Mobile Ad-Hoc Search and Retrieval in the iTrust over Wi-Fi Direct Network

Isaí Michel Lombera, Louise E. Moser, Peter M. Melliar-Smith, Yung-Ting Chuang
Department of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, USA
imichel@ece.ucsb.edu,moser@ece.ucsb.edu,pmms@ece.ucsb.edu,ytchuang@ece.ucsb.edu

***Abstract*—iTrust over Wi-Fi Direct is a peer-to-peer information publication, search and retrieval system for mobile ad-hoc networks. In this paper, we present the fundamental design of iTrust over Wi-Fi Direct. Next, we describe the iTrust over Wi-Fi Direct API and components as implemented on the Android platform for mobile devices, and show how user applications can easily interface with the API to gain P2P functionality. Then, we present the iTrust over Wi-Fi Direct networking model, and the interactions between the Android and Linux stacks. iTrust over Wi-Fi Direct addresses the need for sharing of personal information and simultaneously prevents a third party from censoring information and preventing dissemination of information. It enables users with Wi-Fi Direct enabled mobile devices to publish, search for, and retrieve information.**

***Keywords*-Android; publication, search and retrieval; mobile ad-hoc network; peer-to-peer network; Wi-Fi Direct**

I. Introduction

In our increasingly wireless and mobile world, the typical mobile phone user has come to expect ubiquitous access to public and private information. The traditional information search engines, such as those of 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, on 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 the users 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 by 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 the 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 at 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. 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 [20] is a relatively new wireless technology, based on IEEE 802.11, that enables devices to form a peer-to-peer (P2P) network without the need for a third intermediary device, such as an access point. Wi-Fi P2P is the previous name of Wi-Fi Direct; we use both terms interchangeably in this paper.

In the rest of this paper, we describe the fundamental design of iTrust over Wi-Fi Direct (Section II), the application programming interface (API) and components as implemented on the Android platform (Section III), the associated networking model (Section IV), related work (Section V), and conclusions and future work (Section VI).

II. iTrust Fundamental Design

The iTrust over Wi-Fi Direct network consists of peers that form a mobile ad-hoc network. Multiple iTrust over Wi-Fi Direct networks may exist simultaneously, and a peer may join any such network over time. Peers in the same network are said to be in the same *membership*, although the peers do not *all* need to be within range of each other.

Figure 1 illustrates how information is published, searched for, and retrieved in the iTrust over Wi-Fi Direct network. Any peer with information to share (which we call a *source*) generates metadata describing that information and distributes that metadata to a subset of the membership chosen at random (1). A peer requesting information (which we call a *requester*) distributes a query to a subset of the membership chosen at random (2). In the distribution of both the metadata and the query, a peer that receives the message may *relay* the message to yet another subset of the membership chosen at random. Message flooding is prevented, but is not addressed here. When a peer finds a match between the metadata it holds and a query it receives, we say that an *encounter* occurs (3). The peer with the match sends a message to the requester, which identifies the source holding the desired information (4). The requester then directly fetches the information from the source (5).

In previous work, we described iTrust over HTTP [9] and iTrust over SMS [10]. We have established that the probability of a match is high even if the metadata and the queries are distributed to relatively few peers [7]. Moreover, we have developed mechanisms that prevent malicious peers from censoring information or disrupting the dissemination of information [2].

III. iTrust over Wi-Fi Direct

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. The previously implemented 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. 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 2 illustrates the iTrust over Wi-Fi Direct API and components, their relationships to both user/programmer applications and the underlying Android/Linux mobile platform. We discuss in detail each of the blocks in the diagram and their relationships and interactions below. The center iTrust over Wi-Fi Direct blocks are the most pertinent, but we also discuss the two surrounding user and operating system blocks.

