



Research Center
NRC-TR-2006-002

**“iTouch: RFID Middleware for Boosting Connectivity
&
Intuitive User Interaction in Smart Spaces”**

Zoe Antoniou and Srikant Varadan
Nokia Research Center Cambridge
<http://research.nokia.com>
May 9, 2006

Abstract:

The fusion of interaction and discovery technologies can facilitate easy, intuitive and impromptu access to networked entities by users without the need for significant administrative overhead. Due to mobility, complex connectivity and small user interfaces, traditional service discovery methods fail to meet the demands placed by such environments. This report presents a proof-of-concept design and implementation on how NFC-enhanced mobile devices can enable intuitive service discovery by non-expert users in smart environments through simple gestures such as touch. With this new paradigm, physical space becomes an extension of the traditional GUI. An extensible RFID tag record is presented that can be utilised in a variety of scenarios. The technical realisation uses the UPnP framework for service discovery. The step-by-step process of service discovery and user interaction is described for three demo use cases.

Index Terms:

intuitive user interaction, middleware, pervasive computing, NFC, RFID, smart spaces, UPnP

TABLE OF CONTENTS

Abstract:	1
1. Introduction	3
1.1 Problem Statement.....	3
1.2 Report Organization.....	4
1.3 Prototype Setup	5
2. Why NFC Technology?.....	5
3. Related work.....	6
4. The touch paradigm	7
4.1 Context Awareness.....	8
4.2 User Interaction	9
5. UPnP framework.....	10
5.1 Overview	10
5.2 SSDP Presence Announcement for the Smart Device	10
6. Tag design	11
7. Architecture	14
7.1 Overview	14
7.2 Prototype Design.....	15
7.3 Service Profiles	17
7.4 iTouch API	18
7.4.1 Constructor.....	18
7.4.2 Registration.....	18
7.4.3 Reading from RFID tag.....	20
7.4.4 Receiving data upon RFID read	20
7.4.5 Canceling a Read request.....	22
7.4.6 Arming the RFID hardware	22
7.4.7 NTIP Server Shutdown.....	22
8. User interaction tools	23
8.1 Starting iTouch.....	23
8.2 iTouch hot button	24
8.3 Writing to a tag	25
8.4 Client applications.....	27
8.5 Use case I: Touch to connect to WLAN NAP	27
8.6 Use case II: Touch to connect to WLAN hotspot & launch URL	29
8.7 Use case III: Touch to discover UPnP devices - UPnP printing.....	30
8.8 Ending the demo.....	32
References	33

1. INTRODUCTION

1.1 Problem Statement

The increasing penetration of short-range wireless access mobile devices and the introduction of networked environments are giving rise to new user interaction and service models. These models emerge both as home and workplace are gradually being transformed into smart spaces populated with a diverse set of devices and applications all waiting to be accessed and used. The trend for such spaces is to provide intelligent, *I-centric* service architectures. *I-centric* communication considers the human behaviour as a starting point to adapt the activities of communication systems to it rather than being unaware of user needs and situations. Discovery and interaction technologies can be combined to create a consistent user experience that significantly reduces the administrative and configuration overhead.

Many technologies were originally developed for zero-configuration networks in controlled and familiar environments (e.g. home, office). Recent research has focused on the use of these technologies in mobile computing scenarios. Some well-known service discovery protocols are Service Location Protocol (SLP, by IETF), Jini (a Java-based approach by Sun), Universal Plug and Play (UPnP, by Microsoft) and Bluetooth Service Discovery Protocol (SDP). Currently, these technologies lack the ability to ensure independence from the networking layer, interoperability between devices of different manufacturers, and cannot provide a complete solution for the easy, secure and intuitive interaction between non-expert users and services:

- They require manual intervention and are typically deployed in traditional network topologies where trained users work through powerful hosts [8], [9].
- Discovering services often implies performing multicast searches over wireless interfaces. This can lead to discovery latency issues and, potentially result in long lists of available services and devices.
- Even in cases where the user has identified the type or friendly name for a service, a search may still be necessary in order to launch it.
- The user is required to setup suitable network access in order to reach the desired service.

The successful deployment of such pervasive computing environments will rely on the promptness of the self-configuring procedures and the ease of use of the discovery process for the average user. Meeting these requirements poses a number of logistical and technical challenges. Due to mobility, more complex connectivity, small user interfaces (UIs) and limited text input, issues such as mobile service configuration, discovery and activation will have to take a different path to success than the desktop devices. In this setting, object tagging is a powerful concept for grounding immaterial mobile services in the real world.

Radio Frequency Identification (RFID) has received tremendous attention lately as a promising technology for smart space object tagging and an important enabler in the mobile terminal business. RFID tags based on the Near Field Communication (NFC) technology have a modifiable state and a short operating range. These tags can be interpreted by middleware and relevant applications running on the mobile device itself. This is different from Electronic Product Code (EPC) technology. EPC tags are typically simple object IDs with a reading range of the order of meters, and EPC readers need to access a backbone infrastructure in order to interpret them semantically.

iTouch is RFID-enhanced middleware and applications for mobile devices that enable intuitive user interaction in smart spaces, easy service discovery and face-to-face sharing. The intent is for the mobile phone to play a central role in boosting the usability of services and enriching the user experience, both in the consumer and enterprise domains. iTouch provides a layer of middleware between the proximity connectivity technologies (RFID, Infrared, Camera, Laser, etc) and user-domain applications and service discovery engines.

This report proposes a method that extends the concept of user-to-device interaction beyond the limitations of the traditional GUI interface and provides easy service discovery and launching. The proposed approach enables a non-expert user to discover and use the desired service through a one-step process by touching dedicated service access points, thus, eliminating the need for multiple manual steps and multicast searches. In the new interaction paradigm, user input can be received through touch in the same way as it is received through mouse clicks and menu selections. As a result, physical space becomes an extension to the mobile device GUI and physical objects can be mapped to virtual objects. Discovery does not require detailed knowledge of local network configuration or service initiation processes. An object may be selected, clicked upon or dropped on different application icons depending on the user's intentions and different actions are invoked. For instance, if a user selects a picture on mobile phone X and then touches mobile phone Y, it is interpreted as the user's intent to share the image. If a user selects a picture on a mobile phone and then touches a printer, it is interpreted as the user's intent to print it.

1.2 Report Organization

Section 2 briefly introduces the NFC technology. Section 3 gives an overview of related work. Section 4 presents the *touch* paradigm as an interaction modality between handsets and smart spaces. The UPnP framework is summarised in section 5. The tag design for this experimental prototype is presented in section 6. Section 7 describes the system architecture and the exposed API. Section 8 presents the user interaction tools and the implemented demo use cases.

