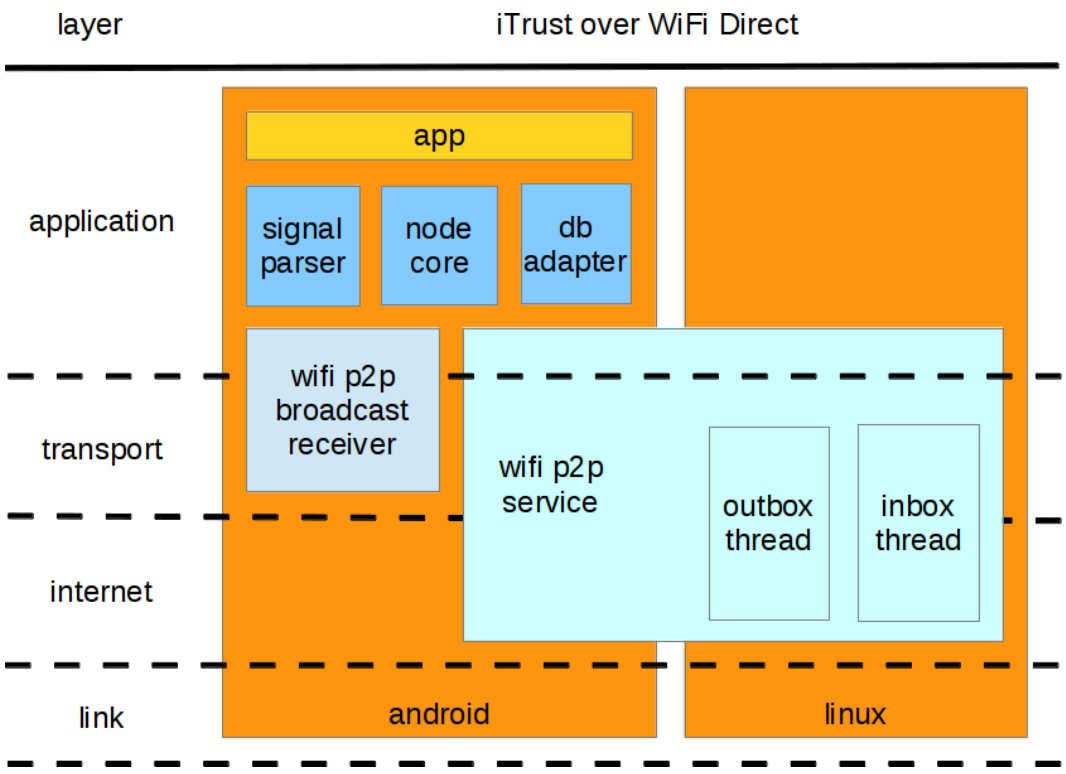



**Figure 5.2:** The iTrust over Wi-Fi Direct networking model.

the placement of the iTrust over Wi-Fi Direct components in their respective layers, beginning with the Link layer and ending with the Application layer.

First, we observe that the iTrust over Wi-Fi Direct components are placed within the layer with which they interact (operating system or user code). Second, while most components individually interact within only a single layer, there are several components that span multiple layers, *e.g.*, the signal parser is entirely within the Application layer and the Wi-Fi P2P broadcast receiver spans both the Transport and the Application layers. Third, a single component might span both network stacks, *e.g.*, the Wi-Fi P2P service component is so complex that it

spans both the Android and Linux network stacks. Finally, although the app is shown in the Android network stack, the user/programmer may chose to use the Linux network stack for purposes not related to iTrust over Wi-Fi Direct. The iTrust over Wi-Fi components are fully contained, and do not require the user to interface directly with a network adapter (indeed, they transparently handle all networking internally); but the programmer may of course add more functionality to the application.

### 5.2.1 Link Layer

The Link layer does not contain any iTrust over Wi-Fi Direct components; however, it is necessary to understand the interaction and differences between Android and Linux within the Link layer.

Using standard Wi-Fi with an access point, the Linux network stack plays a dominant role in providing network access at the Link layer; Android plays only a small role and is (for the most part) insignificant. Similar to the role that Linux plays on portable laptop computers, the Linux kernel provides the Link layer, whereas the user space code (GNU utilities, KDE/GNOME window managers, *etc.*) plays a minor role. Indeed, this organization allows any device, to which the Linux kernel is ported, to gain networking access easily.

However, Wi-Fi Direct is different, and Android plays a much more significant role in the Link layer. Wi-Fi Direct requires much more support from the Android

user space compared to standard Wi-Fi. The differences are vast enough that the entire Wi-Fi Direct access libraries are entirely within the Android Java name spaces. For example, all Wi-Fi Direct code is within the *android.net.wifi.p2p.** name space, instead of the standard Java name spaces in *java.net.**. Consequently, the Linux kernel is not necessarily *aware* of the Wi-Fi P2P network adapter, and the traditional ways of accessing the network state, hardware adapter, *etc.* do not work. Because no other mobile platform provides Wi-Fi Direct support, we do not know whether this organization is a specific Android mobile platform design decision or whether the nature of the protocol does not allow for tight kernel integration. The remaining sections provide more examples of the individualistic nature of Wi-Fi Direct.

### 5.2.2 Internet Layer

The Wi-Fi P2P service component spans both the Android and the Linux network stacks at the Internet layer: the Android side mostly accounts for the Wi-Fi Direct functionality, whereas the Linux side is relatively minor.

Most Wi-Fi P2P service component functionality occurs within the Internet layer on the Android stack. Here, the Wi-Fi Direct hardware adapter is probed and powered on, peers are discovered and connections are made to peers. During the connection phase, there is a pseudo P2P negotiation between peers to exchange basic network information including: MAC address, unique device name

and adapter configurations. Once a connection is made, the Wi-Fi Direct hardware adapter state changes and is internally saved within Android (specifically within an Android *Intent* object). In summary, the Wi-Fi P2P service component within the Internet layer deals with finding and connecting to peers; once a connection is made, control is passed up to the Transport layer (and eventually to the Application layer).

The Linux network stack plays a minor role here; instead of establishing a network, it mostly off-loads this functionality to the Android stack. However, because the Linux network stack is not involved with peer connections taking place on the Android stack, it does not have MAC address information. Although finding MAC addresses of other nodes in the network is *not* part of the TCP/IP model, it is often used by a device for finding *its own* MAC address. (Note that Wi-Fi Direct identifies peers by MAC address.) Because the Linux stack is not aware of its own MAC address (because the Wi-Fi P2P adapter is controlled entirely by the adjacent Android stack), it cannot play any major role in network functions in higher network layers.

Trivially, the Outbox thread and Inbox threads have access to the Linux network stack in the Internet layer (the functionality is similar to traditional client/server sockets). However, most of the socket functionality is in the Transport layer, discussed next.

### 5.2.3 Transport Layer

The Transport layer contains four components: the Wi-Fi P2P broadcast receiver, the Wi-Fi P2P service component, the Outbox thread and the Inbox thread.

The Android stack within the Transport layer contains a portion of the two Wi-Fi P2P components. The Wi-Fi P2P broadcast receiver has the critical functionality of interfacing with the Wi-Fi P2P service component in the Transport layer. Specifically, the Wi-Fi P2P broadcast receiver passes control from its Application layer side to its Transport layer side and passes on event states to the Wi-Fi P2P service component.

The important task of the device's reading its own MAC address is also performed in the Transport layer between the Wi-Fi P2P service component and the Wi-Fi P2P broadcast receiver. Once the Wi-Fi P2P service component establishes a connection with a peer (in the aforementioned Internet layer on the Android stack), an event is triggered within Android (device changed action); this event is caught by the Wi-Fi P2P broadcast receiver. At this point, the Wi-Fi P2P broadcast receiver parses the Android Intent object (previously set by the Wi-Fi P2P service component in the Internet layer) and successfully extracts a valid MAC address associated with the Wi-Fi Direct adapter. The device now effectively can distinguish itself among the other peers, and shares its MAC address with other peers; iTrust over Wi-Fi Direct can then establish and automatically

maintain a persistent network connection with peers. Importantly, the timing of the Intent object parsing is crucial: it *must* be done immediately after the device changed action is received and *before* any other event is received; otherwise, there is no guarantee that the MAC address exists or is readable.