A. Application

The app block at the left of Figure 2 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, which decouples 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 [10], to demonstrate the easy way in which any application can add the publication, search and retrieval functionality of iTrust. In particular, we showed how iTrust may be added to existing chat applications such as AIM, Jabber, XMPP, etc. iTrust over Wi-Fi Direct retains this same functionality; 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 sharing, music sharing, text document collaboration, etc.

B. Signal Parser

The signal parser in iTrust over Wi-Fi Direct is similar to the signal parser in iTrust over SMS, 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,

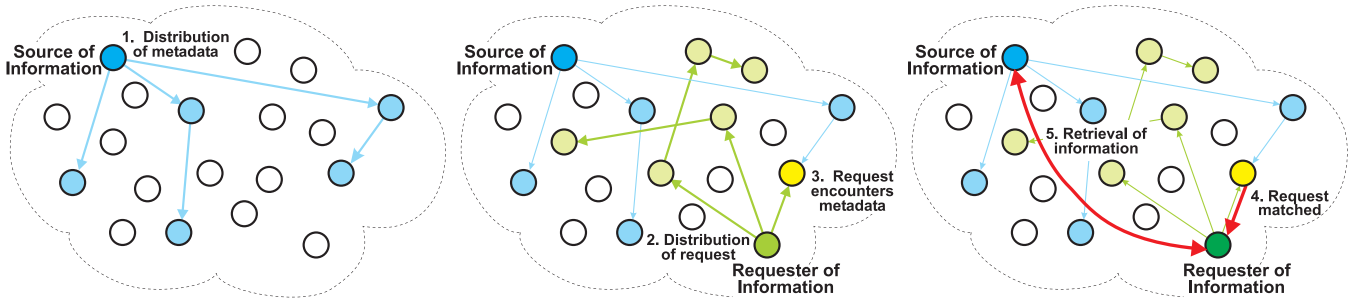

Figure 1. Information publication, search and retrieval in the iTrust over Wi-Fi Direct network.

and retrieving information. The peer management code is quite extensive and is described in detail elsewhere [11] and, thus, is not described here; 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.

C. 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, 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 state, 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 IV). 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 Wi-Fi P2P 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 III-E.

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.

D. Database Adapter

The database (DB) adapter in iTrust over Wi-Fi Direct is structurally similar to that in iTrust over SMS, 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 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 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.

E. 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 is separate 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 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 III-G) and acts as an intermediary between the Inbox thread and the signal parser (all 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 III-H).

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. Again, the peer management details are outside the scope of this paper; however, it is necessary to outline the primary steps required to transmit information between peers in the network.

First, the Wi-Fi P2P service component must check the device's system settings and request permission to control the Wi-Fi P2P hardware adapter. If 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 if the peer is deemed available for a connection (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, if 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) device invitations; thus, the application and the individual using the device do not have to confirm every connection request (including reconnections after accidental disconnects). Fourth, the device begins device negotiation to manage peers within the iTrust over Wi-Fi Direct 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.

F. 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 listen to and monitor 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 the 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* has 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, thus, is outside the scope of this paper; however, it does have an important role for the other components. 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.

G. 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.

H. 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.

I. Android/Linux

The Android/Linux block in Figure 2 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; e.g., the Inbox and Outbox threads use Linux and not Android to setup sockets

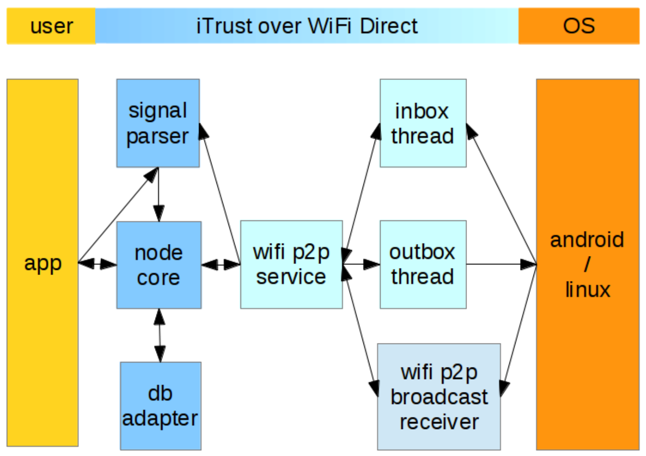

Figure 2. The iTrust over Wi-Fi Direct API and components.