1.3 Prototype Setup

This proof-of-concept implementation is developed for a Nokia 9500 Communicator handset in Symbian OS (Figure 1). The NFC interface is provided through an internally developed prototype, which is connected to the phone through a USB cable. The NFC interface allows reading from and writing to Mifare Ultralight (48 bytes) and Mifare Standard tags (1Kbytes/4Kbytes).



Figure 1: iTouch prototype.

2. WHY NFC TECHNOLOGY?

Though RFID technology has, traditionally, been used in industrial and logistics-related applications, it is well suited for automatic service discovery and configuration. User applications have been limited, focusing on payments at points-of-sale and contactless public transport tickets. Nokia, Philips and Sony recently established the Near Field Communication (NFC) Forum to promote the use of contact-less short-range technologies in a variety of consumer applications [2]. Bringing two NFC devices together should engage the wireless devices' interfaces so that they can (a) exchange data purely over NFC or (b) exchange configuration parameters in order to link in a peer-to-peer network over another wireless medium. A proximity RFID technology operating at the unregulated band of 13.56 MHz with

an operating distance of 0-10cm has been chosen. Reasons for this include compatibility with existing payment and ticketing solutions, maturity and availability of technology and parameters, such as size and power consumption. Due to the very short operating range the usage paradigm of such NFC systems resembles a touch gesture. Technology miniaturization makes it feasible to integrate NFC functionality into consumer products and mobile devices and it is well positioned to revolutionise the user experience.

3. RELATED WORK

This section summarises selected publications in the area of RFID-enhanced service discovery and how they relate to iTouch. One approach has been to explore the integration of RFID technology with Web services. In [16] the authors present a presence-aware infrastructure in order to implement a committee meeting scenario. RFID tags are attached to the devices of the meeting attendees and they are detected when they enter the meeting room. The information on the tags is used as input to the presence manager, the authentication and personalization services and the persistent repository, all of which create a collaborative environment. In [18] the authors analyse example use cases such as Smart Tool Box, Smart Medicine Cabinet, Smart Agenda, RFID Chef and Smart Playing Cards in order to extract generic design concepts. They present two prototype frameworks based on Jini and Web services. Tags are attached to physical objects which hold pointers to their virtual counterparts. The prototype architecture uses RFID interfaces to periodically scan the surroundings for tagged objects. When a tagged object is detected, it is registered, mapped to its virtual counterpart, its activity is logged and associated executables are run based on the application. Though there are several similarities, both [16] and [18] focus on a very different usage model than the one proposed in this report, which is driven by explicit user actions.

The proposed approach in this report is closely aligned with (but independently explored from) Elope as described in [12]. Both architectures investigate how RFID-enhanced physical objects can be discovered and their associated services launched through RFID-enhanced mobile devices. One distinct difference is that Elope focuses on a web-based service discovery framework, whereas this report integrates RFID technology with the UPnP framework. Again, the details of the data representation on the tag are not discussed in [12].

In [11] mobile devices act as the mediator between the user and RFID-augmented physical objects. The tags contain data that trigger context events in the system (referred to as CAPNET-based middleware). The primary focus of [11] is on usability tests, the users' perception of visually marked RFID tags, the social acceptance of a touch-based interaction model, and the users' feedback on security concerns and the user interface. This report places a stronger emphasis on the design of the middleware architecture, the technical realization through the UPnP technology and the details of the data representation on the tag for network connectivity and service discovery (an aspect not addressed in [11]).

Passive RFID tags and Bluetooth nodes are used in [17] to augment everyday products and objects. Bluetooth nodes are attached to RFID scanners and they are used as mobile access points allowing data stored on a passive tag (e.g. product codes) to access the background infrastructure in order to be semantically interpreted. The focus of [17] is on logistics-driven use cases such as Smart Product Monitoring, Smart Medicine Cabinet and Remote Interaction with Smart Objects and Locations. In contrast, this work focuses on consumer-driven services such network access, printing, faxing, teleconferencing and so on.

In [13] the authors present a middleware platform, referred to as MSDA, that manages the dynamic composition of networks, integrates existing middleware protocols (e.g. Jini, UPnP), and provides a generic service to clients for performing service discovery. MSDA-aware clients can connect with services in different discovery domains through ad-hoc networks, hotspots or Internet/Cellular networks. The intended user interaction model is different from the proposed method in this paper. In [13] users can discover a new service by reading an RFID tag but additional manual steps are necessary. The user needs to initiate a search for an available service in the network that can interpret and process the description read from the RFID tag. The type of RFID technology used and the tag data representation are not discussed.

An RFID-enhanced framework for intelligent products is presented in [14]. The key technologies utilised are also RFID and UPnP, as in this paper. The focus, though, is on logistics and production processes and the RFID technology used is based on EPC standards. Objects are enhanced with UPnP functionality. Either RFID sensing or direct UPnP message exchange invokes the service discovery process by providing the object ID. The example case study is a warehouse management system.

In [15] RFID tags of very small capacity (a few bytes) are affixed to physical objects containing a simple ID. The tag reader scans the object ID, determines the current application context and provides the appropriated feedback. The use cases focus on how RFID-enhanced objects can be used to present information on a wireless handheld device. However, this approach does not enable services to access data stored in mobile devices.

4. THE TOUCH PARADIGM

Users do not want to employ technology but rather to interact with their environment. Even though mobile phones have become a commodity, a major part of mobile applications and services is hardly used by today's consumers. For example, basic functions such as calling or text messaging are easy to use and have been widely adopted. In contrast, browsing or file sharing require complex configurations and setup procedures and are less popular at present. In this work:

- The touch paradigm creates an intuitive functionality model from the user perspective. It allows for fast, convenient and intuitive user interaction with smart objects, devices, services and other users.
- Service discovery provides a unique opportunity to boost the adoption of mobile services, in particular wireless proximity services. Rather than require new models of behaviour, social interaction in proximity can build on familiar human activities such as giving, sharing, greeting, self-expression, acknowledgement and so on.