As mentioned in Section 5.2.2, the Linux stack has no major role in Wi-Fi Direct and, for this reason, the traditional Transport layer utilities provided by Unix are not useful. Specifically, the Unix address resolution protocol (ARP) tables cannot be used and, thus, IP addresses (for sending/receiving files) cannot be queried from Linux. Instead, we provide this functionality in the iTrust over Wi-Fi Direct peer management system.

The Wi-Fi P2P service component plays a much more passive role in the Transport layer than in the Internet layer; the component simply passes messages between the Application layer portion (described in Section 5.2.4) and across to the Outbox/Inbox threads in the Linux stack in the Transport layer. Trivially, the Wi-Fi P2P service component creates the persistent Inbox thread and the on-demand Outbox thread. Recall that both the Inbox thread and the Outbox thread exist only within Linux and are not part of Android; for that reason, they exist only in the Linux stack.

For example, when the application sends a request or metadata message, the node core sends the message to the Application layer portion of the Wi-Fi P2P service component, the message goes from the Application layer to the Transport

layer (still within the Wi-Fi P2P service component), transfers from the Android stack to the Linux stack and is passed on to the Outbox thread. When a message is received, Linux informs the Inbox thread in the Transport layer, the message is passed on to the Wi-Fi P2P service component (within the Transport layer from the Linux stack to the Android stack), passed up from the Transport layer to the Application layer within the Wi-Fi P2P service component, and finally passed on to the signal parser.

### 5.2.4 Application Layer

The Application layer is relatively simple compared to the other network layers. Most of the iTrust over Wi-Fi Direct components exist in the Application layer, but are not network related. The signal parser, node core and DB adapter components have no networking functions; indeed, the Wi-Fi P2P service component and the Wi-Fi P2P broadcast receiver are specifically charged with managing all network access in iTrust. The only (minor) exceptions are that the node core sends outgoing messages to the Wi-Fi P2P service component, and the signal parser receives incoming messages from the Wi-Fi P2P service component. All of these component interactions occur on the Android stack in the Application layer.

The Wi-Fi P2P broadcast receiver, within the Application layer, interfaces directly with Android to capture events. In effect, it reads the reactions of the

Wi-Fi Direct network adapter and relays the information to the Wi-Fi P2P service component.

The Wi-Fi P2P service component again spans the Android and Linux stacks within the Application layer. Apart from the small but important roles of relaying messages between threads and the iTrust logic components, the Wi-Fi P2P service component handles the complex task of managing peer connections within the Application layer. The details of the peer management, such as how peer IP addresses are assigned, how peers join the membership and how connections are repaired, are discussed next.

## 5.3 Peer Management

Now, we discuss the limitations of Wi-Fi Direct on the Android operating system, our solutions to the peer management problem, and a cursory analysis of the message cost.

### 5.3.1 Limitations of Wi-Fi Direct on Android

Although Wi-Fi Direct is supported on the Android operating system, version 4.1 and above (and not yet supported on other mobile platforms), there are serious limitations regarding the peer functionality and data routing techniques.

Android regards Wi-Fi Direct as mostly a Data Link layer function with little support for the Network or Transport layers; namely, there is no direct associa-

tion between MAC addresses and IP addresses. Regardless of the Wi-Fi Direct specification and whether or not it mandates this association, the lack of a direct MAC address to IP address mapping inhibits many useful network setups.

On activating the Wi-Fi Direct functionality, a peer broadcasts its MAC identifier, node identifier and user name, and searches for other peers; when other peers are found, an internal list of MAC addresses is updated. This MAC address list can be queried, and individual peers can be selected and connected to form a P2P network. When a network is created, one peer within the group creates a *soft access point*, establishes itself as the network group owner and assigns itself an IP address. The IP address of the group owner is transmitted to all peers, and the connection process effectively ends.

However, there are several critical limitations, because only one peer in the network has an effective IP address. This scheme might be adequate for P2P gaming or other simple tasks, such as simple file transfer between two devices; however, it does not scale well to larger numbers of peers. Without effective IP addresses, three or more peers cannot communicate directly with each other; they are forced to go through the group owner peer.

Furthermore, there is no reliable way for a peer to determine this information by itself, if it is not the group owner. Android provides no (at least documented) way to query for this information. Standard Java methods do not work either; all Java networking functions return information about the Wi-Fi adapter but *not*

the Wi-Fi P2P adapter. The underlying Linux system also provides no help; the Address Resolution Protocol (ARP) table is periodically flushed and not reliable. Additionally, there is no documented way for a node to find its own MAC address for the Wi-Fi P2P adapter.

### 5.3.2 A Method to Manage Peers

To solve this problem, we created the relatively simple method illustrated in Figure 5.3. Part A describes the peer management process; part B describes the metadata distribution process; and part C describes the query distribution and resource transfer process. Parts B and C are included for completeness of the iTrust over Wi-Fi Direct message discussion, and are described briefly below.

As shown in Figure 5.3 part A, we consider two peers $X$ and $Y$ that are in range of each other and are available to connect over Wi-Fi Direct. The following actions occur on each peer independently (each device calls its own functions).

Once the Wi-Fi P2P Broadcast Receiver object is notified that peer(s) are available, it triggers the Wi-Fi P2P Service object. The Wi-Fi P2P Service object determines the availability of the nearby peers and automatically initiates a connection (handled automatically by iTrust whereas, otherwise, it must be manually initiated by Android). A connection process begins, and the peers negotiate (through callbacks from the peers' Wi-Fi P2P Broadcast Receiver objects to the Wi-Fi P2P Service objects); one peer is randomly chosen as the group owner. At

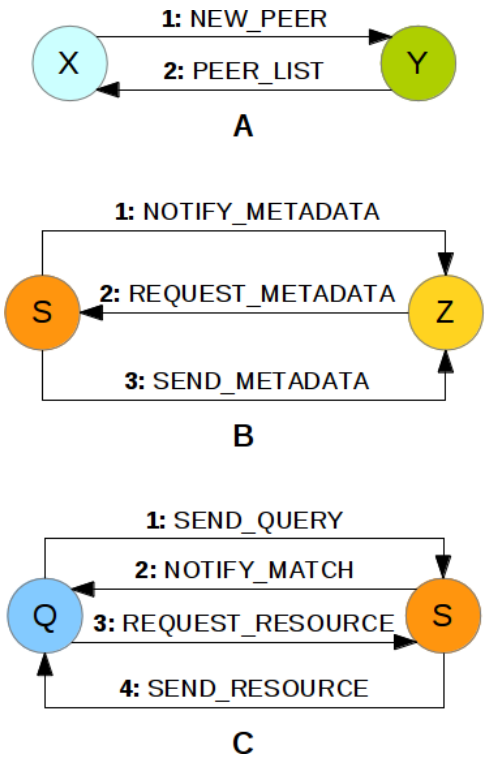



**Figure 5.3:** The iTrust message types: (A) Peer management, (B) Metadata distribution and (C) Resource search and retrieval.

this point, iTrust automatically finalizes the connection without explicit Android confirmation. On finalizing the connection, if a peer determines that it is *not* the group peer owner, it sends a *NEW_PEER* message to the group owner.

In Figure 5.3 part A, $X$ is not the group owner and so it sends (1) a *NEW_PEER* message to $Y$. The *NEW_PEER* message contains only the MAC address of $X$; an undocumented but public feature was found in Android and used to extract the peer's Wi-Fi Direct MAC address (effectively a bit-masked value in a parsed Intent object of the Activity/Service class). Recall that only the IP address of the group owner ($Y$) is known. On accepting the socket data transfer, $Y$ now

knows the MAC address and the IP address of $X$; the MAC address of $X$ is the message payload and, by virtue of creating a socket connection and reading the socket object information, the IP address of $X$ can be determined. The group owner peer saves this MAC/IP address association in its database, generates a JSON list of all saved MAC/IP address associations and distributes the JSON list in (2) the *PEER_LIST* message to $X$. The only data in the *PEER_LIST* message is the JSON list of MAC/IP address associations; peer $X$ decodes the JSON list and saves the associations. Both peers now have *their* up-to-date and identical memberships.

This process, the reporting of MAC addresses and receiving of MAC/IP addresses, can continue for any number of peers in the network (assuming they are within range). If the connection is severed, due to range or noise, iTrust automatically reconnects and rebuilds the MAC/IP address pairs.