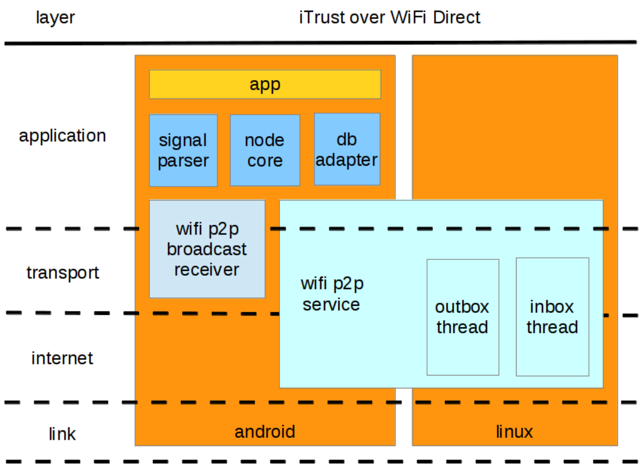

Figure 3. The iTrust over Wi-Fi Direct networking model.

and transfer information. However, unlike traditional Linux systems where the kernel dominates network functionality, Android plays an important role in controlling the Wi-Fi P2P network adapter (in addition to iTrust logic non-specific to network-related functions). We elaborate this interaction between the Android user space and the Linux kernel in creating and using the Wi-Fi Direct connection below.

IV. 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 3 illustrates the iTrust over Wi-Fi Direct networking model by rearranging the components of Figure 2 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 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 layers of the Internet protocol stack: 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 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 space code). Second, most components individually interact within only a single layer, but there are several components that span multiple layers, e.g., the signal parser is entirely within the Application layer but 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 the Linux network stacks. Finally, although the app is shown in the Android network stack, the user/programmer may choose 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); the programmer may of course add more functionality to the application.

A. Link Layer

The Link layer does not contain any iTrust over Wi-Fi Direct components. However, it is necessary to understand the interactions 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 which, for the most part, is 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 mobile device, to which the Linux kernel is ported, to gain network access easily.

However, Wi-Fi Direct is different, and Android plays a much more significant role in the Link layer. Wi-Fi Direct requires 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 namespaces. For example, all Wi-Fi Direct code is within the *android.net.wifi.p2p.** namespace, instead of the standard Java namespaces 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 platform design decision or whether the nature of the protocol does not allow for tight kernel integration.

The remaining sections have more examples of the individualistic nature of Wi-Fi Direct.

B. *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 in the Internet layer; 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 peers 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 a peer by its 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 and Inbox threads have access to the Linux network stack in the Internet layer (the functionality is very similar to traditional client/server sockets). However, most of the socket functionality is in the Transport layer, as discussed in the next section.

C. *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 can distinguish itself from the other peers, and can share 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 IV-B, 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 [11].

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 IV-D) 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 and, thus, 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 is passed down from the Application layer to the Transport layer (still within the Wi-Fi P2P service component), then transfers from the Android stack to the Linux stack, and finally 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.

D. 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 in the Application layer on the Android stack.

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 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 peer management, such as how peer IP addresses are assigned, how peers join the membership, and how connections are repaired, can be found in [11].

V. Related Work

Several researchers [12], [16], [19] have provided methodologies, surveys and comparisons of distributed search for peer-to-peer networks. The structured approach requires the nodes to be organized in an overlay network based on distributed hash tables, trees, rings, etc. The unstructured approach uses randomization, and requires the nodes to find each other by exchanging messages. The iTrust system uses the unstructured approach, which is particularly appropriate for mobile ad-hoc networks.