4.1 Context Awareness

User activities in proximity can exploit the benefits of context information relevant to service discovery. Context information enables the right services to be delivered to the right user at the right time. Objects pertaining to a certain context can be active or passive at given moments in time depending on the situation of the user. Proximity-based services can be categorised in four context-aware categories:

- a) People in places: social interaction, communication and collaboration in close physical range. Example applications are face-to-face content sharing, ad-hoc collaboration and group formation.
- b) Me and my stuff: creation, management and storage of personal and 3rd party content with particular focus on the home environment. For example, personal mobile devices (e.g. phones, PDAs) can access and control services and content on networked home devices (e.g. laptop, audio system, home appliances).
- c) Smart spaces: accessing local content and services relevant to a particular location. Location-based services can find application both in the workplace and in the consumer market. For the former, an example is personal mobile devices discovering and interacting with other devices in a specific room or building (e.g. printers, projectors). Examples for the latter are service activation at point-of-sale locations and content download (e.g. download a movie preview from a poster).
- d) Safe consumption: research and purchase of goods, content and services with perceived security and trust. All transactions have security and privacy requirements but special attention is needed when purchases and monetary transactions are involved such as ticketing and electronic wallet applications.

Location is an important element of context information that can be exploited. NFC technology can provide a convenient way to access location-aware, mobile services and content through hot spots, e.g. NFC-equipped devices could easily read tags at point-of-sale locations. This can serve to compliment cellular coverage and provide the illusion of full mobility, thus, making it less necessary to assure real-time full mobility for all applications.

Yet, commercial success of this business model is dependent on whether users are willing to wait to connect, pricing and sufficient coverage with clearly marked hot spots.

4.2 User Interaction

Enabled by NFC technology, service discovery has a direct impact on the design of smart and intuitive user interfaces (UIs) for pervasive computing. Currently, similar point-and-click interfaces are not flexible. Most RFID readers, bar code scanners and IR remote controls are single purpose devices. In some cases, IR can be used for multiple purposes, but the interface can hardly be characterized as intuitive, or point-and-click.

Traditional graphical user interface displays receive input through mouse clicks and menu selections. With the NFC-touch paradigm the physical space (e.g. a room) becomes an extension to the GUI of a mobile device, where physical objects (i.e. NFC-enabled objects and devices) can be touched upon in order to activate associated services and applications in the same manner as clicking on an icon on a conventional display. An object may be selected, clicked upon or dropped on different applications, which invokes different actions. Three key modes of interaction are shown in Table 1.

MODE	DESCRIPTION	EXAMPLE
Select	selection is H/W independent via the traditional GUI (e.g. mouse click, menu list) or NFC touch. Selected objects can be interrogated or trigger associated applications	user selects a printer that he/she visually discovers in a room to check its properties (e.g. 'accessed through which network?', 'is it a colour printer?').
Select-and-launch	select an object & launch a related application. Association of metadata to a physical object can be accomplished via various means (e.g. bar codes, tags, service description).	user touches the tag on a movie DVD and the browser is launched with the movie URL as an input.
Select-drag-and-drop	select object A, drag it to resource B, drop it on B & resource B launches associated application. This event pattern associates an object with an application that it is not normally associated with.	use a mobile phone to transfer a file from a laptop to a local printer. The phone-laptop touch gesture represents the file select-and-drag. The phone-printer touch gesture represents the drop.

Table 1: *Key interaction modes.*

5. UPNP FRAMEWORK

5.1 Overview

Universal Plug and Play [1] is a set of protocols including the Simple Service Discovery Protocol (SSDP), the Simple Object Access Protocol (SOAP) and the General Event Notification Architecture (GENA) originally developed by Microsoft Corporation and currently under development by the Universal Plug and Play Forum¹. UPnP standardizes the protocols spoken between clients (called control points) and services. It leverages existing standards such as TCP/IP, HTTP and XML.

Devices, services and control points are the basic abstractions of the UPnP device architecture. The device model is hierarchical. In a compound device, the root device and any embedded devices are discoverable. Clients can address a root or an embedded device independently. Soap servers in the device act as entry points for interacting and controlling it. Each service has a set of methods or actions with a set of optional input/output parameters and return values. The control point is the client and the device is the server. Control points can invoke actions on services. All UPnP devices that conform to UPnP Forum specifications follow the same basic pattern of operation: addressing, description, discovery, control, eventing and presentation. SSDP is used for the service discovery process to (a) announce a device's presence to others and (b) search for devices and services. A device sends a multicast message to either advertise its presence to control points or to search for services in a UPnP network. Devices that hear this message respond with a unicast response message.

UPnP uses XML to describe device features and capabilities. For instance, the aforementioned advertisement message contains a URL that points to an XML file in the network that describes the UPnP device's capability. By retrieving this XML file, other devices can inspect the advertised device's features, learn how to use it, control it and interact with it.

5.2 SSDP Presence Announcement for the Smart Device

An example SSDP presence announcement for a printer is shown in Table 2. A brief explanation of the various fields is given next. For a detailed description the reader is referred to [3], [2]. The value of the NTS field identifies this SSDP message as a presence announcement of a new device or service. The Cache-Control field specifies the duration that the presence announcement is valid. The control point (user) device caches the complete service discovery record for the specified time frame. The USN field provides the device Universally Unique ID (UUID). This is one of the most important fields as it is used to uniquely identify a device. It may contain other information about the device type (e.g. root device,

¹ An industry consortium created by Microsoft.

device type, service type). The Server field provides information on the operating system of the device, the product name and version. The NT field has a potential search target description, i.e. how the control point can search for the discovered device or service. Finally, the Location field contains the URL from which the UPnP device description document can be retrieved. This is another essential field.

This presence announcement is included in the printer tag record is shown in Table 2. The printer record is of Type “SSDP1.0”. Its Content field consists of the six sub-records, namely NTS, Cache-Control, USN, Server, NT and Location URL.

Field	Description	Example Data
NT	Specifies the search target value	"urn:schemas-upnp-org:device:Printer:1"
USN	Concatenation of device ID and NT value	"uuid:0e2fc7b3-4c09-4665-b4ae-f6f90448ba99::urn:schemas-upnp-org:device:Printer:1"
Server	Concatenation of OS name & version, UPnP/1.0, product name & version	Microsoft-Windows-NT/5.1 UPnP/1.0 UPnP-Device-Host/1.0
Location	URL of root device description document	"http://192.168.64.11:53911/upnp/device/Printer.xml"
Cache-Control	Number of seconds the announcement is valid	1800 seconds
NTS	Must be 'ssdp:alive'	ssdp:alive

Table 2: SSDP presence announcement for the printer.