We tested these peer management mechanisms on two or three physical Nexus 7 tablets with a test harness of the iTrust over Wi-Fi Direct API. Data were transferred easily and automatically, and no user interaction was required apart from making sure that the devices were in range. Furthermore, when the connection was severed, reconnection was automatically established when the problem was corrected. Currently, Wi-Fi Direct does not work on the Android emulator, so further tests require more physical devices.

### 5.3.3 Metadata and Query Distribution Messages

Parts B and C in Figure 5.3 are similar to the message protocol used in the iTrust over SMS network described in Section 4.4 and [57]. Note that the original iTrust over SMS network required only seven messages, whereas the iTrust over Wi-Fi Direct network requires nine messages (including the two additional peer management messages).

In part B, metadata are distributed in the following order: the source peer $S$ sends (1) the *NOTIFY_METADATA* message, some time later peer $Z$ asks for the metadata using (2) the *REQUEST_METADATA* message, immediately after $S$ sends the metadata using (3) the *SEND_METADATA* message to $Z$.

In part C, a query is distributed and a resource is transferred in order: the requesting peer $Q$ sends (1) the *SEND_QUERY* message, the metadata are encountered on $S$, which sends (2) the *NOTIFY_MATCH* message to $Q$, some time later $Q$ sends (3) the *REQUEST_RESOURCE* message to $S$ (3), immediately after $S$ sends the resource using (4) the *SEND_RESOURCE* message to $Q$.

## 5.4 Performance Evaluation

In Section 5.1, we described the iTrust over Wi-Fi Direct system, which highlighted the similarities of certain parts of iTrust over SMS and iTrust over Wi-Fi Direct. By comparing Figures 4.5 and 5.1 with the component descriptions of

Sections 4.4 and 5.1, respectively, it is clear that the signal parser, node core and DB adapter components are very similar. Because of this similarity, most of the iTrust over SMS performance evaluation results from Section 4.8 also apply to the iTrust over Wi-Fi Direct performance evaluation. In terms of the probability of a match (Section 4.8.1), the number of messages to achieve a match (Section 4.8.2) and the importance of $2 * sqrt(n)$ (Section 4.8.5), the performance evaluation of iTrust over SMS is identical to that of iTrust over Wi-Fi Direct.

However, the performance evaluation of iTrust over SMS related to emulation of the membership (Section 4.8.4) does not carry over to iTrust over Wi-Fi Direct. Unfortunately, the Android emulator does not support Wi-Fi Direct emulation capabilities and therefore all experiments must be performed on real physical devices (as noted in Section 5.3.2). Although it is economically infeasible to create even a small membership as in Section 4.8.4 (*e.g.*, 16 Android devices would have to be purchased), it is possible to create the smallest possible network (two to three devices) and establish correct network functionality and protocol behavior. To this end, we created a three-device iTrust over Wi-Fi Direct network using two Asus Nexus 7 8GB tablets to test the iTrust over Wi-Fi Direct functionality, protocol adherence, graphical user interface functionality and peer management method (as previously noted). Correct functionality was observed, and it was possible to share information between devices by searching for and fetching resources across tablets using Wi-Fi Direct.

Because of the large difference in speed between SMS and Wi-Fi Direct, a discussion of mean search latency in SMS (Section 4.8.6) is not applicable to Wi-Fi Direct. While the mean search latency of iTrust over SMS might vary greatly depending on the network service provider and the signal strength, the mean search latency of iTrust over Wi-Fi Direct is uninteresting. In all of our experiments, the searches performed using iTrust over Wi-Fi Direct took less than one second, well below the threshold that any typical human user would notice. However, the large increase in the speed of iTrust over Wi-Fi Direct, compared to iTrust over SMS, allows users to transfer large file sizes relatively easily and quickly between devices; an ability that was technically possible in iTrust over SMS but cumbersome due to the low transfer speed of SMS. In Section 5.4.1, we consider the resource transfer latency of iTrust over Wi-Fi Direct.

Moreover, an analysis of a medium-size membership in iTrust over SMS (Section 4.8.3) is only partially applicable to iTrust over Wi-Fi Direct. All messages for iTrust over SMS and iTrust over Wi-Fi Direct are similar in size and type, but iTrust over Wi-Fi Direct has two additional message types for peer management. Thus, in Section 5.4.2, we provide an analysis of the additional message cost for peer management in iTrust over Wi-Fi Direct.

### 5.4.1 Resource Transfer Latency

To measure the resource transfer latency, the time taken to transfer resource data from the source node to the requesting node, we created a test driver application on Android that uses the iTrust over Wi-Fi Direct API previously discussed. The application was loaded onto the same previously mentioned Nexus 7 tablets. Each tablet was loaded with 15 resources; the tablets were placed side-by-side and close together (2 cm apart); and queries and retrievals were performed simultaneously. The experimental environment was not shielded in any way and resembled the typical environment of two typical users: a small room with furniture, numerous public Wi-Fi access points nearby but not network connected to the tablets and other nearby Wi-Fi signals from laptops or other wireless networkable devices.

In particular, the 15 resources (files) on each tablet resembled files that would be typically shared among protesters/demonstrators or average persons. The files fall into three categories: small files such as text documents or pictures (under 1MB), medium files such as MP3 audio recordings (about 5MB) and large files such as AVI videos (about 150MB). The typical demonstrator might share PDF files, pictures taken from a camera phone, audio recordings of a gathering, and video recordings of a demonstration. The average person might have similar text files, pictures, music files and movie files.

Table 5.1 shows the file transfer latency metrics for 30 data transfers (15 file fetches from each device). Note that file transfers were done in pairs; each device

| Metrics | Small files | Medium files | Large files |
|---|---|---|---|
| Minimum file size (bytes) | 188236 | 3783308 | 144715776 |
| Maximum file size (bytes) | 955165 | 5755928 | 162666496 |
| Mean file size (bytes) | 453962 | 5013612 | 151327949 |
| Mean transfer time (seconds) | 0.39 | 9.74 | 78.82 |
| Mean transfer rate (kbytes/second) | 2493 | 620 | 1940 |

**Table 5.1:** Resource transfer latency for transferring files between Wi-Fi Direct devices.

attempts to retrieve a file while it simultaneously sends its own file to the other device. This experimental set up was used because a typical demonstrator or protester might not wait to send or retrieve a file; indeed, the file transfer might take place automatically while the device is stored in a bag (for tablets) or a pocket (for smart phones). iTrust over Wi-Fi Direct does not have a fixed policy on prioritizing message sending; it is up to the network medium (in this case Wi-Fi Direct) to handle congestion.

The network performance for iTrust over Wi-Fi Direct with respect to small files is adequate to service the needs of a demonstrator or an average user; with a mean transfer rate of 2493kB/s, the typical small file is transferred in less than a second. Thus, iTrust over Wi-Fi Direct scales acceptably for transferring multiple small files, such as a picture album by the typical user, who takes pictures from a camera phone.

Medium file network performance is unexpectedly low: a typical 5013612B (5MB) file takes almost 10 seconds to transfer, which is relatively slow for Wi-Fi. However, the performance is acceptable in several cases; in particular, it is easy to stream (download and simultaneously playback) an audio file from one tablet to another because the typical MP3 bit-rate is 128kb/s and we achieved 620kB/s in our experiments (note that streaming is often given in bits per second and Wi-Fi Direct is measured in bytes per second). We do not understand why the performance degraded so much (especially for large files), although it is possible that the transfer buffer size did not correctly match the packet payload size. A detailed analysis of the Wi-Fi Direct firmware for the Nexus 7 tablet is required to see if this is the problem.

The large file performance is surprisingly better than the medium file performance; again, there might be a buffer size problem with Wi-Fi Direct, or Android might be flushing the buffer at the wrong time. In this case, the transfer rate is also acceptable to stream video from one device to the other; an AVI video file might not stream well, but 1940kB/s is adequate to stream a high definition video encoded in H.264. However, the mean time to transfer the file might be unacceptable to someone who simply wants to copy the file as soon as possible and not to stream it.

Finally, we note that the mean resource rates for both small and large files are greater than the rate used to transfer the test files from a desktop computer to

the Nexus 7 tablets by USB cable. Therefore, in some cases and for some devices, it might benefit the user to transfer files between devices wirelessly even if a wired connection is available and ready.