The Commotion Wireless project [15] aims to ensure that communication cannot be controlled or cut off by authoritarian regimes, which is also one of the goals of the iTrust project. Their device-as-an-infrastructure distributed communication platform integrates Wi-Fi enabled mobile phones, computers and other personal devices to create a metro-scale network that supports local peer-to-peer communication, as well as local-to-Internet communication.

Thomas and Robble [17] have created a mobile ad-hoc network for disaster and emergency relief, using the Wi-Fi chips in Android phones, allowing them to connect directly without using cellular networks, like iTrust over Wi-Fi Direct. Their Smart Phone Ad-Hoc Networks (SPAN) project reconfigures the onboard Wi-Fi chip of a smartphone to act as a Wi-Fi router to nearby similarly configured smartphones. SPAN intercepts communication at the Global Handset Proxy, so that typical applications, such as e-mail, Twitter, etc., still work. In contrast, our implementation of iTrust for mobile ad-hoc networks uses Wi-Fi Direct, which Android now supports.

The Serval project [3] 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 phones. At the time, Android phones did not support Wi-Fi Direct, 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. [8] describe an opportunistic platform for Android-based devices using Wi-Fi in a mobile ad-hoc network. Their platform is intended to address concerns of scalability, flexibility and bandwidth in cellular networks by supporting local peer-to-peer communication between nodes. It enables peers to query for information and receive responses locally and, thus, to save network bandwidth, particularly when that information is large.

Motta and Pasquale [13] describe a JXTA middleware architecture for peer-to-peer networks, which exploits the features of mobile devices and optimizes mobile resources. They apply the JXTA middleware to a search infrastructure for structured peer-to-peer networks that uses resource indexing based on distributed hash tables. The iTrust over Wi-Fi Direct system for publication, search and retrieval uses an unstructured approach, which is more appropriate for mobile ad-hoc networks.

The Mobile Agent Peer-To-Peer (MAP2P) system [5] supports mobile devices in a Gnutella [4] file-sharing network using mobile agents. A mobile agent attaches itself to the peer-to-peer network, and acts as a proxy for the mobile device. The iTrust system has a lower message cost than Gnutella and, thus, a lower message cost than MAP2P.

The 7DS system [14] supports information sharing among peers in a mobile ad-hoc network. It uses multicasting of queries together with a multi-hop flooding algorithm. 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 [7]. Moreover, iTrust does not relay metadata or requests that a node has seen previously.

The Distributed Mobile Search Service [6] broadcasts query results locally and forwards them over several hops. It is based on a distributed index, stored in a local index cache on each mobile device, that contains keywords and corresponding document identifiers. iTrust over Wi-Fi Direct likewise maintains a distributed index, with metadata and corresponding node addresses and resource ids stored on the mobile devices. However, iTrust distributes metadata and the corresponding node addresses and resource ids first, rather than on receipt of the query results.

Tiago et al. [18] describe a system for mobile search in social networks based on the Drupal content site management system. Their system is fully distributed, is based on the network of social links formed from the mobile phone's address book, and exploits the independence of nodes. iTrust over Wi-Fi Direct is not based on the links provided by the mobile phone's address book but, rather, on the nodes within a node's neighborhood.

VI. Conclusion and Future Work

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. In this paper, we have described the iTrust over Wi-Fi Direct API and components as implemented on the Android platform for mobile devices. 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. In the future, we plan to experiment with the iTrust over Wi-Fi Direct implementation and to evaluate its performance. We also plan to add an easy-to-use graphical user interface and to release an app that is useable even by computer novices.

Acknowledgment

This research was supported in part by U.S. National Science Foundation grant number NSF CNS 10-16193.

References

[1] Android Developers, Android 4.0 Platform Highlights, http://developer.android.com/sdk/android-4.0-highlights.html [retrieved: May 15, 2013]

[2] 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), Sep. 2012, pp. 179–192.