6. TAG DESIGN

The proposed NFC-based tag record design is a flexible and extensible structure that can be used to exchange the IntuiSec commands and data in a variety of use cases (Figure 2). The information exchanged is the Payload. The Payload contains a Header and a record list with one or more records. The Header contains the length of the Payload and it is used to determine how much data must be read from a tag. Each record is a sequence of three elements, a triplet of (Type, Content-Length, Content). The record Type identifies the structure and semantics of the record by providing the Type name. The Content-Length identifies the length of the record Content. The record Content contains the actual data.

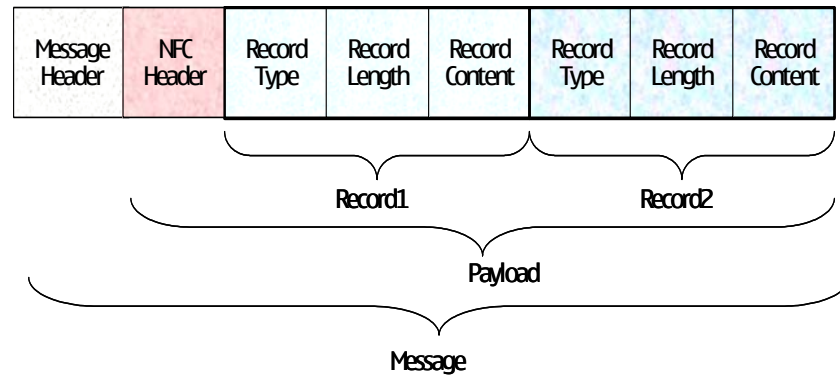


Figure 2: The basic tag structure.

Five record types are defined:

- **KNtipWlanSsid:** which contains the SSID name.
- **KNtipAppData:** which is used in conjunction with other record types and can contain different pieces of information. For instance, when used as part of the SSDP1.0 type, it contains the Device_URL, Device_type, Device_UUID (always in this order).
- **SSDP1.0:** which contains two records, a KNtipWlanSsid and a KNtipAppData. The UPnP Service Discovery Server subscribes to this record.
- **BRWLaunch:** which contains two records, a KNtipWlanSsid and a KNtipAppData. A background client application registers for this type.
- **MYDATA:** the content in this record is a text string for displaying purposes. iTouchConfig registers for this type.

These records can be used alone or in combination depending on the intended task. For example, the tag for Service Discovery uses the composite SSDP1.0 record containing two other records (Figure 3). For connection setup alone, the tag contains only the NAP access record (KNtipWlanSsid). For launching a browser with a given URL through a WLAN, the composite BRWLaunch record is used (Figure 4). Finally, for a tag containing plain text MYDATA record is used (Figure 5).

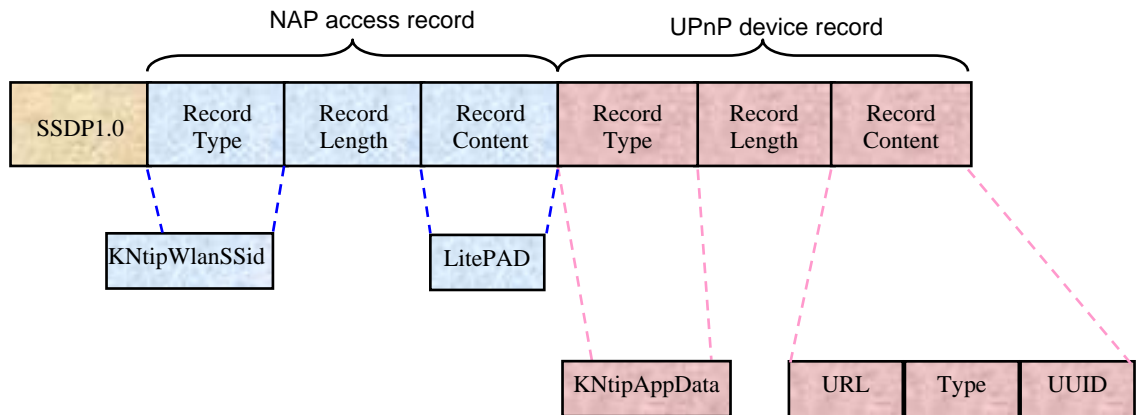


Figure 3: Tag design for SD module.

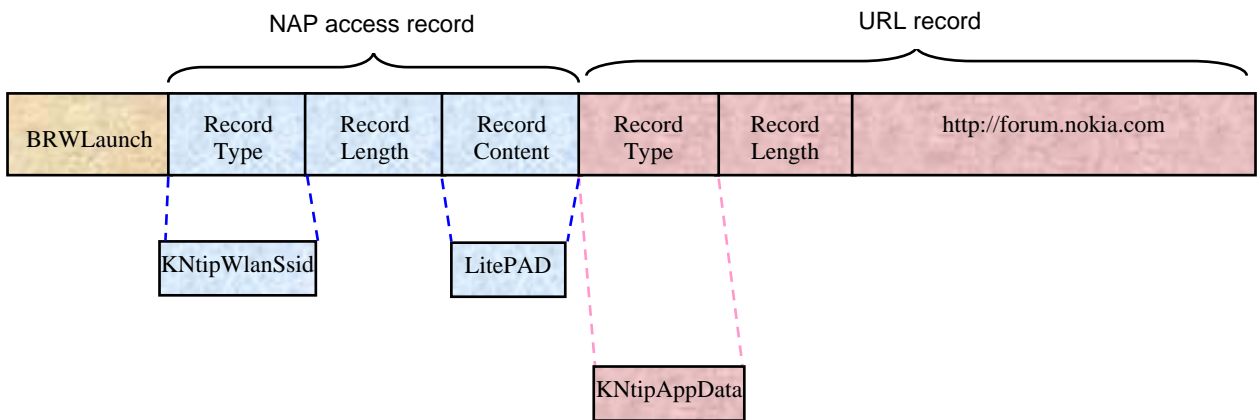


Figure 4: Tag design for Connection and Browser launch.

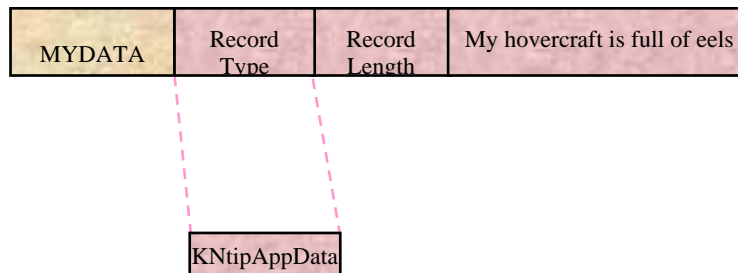


Figure 5: Tag design for iTouch core functionality (read and display text).