### 5.4.2 Message Cost

Given a network of $n$ peers, the number of messages required to synchronize all peer lists is $n + (n-1) + (n-2) + ... + 1 = \frac{n(n+1)}{2}$, for $n \geq 2$. This calculation assumes that a new *PEER_LIST* message is sent immediately after every *NEW_PEER* message but before another peer joins the network (as the process repeats). Immediately sending the *PEER_LIST* message after a *NEW_PEER* message is feasible in small networks; however, it quickly becomes impractical for large networks. In the case of large networks, a simple delay before sending the *PEER_LIST* message allows more time for other peers to "report in"; this delay can be set dynamically by iTrust depending on the membership size.

## 5.5 Summary

In this chapter, we have presented iTrust over Wi-Fi Direct and described both an Android implementation and a new peer management technique which overcomes the current limitations of Wi-Fi Direct. The iTrust over Wi-Fi Direct implementation works on Android devices, including tablets and smart phones, and enables several physically near devices to distribute and search for informa-

tion among themselves, completely separate from any form of centralized control. We extended the iTrust over SMS implementation to iTrust over Wi-Fi Direct; basic iTrust functionality, such as keyword/metadata matching, database schemas, *etc.*, is mostly unchanged. We added several new components: Wi-Fi P2P service, Wi-Fi P2P broadcast receiver, inbox and outbox. To understand the iTrust over Wi-Fi Direct system more fully, we examined the networking model of the Android implementation and explained the interaction between the Android and Linux networking stacks. Finally, we presented a performance evaluation of iTrust over Wi-Fi Direct, including a discussion of resource transfer latency and peer management message cost.

# Chapter 6

# Related Work

A variety of networks and systems share some of the characteristics of the iTrust information publication, search and retrieval system. In this chapter, first we discuss several centralized and decentralized systems and how they compare to the iTrust system. Second, we consider other structured and unstructured peer-to-peer networks and compare them to iTrust by focusing on metadata and query message dissemination. Next, we discuss the trust, reputation and malicious nodes aspects of iTrust and how other systems address these issues. Finally, we highlight other mobile search strategies over cellular and ad-hoc networks, and compare them to iTrust.

## 6.1 Centralized and Decentralized Search

The centralized search engine strategy, such as that of Google, stores metadata for information in a centralized index, and matches keywords in the requests against the metadata stored at the central site. The centralized search

engine strategy is used commercially for Internet search because it is efficient, but it is vulnerable to manipulation, filtering and censorship. The centralized publish/subscribe approach [31] also uses a centralized index, against which the queries are matched, raising the same issues of trust as the centralized search engine strategy.

Bender *et al.* [7] recognize the need for decentralized peer-to-peer Web search because "existing Web search is more or less exclusively under the control of centralized search engines."

Gnutella [39], one of the first decentralized networks, uses flooding of requests to find information. Extensions of Gnutella involve super nodes [97], which improve efficiency but incur some of the trust risks of centralized strategies, and biased random data replication [12], which use randomization and replication like iTrust does.

Freenet [24] is a more sophisticated and efficient system than Gnutella, because it learns from previous requests. In Freenet, nodes that successfully respond to requests receive more metadata and more requests. Thus, it is easy for a group of untrustworthy nodes to conspire together to gather most of the searches into their group, making Freenet vulnerable to subversion.

Other peer-to-peer systems, such as that of Lv *et al.* [63], use random walks to improve on the flooding of Gnutella. They start with uniform random replication of data, but then adaptively adjust the replication degree based on the query

rate, and use square root replication to improve performance. They also consider creation and deletion of the replicas of the data (or metadata). BubbleStorm [86] replicates both queries and data, and combines random walks with flooding to perform exhaustive search. It also considers churn, leaves and crashes, like the iTrust membership protocol does.

Pub-2-Sub [89] is a publish/subscribe service for unstructured peer-to-peer networks of cooperative nodes, that uses directed routing (instead of gossiping) to distribute subscription and publication messages to the nodes.

PlanetP [27] maintains a local index that contains metadata for documents published locally by a peer, and a global index that describes all peers and their metadata. It replicates the global index throughout the network using gossiping. Galanx [94] uses a local peer index to direct user queries to relevant nodes in the network. It is based on the Apache Web server and on the BerkeleyDB data store. Our iTrust over HTTP system likewise utilizes the Apache Web server, and maintains a local index of metadata and corresponding URLs for the data. None of the above systems is particularly concerned with trust, as iTrust is.

## 6.2 Structured and Unstructured Networks

The structured approach [8, 42, 51, 85] requires the nodes to be organized in an overlay structure, based on Distributed Hash Tables (DHTs), trees, rings, *etc.* The structured approach is more efficient than the unstructured approach, but

it involves administrative control and additional overhead for constructing and maintaining the overlay network. Moreover, churn or malicious disruptions can break the structure.

Adamic and Adar [3], and also Watts *et al.* [95], have investigated the effectiveness of search in social networks, which appears to depend on the structured nature of those networks and a few highly-connected nodes. Many searchers were able to exploit that structure to find information in relatively few steps. In experiments with students where such structure does not exist, such local search strategies were less effective.

The unstructured approach [24, 34, 39, 45, 86, 89, 96, 100] is typically based on gossiping, uses randomization, and requires the subscriber nodes and the publisher nodes to find each other by exchanging messages over existing links.

Cohen and Shenker [25] have studied how replication can be used to improve search in unstructured peer-to-peer networks. They show that square root replication is theoretically optimal in terms of minimizing the overall search traffic. They replicate objects based on access frequencies (popularities), whereas iTrust uses uniform random replication of objects, so that popular nodes are not more vulnerable to attacks.

GIA [12] is an unstructured Gnutella-like peer-to-peer system that combines biased random walks with one-hop data replication to make search more scalable. Likewise, Sarshar *et al.* [80] combine random walk data replication with a two-

phase query scheme in a Gnutella-like network for scalability. Yang and Garcia-Molina [97] use super nodes to improve efficiency, but reintroduce some of the trust risks of centralized strategies in doing so.

Zhong and Shen [100] use random walks for requests, where the number of nodes visited by a request is proportional to the square root of the request popularity, as in [25]. Ferreira *et al.* [34] use random walks to replicate both queries and data to the square root of the number of nodes in the network. Unlike [25], in their system, replication of metadata and requests is independent of access frequency (popularity), as in iTrust. Like these other researchers, we also exploit the square root function in iTrust.

## 6.3 Trust, Reputation and Malicious Nodes

Several other systems are similar to iTrust in that they are concerned with trust. Quasar [96] is a probabilistic publish/subscribe system for social networks with many social groups. The authors note that "an unwarranted amount of trust is placed on these centralized systems to not reveal or take advantage of sensitive information." iTrust does not use a structured overlay, and has a different trust objective than Quasar.

OneSwarm [45] is a peer-to-peer data sharing system that allows data to be shared either publicly or anonymously, using a combination of trusted and untrusted peers. OneSwarm is part of an effort to provide an alternative to cloud

computing that does not depend on centralized trust. Its initial goal is to protect the privacy of the users; iTrust does not aim to conceal the users like OneSwarm does.

Safebook [28] is a social network that preserves anonymity, by communicating information through intermediary nodes. Both OneSwarm and Safebook aim to protect the users' privacy, which iTrust does not aim to do. Rather, the trust objective of iTrust is to support the free flow of information and to prevent censorship, filtering and subversion of information. However, some of the ideas of those systems might be useful for a future version of iTrust.

Jesi *et al.* [46] identify malicious nodes in a random overlay network based on gossiping and put them on a blacklist. They focus on hub attacks in which colluding malicious nodes partition the network by spreading false rumors. iTrust does not use gossiping but, rather, uses random distribution of the metadata and the requests and, thus, is less subject to hub attacks.

Zhou and Hwang [101] present a gossip-based reputation aggregation system, named GossipTrust, for unstructured peer-to-peer networks. The system aggregates locally-generated feedback from the peers using gossiping, to obtain global reputation scores, which allow peers to make informed decisions about which peers to trust.

Damiani *et al.* [30] present a reputation-based protocol, named XRep, for choosing reliable resources in peer-to-peer networks. Reputation sharing is real-

ized through a distributed polling algorithm, by which requestors can assess the reliability of a resource before initiating a download, thus reducing the spread of malicious content.