[3] P. Gardner-Stephen, "The Serval project: Practical wireless ad-hoc mobile telecommunications," http://developer.serval project.org/site/docs/2011/Serval_Introduction.html [retrieved: May 15, 2013]

[4] Gnutella, http://www.gnu.org/philosophy/gnutella.html [retrieved: May 15, 2013]

[5] H. Hu, B. Thai, and A. Seneviratne, "Supporting mobile devices in the Gnutella file sharing network with mobile agents," Proc. 8th IEEE Symposium on Computers and Communications, Kemer-Antalya, Turkey, Jul. 2003, pp. 1035–1040.

[6] C. Lindemann and O. P. Waldhorst, "A distributed search service for peer-to-peer file sharing in mobile applications," Proc. Second International Conference on Peer-to-Peer Computing, Linkoping, Sweden, Sep. 2002, pp. 73–80.

[7] P. M. Melliar-Smith, L. E. Moser, I. Michel Lombera, and Y. T. Chuang, "iTrust: Trustworthy information publication, search and retrieval," Proc. 13th International Conference on Distributed Computing and Networking, Hong Kong, China, Jan. 2012, LNCS 7129, Springer, pp. 351–366.

[8] P. Meroni, E. Pagani, G. P. Rossi, and L. Valerio, "An opportunistic platform for Android-based mobile devices," Proc. Second International Workshop on Mobile Opportunistic Networking, Pisa, Italy, Feb. 2010, pp. 191–193.

[9] 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," Proc. Third IARIA International Conference on the Evolving Internet, Luxembourg City, Luxembourg, Jun. 2011, pp. 7–13.

[10] I. Michel Lombera, L. E. Moser, P. M. Melliar-Smith, and Y. T. Chuang, "Decentralized search and retrieval for mobile networks using SMS," Proc. IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications, Barcelona, Spain, Oct. 2012, pp. 134–141.

[11] I. Michel Lombera, L. E. Moser, P. M. Melliar-Smith, and Y. T. Chuang, "Peer management for iTrust over Wi-Fi Direct," Proc. International Symposium on Wireless Personal Multimedia Communications, Atlantic City, NJ, Jun. 2013.

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

[13] R. Motta and J. Pasquale, "Wireless P2P: Problem or opportunity," Proc. Second IARIA Conference on Advances in P2P Systems, Florence, Italy, Oct. 2010, pp. 32–37.

[14] M. Papadopouli and H. Schulzrinne, "Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices," Proc. ACM Symposium on Mobile Ad Hoc Networking and Computing, Long Beach, CA, 2001, pp. 117–127.

[15] P2P Foundation, Commotion, http://p2pfoundation.net/ Commotion [retrieved: May 15, 2013]

[16] J. Risson and T. Moors, "Survey of research towards robust peer-to-peer networks: Search methods," Tech. Rep. UNSW-EE-P2P-1-1, Univ. New South Wales, Sep. 2007, RFC 4981, http://tools.ietf.org/html/rfc4981 [retrieved: May 15, 2013]

[17] J. Thomas and J. Robble, "Off grid communication with Android: Meshing the mobile world," Proc. IEEE Conference on Technologies for Homeland Security, Waltham, MA, Nov. 2012, pp. 401–405.

[18] P. Tiago, N. Kotiainen, M. Vapa, H. Kokkinen, and J. K. Nurminen, "Mobile search – Social network search using mobile devices," Proc. 5th IEEE Consumer Communications and Networking Conference, Las Vegas, NV, Jan. 2008, pp. 1201–1205.

[19] D. Tsoumakos and N. Roussopoulos, "A comparison of peer-to-peer search methods," Proc. Sixth International Workshop on the Web and Databases, San Diego, CA, Jun. 2003, pp. 61–66.

[20] Wi-Fi Alliance, Wi-Fi Direct, http://www.wi-fi.org/discover-and-learn/wi-fi-direct [retrieved: May 15, 2013]