7. ARCHITECTURE

7.1 Overview

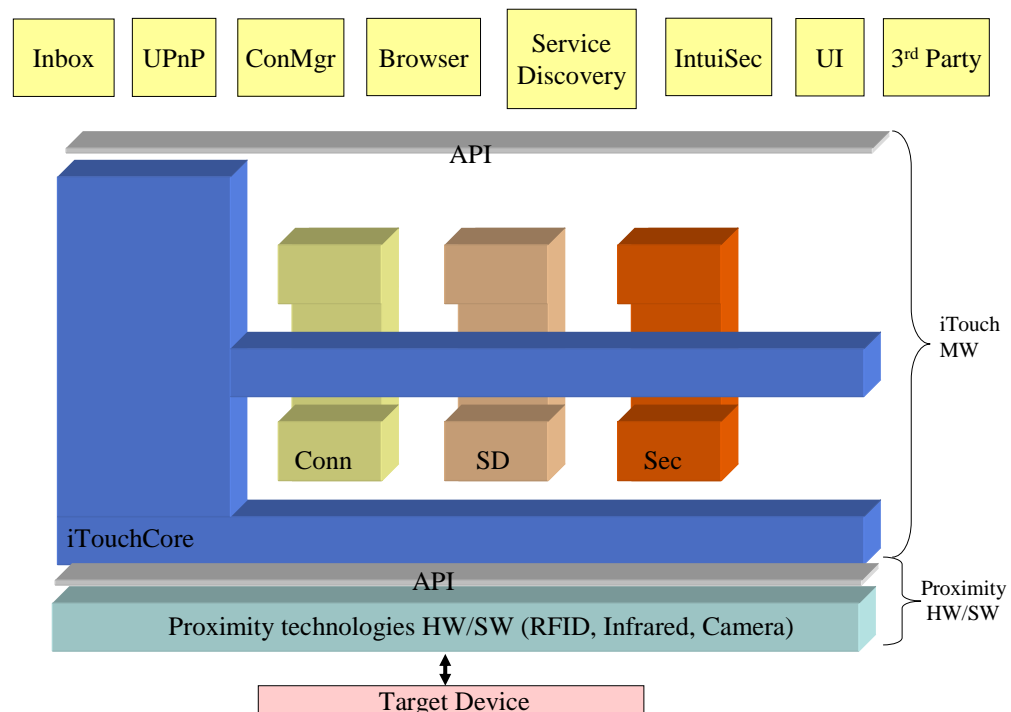


Figure 6: iTouch vision: enable a new set of applications & tools by utilising a variety of proximity technologies.

iTouch (Figure 6) is an NFC-enhanced middleware architecture for mobile devices that enables intuitive user interaction in smart spaces, easy service discovery and face-to-face sharing. It provides a layer of indirection between (a) a variety of proximity connectivity technologies (RFID, Infrared, Camera, Laser etc), and (b) user-domain applications, tools and service discovery engines. These technologies offer a means to exchange information, interact with devices and tagged objects and discover services (represented by Target Device layer). Communication between the different layers is realized through APIs.

iTouch consists of one core module, iTouchCore, and multiple specialized modules. iTouchCore provides the basic read/write, send/receive functionality, as well as, composition and parsing of the NFC records. Client applications register with iTouch middleware through

the iTouch API in order to indicate which data records they are interested in receiving. A client can register to receive specific record types, parse them and process them further. This allows a client to maintain a level of privacy and have complete control of the interpretation of the data. Specialized modules can be added to perform dedicated tasks and add value to the middleware. In the proposed design there are three specialised modules: ‘Connectivity’, ‘Service Discovery’ and ‘Security’.

7.2 Prototype Design

This section presents the proof-of-concept design for iTouch, shown in Figure 7. iTouchCore is, also, referred to as the Link Management layer.

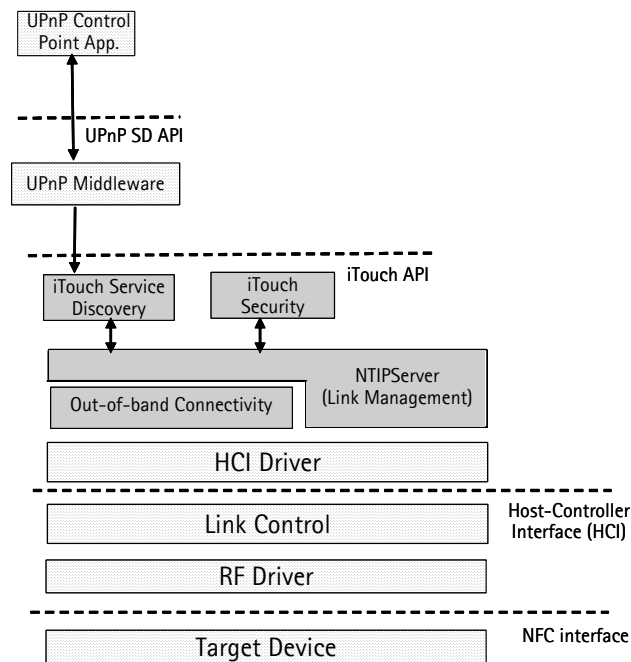


Figure 7 – Proposed middleware architecture.

Starting from the lower layers, a Host Controller Interface (HCI) driver resides on the mobile device and communicates with an NFC transceiver unit over a USB interface. The Link Control layer on the transceiver performs the RF communication. The transceiver is controlled by the host via commands sent through the HCI driver. The design allows for the transceiver to be “armed” at the press of a button on the host, activating the RF layer. At all other times, the RF unit is in idle mode, thus conserving power. Data is exchanged in the form of NFC packets

which are communicated over the air interface to target devices (send/receive) or tags (read/write). Target devices are enhanced with the same set of middleware tools.

The HCI driver interacts with the Link Management layer. In this layer, a variable length packet format is defined for the frames communicated over RF. This format is based on NDEF guidelines (NFC Data Exchange Format, formerly called NTIP, NFC Transfer Interchange Packet) as defined by the NFC Forum². The heart of the Link Management layer is the NtipServer. The NtipServer can manage multiple clients on the host device. It operates in the asynchronous mode, collecting requests from all clients. It uses a unique type string that binds each request to the client. Both applications and specialized middleware modules can act as clients, by registering with the NtipServer to receive specific record types. This allows clients to maintain a level of privacy and have complete control of the interpretation of the data. Record types are mapped to clients in an internal registration database.