Condie *et al.* [26] present a protocol for finding adaptive peer-to-peer topologies that protect against malicious peers that upload corrupt, inauthentic or misnamed content. Peers improve the trustworthiness of the network by forming connections, based on local trust scores defined by past transactions. Effectively, their protocol disconnects malicious peers and moves them to the edge of the network.

## 6.4 Mobile Search over Cellular Networks and Ad-Hoc Networks

In a study of mobile search behavior, Kamvar *et al.* [47] found that most mobile searchers use the search service for a short period of time, do not engage in exploration, and have a specific topic in mind. In a subsequent study [48], they found that the diversity of mobile search topics is rather limited. Evans and Chi [32] have provided an analysis of the activities of individuals conducting search over social networks, with a focus on foraging and sense making. Church and Smyth [23] have also addressed the information needs of mobile users.

Existing mobile search services include AOL Mobile [69], Google SMS [40], Windows Live Mobile [70] and Yahoo! OneSearch [74]. Those services provide Web search on mobile devices, and use conventional centralized Web search engines. They provide a limited set of pre-defined topics, and use either special keywords within a search query (*e.g.*, "directions" to obtain directions) or a specialized parser to determine the intended topic (*e.g.*, "INTC" for a stock quote). For queries related to arbitrary topics, the results obtained are sometimes not meaningful or not consistent. Moreover, the centralized search engines on which those systems depend are subject to censorship, filtering and subversion.

The SMSFind system [13, 14] also utilizes a conventional centralized search engine at the back-end. However, it does not use pre-defined topics but, rather, allows the user to enter an explicit contextual hint about the search topic. SMSFind uses information retrieval techniques to extract an appropriate condensed 140-byte snippet as the final SMS search response, which iTrust currently does not do but which might be valuable for a future version of iTrust.

The Mobile Agent Peer-To-Peer (MAP2P) system [44] supports mobile devices in a Gnutella file-sharing network using mobile agents. The mobile agent (rather than the mobile device) attaches itself to the peer-to-peer network, and acts as a proxy for the mobile device. In some respects, the MAP2P mobile agent is similar to the iTrust SMS-HTTP bridge node, but iTrust has a lower message cost than Gnutella and, thus, MAP2P.

The Distributed Mobile Search Service [53] broadcasts query results locally and forwards them over several hops. It is based on a passive distributed index that comprises, on each mobile device, a local index cache, containing keywords and corresponding document identifiers, where all received query results are cached. The iTrust system also maintains a distributed index, with metadata keywords and corresponding URLs stored on the iTrust nodes. However, iTrust distributes the metadata and corresponding URLs first, rather than on receipt of the query results and, thus, has a lower message cost.

The 7DS system [75] supports information sharing among mobile devices. The 7DS system uses a multi-hop flooding algorithm together with multicasting of queries, which is not trustworthy. In contrast, the iTrust system forwards messages selectively to nodes based on a relay probability that limits the number of nodes to which the metadata and the requests are distributed to about $2\sqrt{n}$ nodes, where $n$ is the number of nodes in the membership [65].

Smozzy [83] is a service that allows a user to browse the Web using only SMS/MMS on any MMS-capable mobile phone. Users send a URL through SMS to a Smozzy server; the Smozzy server fetches the URL and repackages it into an MMS message. The user reads the MMS message as if it were a Web page. Smozzy is an Android app with the primary purpose of letting T-Mobile text message only users accessing the Web. A user pays much less for SMS/MMS access than normal 2/3/4G data access (currently \$15/month for unlimited SMS/MMS versus

$30/month for 4GB throttled data). In turn, SSL is not supported for users and T-Mobile will have their network control-channel saturated. iTrust is not meant for viewing existing Web sites but, instead, for allowing individuals to share files in a decentralized manner (*i.e.*, no Smozzy server).

PeopleNet [72] is a social network that exploits physical location to facilitate searching. The authors observed a rapid increase in the number of copies of a query as it propagates in search of data, akin to flooding. Thus, they advocate a swap strategy in which a request migrates but does not replicate itself. iTrust over SMS explicitly manages the replication of queries to achieve a desired probability of finding a match.

Motta and Pasquale [73] recognized the opportunity that Wi-Fi Direct presents, even before it became available on Android. They describe a JXTA middleware architecture for peer-to-peer networks, which optimizes mobile resources and exploits the features of mobile devices. They apply the JXTA middleware to a search infrastructure for structured peer-to-peer networks that uses resource indexing based on distributed hash tables (DHTs). The iTrust over Wi-Fi Direct system for search and retrieval uses an unstructured approach, which is more appropriate for mobile ad-hoc networks.

Like the iTrust project, the Commotion Wireless project [35] aims to ensure that communication cannot be controlled or cut off by authoritarian regimes. Their device-as-an-infrastructure distributed communication platform integrates

Wi-Fi enabled mobile phones, computers and other personal devices to create a metro-scale communication network that supports local peer-to-peer communication and local-to-Internet communication.

Thomas and Robble [87] have created a mobile ad-hoc network for disaster and emergency relief, using the Wi-Fi chips in Android smart phones, allowing them to connect without using cellular networks. Their Smart Phone Ad-Hoc Networks (SPAN) project reconfigures the onboard Wi-Fi chip of a smart phone to act as a Wi-Fi router to nearby similarly configured smart phones. SPAN intercepts communications at the Global Handset Proxy, so that typical applications, such as email, Twitter, *etc.*, still work. In contrast, iTrust for mobile ad-hoc networks uses Wi-Fi Direct, which Android now supports.

The Serval project [38] is developing a wireless ad-hoc mobile phone platform, named Serval BatPhone. The project targets rural and remote populations, disaster and emergency relief, and governments that disable the Internet or the cellular network. The team chose to use Wi-Fi ad-hoc mode in the ISM2400 band and Android mobile phones. At the time, Android phones did not support ad-hoc Wi-Fi, so they had to manipulate the Wi-Fi hardware on the Android phones. Our implementation of iTrust for mobile ad-hoc networks uses Wi-Fi Direct, which Android now supports.

Meroni *et al.* [67] describe an opportunistic platform for Android-based devices using Wi-Fi in a mobile ad-hoc network. The opportunistic platform they propose

is intended to address concerns of scalability, flexibility and bandwidth in cellular networks by supporting local peer-to-peer communication between nodes. Their platform enables nodes to query for information and receive responses locally and, thus, to save network bandwidth, if that information is large.

## 6.5 Summary

In this chapter, we have presented systems and networks that bear some similarity to iTrust, and have categorized the related work into four separate sections. The centralized and decentralized search section compares and contrasts centralized and decentralized systems that share some characteristics with iTrust. The section on structured and unstructured peer-to-peer networks highlights the message routing and information distribution techniques of those networks. The section on trust, reputation and malicious nodes discusses node anonymity, user privacy, node reputations, *etc.* Finally, the section on mobile search over cellular and ad-hoc networks addresses networks that relay searches.

Compared to these other networks and systems, the iTrust system is a decentralized system for information publication, search and retrieval. iTrust operates over an unstructured peer-to-peer network that use probabilistic distribution of metadata and queries. iTrust does not attempt to conceal information but, rather, aims to guarantee the free flow of information. Our implementations of iTrust use a variety of network technologies, including HTTP over the Internet, SMS over

the mobile cellular network and Wi-Fi Direct over mobile ad hoc networks. Thus, iTrust enables users to share information directly among themselves across heterogeneous networks.

# Chapter 7

# Conclusions and Future Work

We have described iTrust, a novel information publication, search and retrieval system with no centralized mechanisms and no centralized control. iTrust involves distribution of metadata and requests, matching of requests and metadata and retrieval of information corresponding to the metadata. We have shown that, with iTrust, the probability of matching a query is high even if some of the participating nodes are subverted or non-operational. The iTrust system is particularly valuable for individuals who wish to share information, without having to worry about subversion or censorship of information.

The iTrust over HTTP implementation follows the design of iTrust by combining the Apache Web server and JAR files, the custom PHP code to manage metadata and searching, and the public interface accessible over a Web browser. An administrator can easily setup an iTrust over HTTP node and join an iTrust network. A user can upload and distribute metadata, perform search queries and fetch resources easily using a Web browser. We compared the performance

of the iTrust over HTTP implementation with the results from the theoretical probabilistic analysis, using several performance metrics.

The iTrust with SMS system enables SMS-capable mobile phones to communicate with iTrust SMS-HTTP bridge nodes that act as relays to iTrust over HTTP nodes for information search and retrieval in the iTrust network. Thus, an SMS-capable mobile phone can access information on any number of interconnected iTrust over HTTP nodes, and the iTrust over HTTP nodes can be queried from any SMS-capable mobile phone for search and retrieval of information. An Android mobile phone application provides a custom interface to facilitate search and retrieval over the iTrust network.

The iTrust over SMS system enables SMS-capable mobile phones to communicate directly over the cellular telephony network to distribute metadata and to search for and retrieve information. Information stored locally on any iTrust over SMS mobile device can be sent directly to another such mobile device by Instant Messages; Internet access is not required. In the iTrust over SMS network, mobile devices can send information to mobile devices using different platforms, such as Android, iOS, *etc.*, as long as each platform implements the iTrust over SMS protocol. The iTrust over SMS protocol is implemented as an Android application, and the custom user interface for iTrust with SMS was re-purposed to create a low processor intensive and rudimentary iTrust over SMS graphical user interface. A developer for any mobile phone based on the Android platform can use the

iTrust over SMS API to add this search and retrieval functionality to an existing application or even create a new application based on iTrust over SMS. Our performance evaluation of iTrust over SMS shows, by analysis for a relatively large network and by emulation for a smaller network, that the probability of information retrieval is high, even if some of the mobile phones are not available. It also shows that the mean latency is consistently less when the participating nodes use the same mobile service provider, and consistently more when they use different mobile service providers.

We have created a rich graphical user interface in Android for iTrust over SMS which enables smart phone users to use many familiar touch gestures and widgets to search and distribute information. We have presented use cases for iTrust over SMS using this rich interface, namely, the sporadic, casual and avid searchers who distribute and search for documents, as well as the pure searchers who search for but do not distribute documents.

The iTrust over Wi-Fi Direct system is a peer-to-peer information publication, search and retrieval system that operates over mobile ad-hoc networks. We have described the iTrust over Wi-Fi Direct API and components as implemented on the Android platform for mobile devices, and have shown how user applications can easily interface with the API to gain P2P functionality. We have presented the iTrust over Wi-Fi Direct networking model, and have described the interactions between the Android and Linux stacks.

We have described peer management for iTrust over Wi-Fi Direct on the Android platform that enables peers to construct a mobile ad-hoc network for decentralized search and retrieval. The peer management algorithm for iTrust over Wi-Fi Direct automatically discovers and connects to available peers. We highlighted deficiencies of the Android platform with Wi-Fi Direct, and presented our peer management solution to address those limitations. The iTrust over Wi-Fi Direct peer management algorithm facilitates the creation of mobile ad-hoc networks over mobile devices, and is a step towards making decentralized personal search feasible and more convenient.

In the future, our primary goal is to make decentralized search more accessible to the lay person who is consumer technology savvy. For iTrust over HTTP, we plan to release, as open source software, the entire iTrust over HTTP code base and associated files to a code repository where it can be used and downloaded by anyone. iTrust over HTTP is already user friendly, and it can be integrated into existing decentralized systems to augment their functionality. For iTrust over SMS and iTrust over Wi-Fi Direct, we also plan to release the code as open source software on the Android platform. In particular, we plan to release the Android-based software directly to the Android Market (now called the Play Store), because it already can be used at some level by networking novices.

For enhancements and extra features, we plan to investigate other networks and technology that are used for secure information sharing. The data chunking

and distribution method of BitTorrent, when combined with information encoding transforms, might be useful for replicating data across the network. Disparate chunks might be distributed throughout the network and a requesting node might retrieve partial chunks and then reconstruct missing information by reverse transformations. For wireless communications in particular, securing the information transfer between devices becomes important. Here, existing technology used to encrypt information can be reused; in particular, Android support for the High-Definition Multimedia Interface (HDMI) can enable devices to create a virtual (and wireless) video link. The resulting HDMI wireless *tunnel* can then be used to share information securely using the High-bandwidth Digital Content Protection (HDCP) technology – a technology akin to Secure Shell (SSH) tunneling with encryption.

In any case, we hope that the research and novel insights of iTrust, in the area of decentralized information sharing, benefit not only the Internet at large but also the cellular telephony network and the many smaller mobile ad-hoc networks which continue to enrich the lives of individuals worldwide.

# Bibliography

[1] Android developers, Android 4.0 platform highlights. http://developer.android.com/sdk/android-4.0-highlights.html.

[2] A. Acero, N. Bernstein, R. Chambers, Y. C. Ju, X. Li, J. Odell, P. Nguyen, O. Scholz, and G. Zweig. Live search for mobile: Web services by voice on the cellphone. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5256–5259, Las Vegas, NV, March–April 2008.

[3] L. Adamic and E. Adar. How to search a social network. *Social Networks*, 27(3):187–203, July 2005.

[4] Wi-Fi Alliance. Wi-Fi Direct. http://www.wi-fi.org/discover-and-learn/wi-fi-direct.

[5] C. M. Badger, L. E. Moser, P. M. Melliar-Smith, I. Michel Lombera, and Y. T. Chuang. Declustering the iTrust search and retrieval network to increase trustworthiness. In *Proceedings of the 8th International Conference*

*on Web Information Systems and Technologies*, pages 312–322, Porto, Portugal, April 2012.

[6] R. Belen. *Detecting Disguised Missing Data*. PhD thesis, Middle East Technical University, 2009.

[7] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. P2P content search: Give the Web back to the people. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems*, pages 131–136, Santa Barbara, CA, February 2006.

[8] S. Bianchi, P. Felber, and M. Gradinariu. Content-based publish/subscribe using distributed r-trees. In *Proceedings of Euro-Par*, pages 537–548, Rennes, France, August 2007.

[9] A. Binzenhöfer and K. Leibnitz. Estimating churn in structured P2P networks. In *Proceedings of the 20th International Teletraffic Conference on Managing Traffic Performance in Converged Networks*, pages 630–641, Berlin, Germany, 2007.

[10] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the 7th International Conference on the World Wide Web*, pages 107–117, Brisbane, Australia, April 1998.

[11] T. D. Chandra, V. Hadzilacos, S. Toueg, and B. Charron-Bost. On the impossibility of group membership. In *Proceedings of the Fifteenth Annual*

*ACM Symposium on Principles of Distributed Computing*, pages 322–330, 1996.

[12] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P systems scalable. In *Proceedings of the ACM Special Interest Group on Data Communication Conference*, pages 407–418, Karlsruhe, Germany, August 2003.

[13] B. Chen, J.and Linn and L. Subramanian. SMS-based contextual Web search. In *Proceedings of the 2009 ACM SIGCOMM MobiHeld Workshop*, pages 19–24, Barcelona, Spain, August 2009.

[14] J. Chen, L. Subramanian, and E. Brewer. SMS-based Web search for low-end mobile devices. In *Proceedings of the 16th ACM MobiCom International Conference on Mobile Computing and Networking*, pages 125–136, Chicago, IL, September 2010.

[15] G. V. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: A comprehensive study. *ACM Computing Surveys*, 33(4):427–469, 2001.

[16] Y. T. Chuang, I. Michel Lombera, P. M. Melliar-Smith, and L. E. Moser. Detecting and defending against malicious attacks in the iTrust information retrieval network. In *Proceedings of the International Conference on Information Networking*, pages 263–268, Bali, Indonesia, February 2012.

[17] Y. T. Chuang, I. Michel Lombera, P. M. Melliar-Smith, and L. E. Moser. Protecting the iTrust information retrieval network against malicious attacks. *Journal of Computing Science and Engineering*, 6(3):179–192, September 2012.

[18] Y. T. Chuang, I. Michel Lombera, L. E. Moser, and P. M. Melliar-Smith. Trustworthy distributed search and retrieval over the Internet. In *Proceedings of the International Conference on Internet Computing*, pages 169–175, Las Vegas, NV, July 2011.

[19] Y. T. Chuang, P. M. Melliar-Smith, L. E. Moser, and I. Michel Lombera. Detection and defensive adaptation algorithms for protecting against malicious attacks in the iTrust information retrieval network. In preparation.

[20] Y. T. Chuang, P. M. Melliar-Smith, L. E. Moser, and I. Michel Lombera. Membership management for the iTrust information retrieval network. Submitted.