Incoming NFC packets are received by the NtipServer. These packets can contain one or more records. The NtipServer parses them and extracts the NFC record(s). Next, it determines the destination of these records based on the registration database. In the reverse direction, client applications can use public functions exposed through the iTouch API to write to tags or send messages through the NFC interface. In this case, the NtipServer receives a set of data parameters through the API and composes an NFC record in the appropriate format before sending it over the air interface.

The specialised modules are described next:

- **Out-of-band Connectivity module:** this module is designed to setup out-of-band network connections, such as WLAN or Bluetooth. It can declare its own record type(s). When such a record type is extracted from an NFC packet in the Link Management layer, it is passed to this module where it is parsed, processed and the necessary steps are taken to establish the connection. This module is responsible for processing the KNtipWlanSsid records as described in section 6. In the current implementation, only the SSID of the WLAN NAP is used, although it is possible to add WEP keys. Upon extracting the SSID, this module updates the CommsDB tables and creates a new IAP (Internet Access Point) ID reference. The CommsDB database is a set of tables containing the connection setup parameters in the Symbian-based N9500 software architecture. Once created, the IAP's ID is passed to the client application that has subscribed for the KNtipWlanSsid record as a pointer to a WLAN connection that is set up and ready to be established. In the case of the SSDP1.0 or BRWLaunch records other data (KNtipAppData) is, also, passed to the

² The names NDEF and NTIP are used interchangeably. NFC standardization is ongoing, hence naming conventions and record formats are still evolving.

Connectivity module. This data is merely relayed to the client application along with the IAP ID.

- Service Discovery module: this module is designed to perform service discovery. Similar to the Out-of-band Connectivity module, it can, declare its own record type(s) that contain service discovery descriptions. For the case of UPnP service discovery, this module processes the SSDP1.0 records (section 6). First, the Link Management layer passes the SSDP1.0 record to the Out-of-band Connectivity module, which processes the KntipWlanSsid record and creates the IAP ID. Next, the IAP ID and KntipAppData record (containing the UPnP service discovery parameters) are passed to the Service Discovery module. The UPnP parameters (Device_URL, Device_type and Device_UUID) are extracted and passed to the UPnP Service Discovery Server with the IAP ID.
- Security module: In the current design, this module provides an integration point with the IntuiSec security framework [6] that utilizes proximity wireless technologies such as NFC to perform secure network setup and device discovery. This module is described in more detail in [7].

7.3 Service Profiles

iTouch middleware defines the following profiles of operation as shown below.

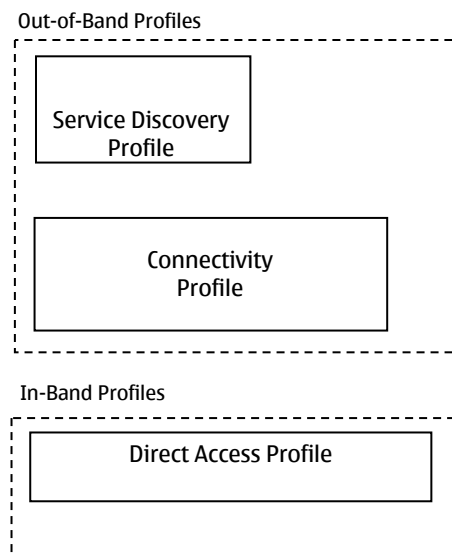


Figure 8: In-band and Out-of-Band Connectivity Profiles.

The Direct Access profile defines the basic mode of operation. This is left exposed to allow the design to be extended by defining new services as profiles in the architecture. Under the current implementation, the Direct Access profile allows the user to create a record of type MYDATA, write to a tag, read it and display its contents on the screen.

The Connectivity and Service Discovery modes constitute the Out-of-Band profile, where the clients can expect some out-of-band processing from the MW besides reading through the RFID medium. The Connectivity profile is used by clients that require a connection to be setup upon reading the tag. The SD Profile is used to discover and interact with UPnP devices such as printers, media servers and renderers and so on.

7.4 iTouch API

7.4.1 Constructor

```
CNtipClient* NewL(TInt aMaxOutstandingReadReq =
NtipServer::KNtipMaxClientRequests);
```

Description	Creates a new session with the Ntip server.	
Parameters	TInt aMaxOutstandingReadReq	Max no. of (async) read requests that can remain outstanding for this client.
Return	Initialized CNtipClient object.	

7.4.2 Registration

7.4.2.1 Basic Read (Direct Access Profile)

```
TInt SetNtipObserverL(TDesC8& aType, MNtipObserver* aObserver);
```

Description	Registers interest for a specified record type with the Ntip server by setting the observer. In the direct access mode, the client should only expect to read a record from RFID, without any out of band processing by the Ntip server. The Ntip server will discard all RFID reads of the given type if there is no outstanding read request from this client. A client can be bound to more
-------------	--

	than one type by calling this method more than once. The Ntip server will allow only one client to register for a specific record type (irrespective of direct or connectivity mode), on a FCFS basis.	
Parameters	TDesC8& aType	Type for which registering (e.g. "MYDATA")
	MNtipObserver* aObserver	Callback to invoke upon reading a record of specified type
Return	KErrNone on success. KErrAlreadyExists if another client has already claimed this record type. Else one of the other system wide error codes.	

7.4.2.2 Connect-and-Read (Connectivity Profile)

```
TInt SetNtipConnObserverL(TDesC8& aType, MNtipConnObserver* aObserver);
```

Description	Registers interest for a specified record type with the Ntip server by setting the observer. In the connectivity mode, the client should expect the Ntip server to setup an IAP (over Bluetooth or 802.11) and pass the corresponding IAP id, along with application data to the client. The Ntip server will discard all RFID reads of the given type if there is no outstanding read request from this client. A client can be bound to more than one type by calling this method more than once. Ntip server will allow only one client to register for a specific record type (irrespective of direct or connectivity mode), on a FCFS basis.	
Parameters	TDesC8& aType	Type for which registering (e.g. "SSDP1.0").
	MNtipConnObserver* aObserver	Callback to invoke upon reading a record of specified type.
Return	KErrNone on success. KAlreadyExists if another client has already claimed this record type. Else one of the other system wide error codes.	

7.4.3 Reading from RFID tag

```
TInt ReadNtip();
```

Description	Requests the Ntip server for a read of the type specified in SetNtipObserverL. This is an asynchronous call.	
Parameters	None	
Return	KErrNone if Ntip server acknowledged the read request, else a system wide error code.	

Notes: Calling this method does not start the RFID read loop (that is done by the user pressing the “ARM” button on the device). This method simply informs the Ntip server that there is an outstanding request from this client.

7.4.4 Receiving data upon RFID read

7.4.4.1 Basic Read (Direct Access Profile)

```
class MNtipObserver
{
    virtual void HandleNtipRecordL(TDesC8& aType, TDesC8& aData) = 0;
};
```

Description	This class contains the callback to be invoked after the Ntip server has read a record (in the direct access mode) of the specified type over the RFID medium.	
Parameters	TDesC8& aType	The type for which the application had set this observer.

	TDesC8& aData	The buffer containing the application data that was read.
Return	None	

7.4.4.2 Connect-and-Read (Connectivity Profile)

```
class MntipConnObserver
{
    virtual void HandleNtipConnRecordL(TDesC8& aType, TInt aIAP, TDesC8& aData) = 0;
    virtual void HandleConnSetupProgressIndication(TInt aPercentageCompleted) { /*
default implementation - do nothing */}
};
```

Description	This class contains the callback to be invoked after the Ntip server has read a record (in the connectivity mode) of the specified type over RFID medium.	
Parameters	TDesC8& aType	The type for which the application had set this observer.
	TInt aIAP	The IAP corresponding to the link that was setup. The client should make no assumptions about the underlying medium except that it runs IP.
	TDesC8& aData	The buffer containing the application data that was read.
Return	None	

This observer also provides a `HandleConnSetupProgressIndication()` callback method which is invoked while the MW is preparing the connection. This may be used by UI based applications to display suitable feedback to the user.

7.4.5 Canceling a Read request

```
TInt CancelRead();
```

Description	Cancels the last read request made by this client.	
Parameters	None	
Return	KErrNone if a request was cancelled. KErrNotFound if there was no outstanding request.	

7.4.6 Arming the RFID hardware

```
void ArmTrigger();
```

Description	Starts the read loop on the RFID transceiver, and moves the NTIP server into the LISTEN state. This method is meant for internal use only, to emulate the “hot button” feature on the phone. It should not be used by clients.	
Parameters	None	
Return	None	

7.4.7 NTIP Server Shutdown

```
void CloseTrigger();
```

Description	Stops the NTIP server. Meant for internal use only, should not be used by clients.	
Parameters	None	
Return	None	

8. USER INTERACTION TOOLS

This section presents the user interaction tools and use cases that were implemented to demonstrate the features of iTouch. Part of the demonstration tools is an application, called *iTouchConfig*, which provides basic functions such as starting and stopping the NtipServer, reading from and writing to an RFID tag, sending to and receiving from another NFC-enabled device. In addition, an RFID hot-button has been implemented to arm the RFID reader before each reading action. There are three use cases showcasing WLAN connection setup, reading URLs and launching the browser and UPnP service discovery.

8.1 Starting iTouch

Launch the iTouchConfig application and choose the Start iTouch option from the menu. This initiates automatically a connection to the RFID hardware (Figure 9). Once iTouch server has been started, you can exit iTouchConfig. This will not kill the server.



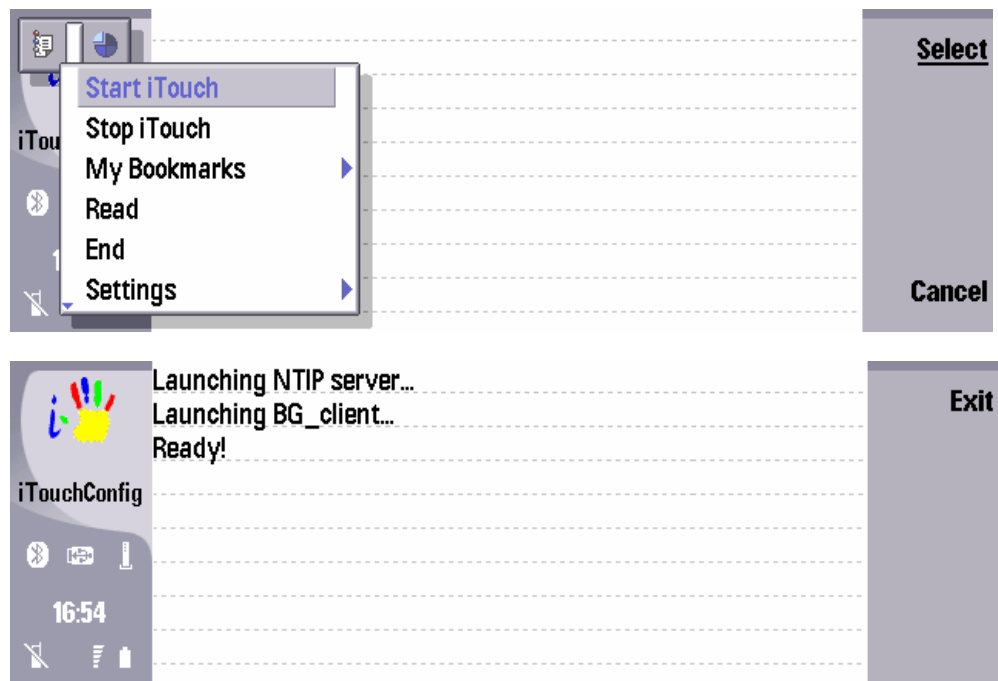


Figure 9: Starting the iTouch Server.

8.2 iTouch hot button

“My Own” button has been implemented as an iTouch hot button. Pressing the iTouch button (after the middleware server has been started) starts the `ReadNtip()` loop in the HW. It remains active till a read operation has been completed, or till a timeout period is elapsed (~ 10sec). Each read operation requires pressing the button. This functionality is implemented by using the `ArmTrigger()` method. Upon pressing the iTouch hot button, the application prompts the user to touch the RFID tag and exits thereafter (Figure 10). The motivation for the hot button is based on the following:

- Having the HW in a reading loop for a prolonged period of time could drain the battery.
- An RFID hot button is one way of designing a very simple “touch-and-click” usage model. It eliminates the need for several manual steps through menus.



Figure 10: Hot-button to initiate reading through the RFID interface.

8.3 Writing to a tag

iTouchConfig offers four bookmarked options for writing as shown in Figure 11. The first three contain default but editable data for Connection Setup (Touch-to-Connect), Connection Setup and Browser Launch (Touch-to-Launch), Connection Setup and Service Discovery (Touch-to-Discover). The fourth bookmark option demonstrates the basic write functionality of simply adding text.