[21] Y. T. Chuang, P. M. Melliar-Smith, L. E. Moser, and I. Michel Lombera. Statistical inference and dynamic adaptation for the iTrust search and retrieval system. Submitted.

[22] K. Church, J. Neumann, M. Cherubini, and N. Oliver. SocialSearchBrowser: A novel mobile search and information discovery tool. In *Proceedings of the*

*15th International Conference on Intelligent User Interfaces*, pages 101–110, Hong Kong, China, February 2010.

[23] K. Church and B. Smyth. Understanding the intent behind mobile information needs. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, pages 247–256, Sanibel Island, FL, February 2009.

[24] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, Berkeley, CA, July 2001. Springer.

[25] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proceedings of the ACM Special Interest Group on Data Communications Conference*, pages 177–190, Pittsburgh, PA, August 2002.

[26] T. Condie, S. D. Kamvar, and H. Garcia-Molina. Adaptive peer-to-peer topologies. In *Proceedings of the 4th IEEE International Conference on Peer-to-Peer Computing*, pages 53–62, Zurich, Switzerland, August 2004.

[27] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. PlanetP: Using gossiping to build content addressable peer-to-peer information sharing communities. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, pages 236–246, Seattle, WA, June 2003.

[28] L. A. Cutillo, R. Molva, and M. Onen. Safebook: A distributed privacy preserving online social network. In *Proceedings of the IEEE World of Wireless, Mobile and Multimedia Networks Conference*, pages 1–3, Lucca, Italy, June 2011.

[29] W. Dai, L. E. Moser, P. M. Melliar-Smith, I. Michel Lombera, and Y. T. Chuang. The iTrust local reputation system for mobile ad-hoc networks. In *Proceedings of the 2013 International Conference on Wireless Networks*, Las Vegas, NV, July 2013, to appear.

[30] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 207–216, 2002.

[31] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermerrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, June 2003.

[32] B. M. Evans and E. H. Chi. Towards a model of understanding social search. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 485–494, San Diego, CA, November 2008.

[33] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume I. John Wiley & Sons, 1968.

[34] R. A. Ferreira, M. K. Ramanathan, A. Awan, A. Grama, and S. Jagannathan. Search with probabilistic guarantees in unstructured peer-to-peer networks. In *Proceedings of the 5th IEEE International Conference on Peer-to-Peer Computing*, pages 165–172, Konstanz, Germany, August 2005.

[35] P2P Foundation. Commotion. http://p2pfoundation.net/Commotion.

[36] A. Ganesh, A. M. Kermarrec, and L. Massoulié. SCAMP: Peer-to-peer lightweight membership service for large-scale group communication. *Networked Group Communication*, pages 44–55, 2001.

[37] A. Ganesh, A. M. Kermarrec, and L. Massoulie. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2):139–149, February 2003.

[38] P. Gardner-Stephen. The Serval project: Practical wireless ad-hoc mobile telecommunications. http://developer.servalproject.org/site/docs/2011/Serval_Introduction.html.

[39] Gnutella. http://en.wikipedia.org/wiki/Gnutella.

[40] Google SMS. http://www.google.com/sms.

[41] K. P. Gummadi, A. Mislove, and P. Druschel. Exploiting social networks for Internet search. In *Proceedings of the 5th ACM SIGCOMM Workshop on Hot Topics in Networks*, pages 79–84, Irvine, CA, November 2006.

[42] A. Gupta, O. D. Sahin, D. Agrawal, and A. El Abbadi. Meghdoot: Content-based publish/subscribe over P2P networks. In *Proceedings of the 5th ACM/IFIP/USENIX International Middleware Conference*, pages 254–273, Toronto, Canada, 2004.

[43] O. Heen, E. Le Merrer, C. Neumann, and S. Onno. Distributed and private group management. In *Proceedings of the 31st International Symposium on Reliable Distributed Systems*, pages 191–200, Irvine, CA, October 2012.

[44] H. Hu, B. Thai, and A. Seneviratne. Supporting mobile devices in Gnutella file sharing network with mobile agents. In *Proceedings of the 8th IEEE Symposium on Computers and Communications*, pages 1035–1040, Kemer-Antalya, Turkey, July 2003.

[45] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy preserving P2P data sharing with OneSwarm. In *Proceedings of the ACM SIGCOMM 2010 Special Interest Group on Data Communication Conference*, pages 111–122, New Delhi, India, September 2010.

[46] G. P. Jesi, D. Hales, and M. Van Steen. Identifying malicious peers before it's too late: A decentralized secure peer sampling service. In *Proceedings of the 1st International Conference on Self-Adaptive and Self-Organizing Systems*, pages 237–246, Boston, MA, July 2007.

[47] M. Kamvar and S. Baluja. A large scale study of wireless search behavior: Google mobile search. In *Proceedings of the 24th ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 701–709, Montreal, Quebec, Canada, April 2006.

[48] M. Kamvar, M. Kellar, R. Patel, and Y. Xu. Computers and iPhones and mobile phones, oh my!: A log-based comparison of search users on different devices. In *Proceedings of the 18th International Conference on the World Wide Web*, pages 801–810, Madrid, Spain, April 2009.

[49] J. Larsen, K. W. Axhausen, and J. Urry. Geographies of social networks: Meetings, travel and communications. *Mobilities*, 1(2):261–283, July 2006.

[50] C. Leng, W. W. Terpstra, B. Kemme, W. Stannat, and A. P. Buchmann. Maintaining replicas in unstructured P2P systems. In *Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies*, page 19, Madrid, Spain, December 2008.

[51] J. Li, B. Loo, J. Hellerstein, F. Kaashoek, D. Karger, and R. Morris. On the feasibility of peer-to-peer Web indexing and search. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems, Lecture Notes in Computer Science 1735*, pages 207–215, 2003.

[52] J. Liang and K. Nahrstedt. RandPeer: Membership management for QoS sensitive peer to peer applications. In *Technical Report UIUCDCS-R-*

*2005-2576,* Department of Computer Science, University of Illinois, Urbana Champaign, May 2005.

[53] C. Lindemann and O. P. Waldhorst. A distributed search service for peer-to-peer file sharing in mobile applications. In *Proceedings of the 2nd International Conference on Peer-to-Peer Computing*, pages 73–80, Linkoping, Sweden, September 2002.

[54] H. Liu, X. Liu, W. Song, and W. Wen. An age-based membership protocol against strong churn in unstructured P2P networks. In *Proceedings of the 2011 International Conference on Network Computing and Information Security*, volume 2, pages 195–200, Guilin, China, May 2011.

[55] I. Michel Lombera, Y. T. Chuang, P. M. Melliar-Smith, and L. E. Moser. Trustworthy distribution and retrieval of information over HTTP and the Internet. In *Proceedings of the 3rd International Conference on the Evolving Internet*, pages 7–13, Luxembourg City, Luxembourg, June 2011.

[56] I. Michel Lombera, Y. T. Chuang, L. E. Moser, and P. M. Melliar-Smith. Decentralized mobile search and retrieval using SMS and HTTP to support social change. In *Proceedings of the 3rd International Conference on Mobile Computing, Applications, and Services*, pages 152–171, Los Angeles, CA, October 2011.

[57] I. Michel Lombera, L. E. Moser, P. M. Melliar-Smith, and Y. T. Chuang. Mobile decentralized search and retrieval using SMS and HTTP. *ACM Mobile Networks and Applications Journal*, 18(1):22–41, 2013.

[58] I. Michel Lombera, L. E. Moser, P. M. Melliar-Smith, and Y. T. Chuang. Trustworthy mobile ad-hoc wireless networks over Wi-Fi Direct. In preparation.

[59] I. Michel Lombera, L. E. Moser, P. M. Melliar-Smith, and Y. T. Chuang. Mobile ad-hoc search and retrieval in the iTrust over Wi-Fi Direct Network. In *Proceedings of the Ninth International Conference on Wireless and Mobile Communications*, Nice, France, July 2013, to appear.

[60] I. Michel Lombera, L. E. Moser, P. M. Melliar-Smith, and Y. T. Chuang. A mobile peer-to-peer search and retrieval service for social networks. In *Proceedings of the IEEE 1st International Conference on Mobile Services*, pages 72–79, Honolulu, HI, June 2012.

[61] I. Michel Lombera, L. E. Moser, P. M. Melliar-Smith, and Y. T. Chuang. Peer management for iTrust over Wi-Fi Direct. In *Proceedings of the 2013 International Symposium on Wireless Personal Multimedia Communications*, Atlantic City, NJ, June 2013, to appear.

[62] I. Michel Lombera, L. E. Moser, P. M. Melliar-Smith, and Y. T. Chuang. Decentralized search and retrieval for mobile networks using SMS. In *Pro-*

*ceedings of the IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 134–141, Barcelona, Spain, October 2012.

[63] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th International Conference on Supercomputing*, pages 84–95, Baltimore, MD, June 2002.

[64] S. Maki, T. Aura, and M. Hietalahti. Robust membership management for ad-hoc groups. In *Proceedings of the 5th Nordic Workshop on Secure IT Systems*, pages 23–41, Reykjavik, Iceland, 2000.

[65] P. M. Melliar-Smith, L. E. Moser, I. Michel Lombera, and Y. T. Chuang. iTrust: Trustworthy information publication, search and retrieval. In *Proceedings of the 13th International Conference on Distributed Computing and Networking*, pages 351–366, Hong Kong, China, January 2012.

[66] X. Meng, P. Zerfos, V. Samanta, S. H. Y. Wong, and S. Lu. Analysis of the reliability of a nationwide short message service. In *Proceedings of the 26th Annual IEEE Conference on Computer Communications*, pages 1811–1819, Anchorage, AK, May 2007.

[67] P. Meroni, E. Pagani, G. P. Rossi, and L. Valerio. An opportunistic platform for Android-based mobile devices. In *Proceedings of the Second International*

*Workshop on Mobile Opportunistic Networking*, pages 191–193, Pisa, Italy, February 2010.

[68] J. Mischke and B. Stiller. A methodology for the design of distributed search in P2P middleware. *IEEE Network*, 18(1):30–37, 2004.

[69] AOL Mobile. http://www.aolmobile.com.

[70] Windows Live Mobile. http://home.mobile.live.com.

[71] R. Morselli, B. Bhattacharjee, A. Srinivasan, and M. A. Marsh. Efficient lookup on unstructured topologies. In *Proceedings of the 24th ACM Symposium on Principles of Distributed Computing*, pages 77–86, Las Vegas, NV, July 2005.

[72] M. Motani, V. Srinivasan, and P. S. Nuggehalli. Peoplenet: Engineering a wireless virtual social network. In *Proceedings of the 11th ACM International Conference on Mobile Computing and Networking*, pages 243–257, Cologne, Germany, August 2005.

[73] R. Motta and J. Pasquale. Wireless P2P: Problem or opportunity. In *Proceedings of the Second IARIA Conference on Advances in P2P Systems*, pages 32–37, Florence, Italy, October 2010.

[74] Yahoo! OneSearch. http://mobile.yahoo.com/onesearch.

[75] M. Papadopouli and H. Schulzrinne. Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing*, pages 117–127, Long Beach, CA, 2001.

[76] B. Peng, L. E. Moser, P. M. Melliar-Smith, Y. T. Chuang, and I. Michel Lombera. A distributed ranking algorithm for the iTrust information search and retrieval system. In *Proceedings of the 9th International Conference on Web Information Systems and Technologies*, pages 199–208, Aachen, Germany, May 2013.

[77] C. Raval and S. Hegde. Ant-CAMP: Ant based congestion adaptive multipath routing protocol for wireless networks. In *Proceedings of the International Conference on Emerging Trends in Networks and Computer Communications*, pages 463–468, 2011.

[78] J. Risson and T Moors. Survey of research towards robust peer-to-peer networks: Search methods. In *Technical Report UNSW-EE-P2P-1-1*, pages RFC 4981, http://tools.ietf.org/html/rfc4981, University of New South Wales, September 2007.

[79] D. R. Sandler and D. S. Wallach. Birds of a fethr: Open, decentralized micropublishing. In *Proceedings of the 8th International Workshop on Peer-to-Peer Systems*, page 1, Boston, MA, April 2009.

[80] N. Sarshar, P. O. Boykin, and V. P. Roychowdhury. Percolation search in power law networks: Making unstructured peer-to-peer networks scalable. In *Proceedings of the 4th International Conference on Peer-to-Peer Computing*, pages 2–9, Zurich, Switzerland, August 2004.

[81] A. Schiper and S. Toueg. From set membership to group membership: A separation of concerns. *IEEE Transactions on Dependable and Secure Computing*, 3(1):2–12, 2006.

[82] R. Schusteritsch, S. Rao, and K. Rodden. Mobile search with text messages: Designing the user experience for Google SMS. In *Proceedings of the 23rd ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1777–1780, Portland, OR, April 2005.

[83] Smozzy. http://www.smozzy.com.

[84] T. Sohn, K. A. Li, W. G. Griswold, and J. D. Hollan. A diary study of mobile information needs. In *Proceedings of the 26th ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 433–442, Florence, Italy, April 2008.

[85] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technolo-*

*gies, Architectures and Protocols for Computer Communications*, pages 149–160, San Diego, CA, August 2001.

[86] W. W. Terpstra, J. Kangasharju, C. Leng, and A. P. Buchmann. Bubblestorm: Resilient, probabilistic, and exhaustive peer-to-peer search. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, pages 49–60, Kyoto, Japan, August 2007.

[87] J. Thomas and J. Robble. Off grid communication with Android: Meshing the mobile world. In *Proceedings of the IEEE Conference on Technologies for Homeland Security*, pages 401–405, Waltham, MA, November 2012.

[88] P. Tiago, N. Kotiainen, M. Vapa, H. Kokkinen, and J. K. Nurminen. Mobile search – social network search using mobile devices. In *Proceedings of the 5th IEEE Consumer Communications and Networking Conference*, pages 1201–1205, Las Vegas, NV, January 2008.

[89] D. A. Tran and C. Pham. Enabling content-based publish/subscribe services in cooperative P2P networks. *Computer Networks*, 52(11):1739–1749, August 2010.

[90] G. Tsirulnik. Global SMS traffic to reach 8.7 trillion by 2015: Study, February 3, 2011. http://www.mobilecommercedaily.com/2011/02/03/global-sms-traffic-to-reach-8-7-trillion-by-2015.

[91] D. Tsoumakos and N. Roussopoulos. A comparison of peer-to-peer search methods. In *Proceedings of the Sixth International Workshop on the Web and Databases*, pages 61–66, San Diego, CA, June 2003.

[92] D. Tsoumakos and N. Roussopoulos. Adaptive probabilistic search for peer-to-peer networks. In *Proceedings of the IEEE 3rd International Conference on Peer-to-Peer Computing*, pages 102–109, Linkoping, Sweden, September 2003.

[93] S. Voulgaris, D. Gavidia, and M. Van Steen. CYCLON: Inexpensive membership management for unstructured P2P overlays. *Journal of Network and Systems Management*, 13(2):197–217, June 2005.

[94] Y. Wang, L. Galanis, and D. J. de Witt. Galanx: An efficient peer-to-peer search engine system. In *Technical Report*. University of Wisconsin, Madison, 2003.

[95] D. J. Watts, P. S. Dodds, and M. E. J. Newman. Identity and search in social networks. *Science*, 296(5571):1302–1305, May 2002.

[96] B. Wong and S. Guha. Quasar: A probabilistic publish-subscribe system for social networks. In *Proceedings of the 7th International Workshop on Peer-to-Peer Systems*, pages 2–8, Tampa Bay, FL, February 2008.

[97] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems*, pages 5–14, Vienna, Austria, July 2002.

[98] J. Yang, Y. Zhong, and S. Zhang. An efficient interest-group-based search mechanism in unstructured peer-to-peer networks. In *Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing*, pages 247–252, Shanghai, China, October 2003.

[99] D. Zage, C. Livadas, and E. M. Schooler. A network-aware distributed membership protocol for collaborative defense. In *Proceedings of the International Conference on Computational Science and Engineering*, volume 4, pages 1123–1130, 2009.

[100] M. Zhong and K. Shen. Popularity-biased random walks for peer-to-peer search under the square-root principle. In *Proceedings of the International Workshop on Peer-to-Peer Systems, Lecture Notes in Computer Science 4490*, pages 877–880. Springer, 2007.

[101] R. Zhou and K. Hwang. Gossip-based reputation aggregation for unstructured peer-to-peer networks. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium Conference*, pages 26–30, Long Beach, CA, March 2007.