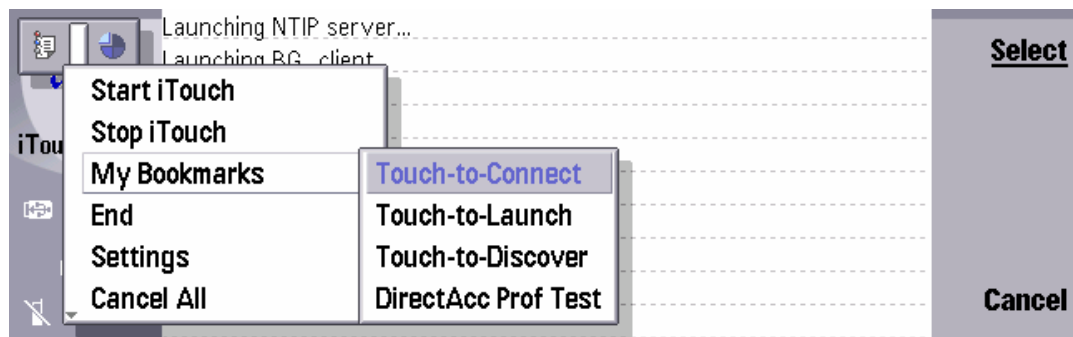


Figure 11: Bookmarked options for writing to an RFID tag.

For the Touch-to-Connect case, the user needs to enter the WLAN SSID. A default entry exists (Figure 12). Next, press OK and touch the tag to complete the task. In the future more fields such as connection mode, WEP keys, channel and so on can be added.

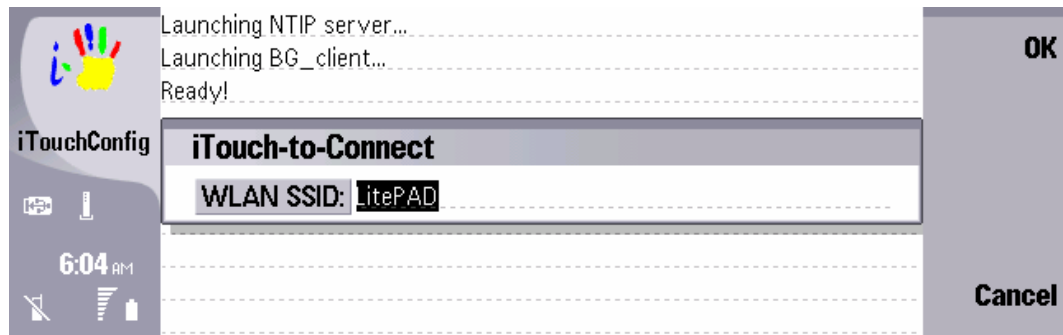


Figure 12: Writing a WLAN NAP record to a tag.

For the Touch-to-Launch case, the user needs to enter the WLAN SSID and the URL. Default entries exist (Figure 13). Next, press OK and touch the tag to complete the task.

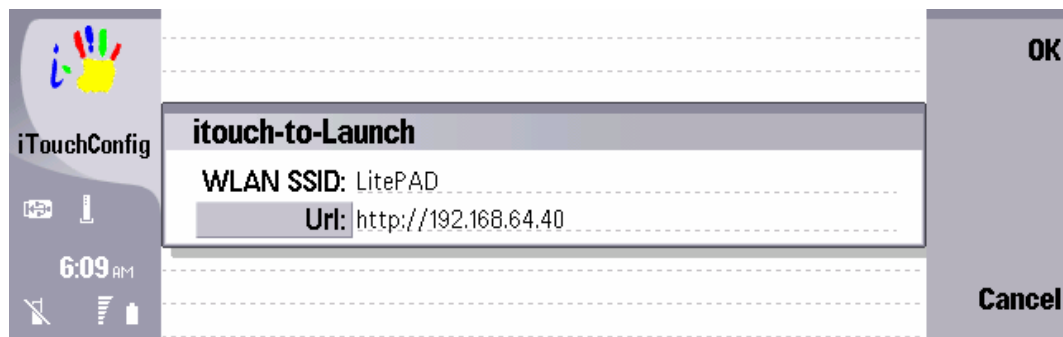


Figure 13: Writing a tag for launching a URL through a local WLAN.

For the Touch-to-Discover case, the user needs to enter the WLAN SSID, the device location URL, the device type and UUID. Default entries exist (Figure 14). Next, press OK and touch the tag to complete the task.

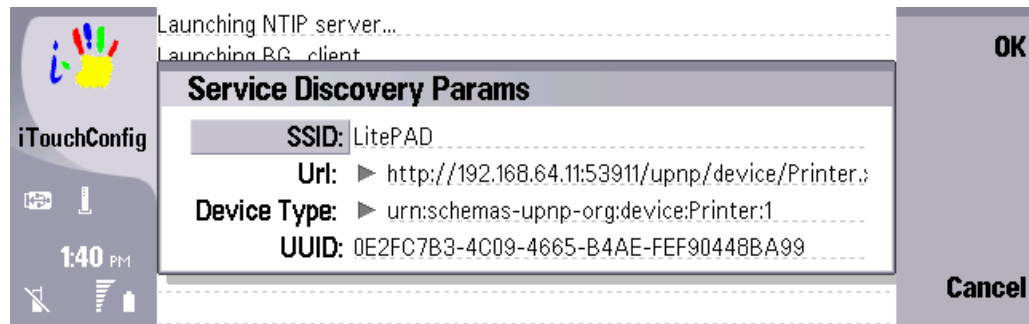


Figure 14: Write a tag for UPnP Service Discovery.

8.4 Client applications

The Touch-to-Connect use case is implemented within the iTouchConfig application. The use case tests the setting up of a WLAN connection, and shows a dialog to the user notifying that the NAP with a given SSID has been discovered.

The Touch-to-Launch use case is written to showcase the middleware's ability to launch applications (browser, email) that require a local connection. A separate background client application has been written to run as a server. This application registers for the BRWLaunch record. When such a record is read, it is processed by iTouch middleware and passed to the client application as it is described in section 7. The record triggers the launch of the browser and loads the URL. This allow iTouch functionality to be demonstrated even when the phone is in the idle screen.


For the Touch-to-Discover use case, the UPnP Service Discovery Server (shown as UPnP middleware in Figure 7) runs as a client to the NtipServer. It registers to receive the content of the SSDP1.0 records.

8.5 Use case I: Touch to connect to WLAN NAP

Description: iTouch Server has been started. The WLAN NAP tag has been written. User touches WLAN tag with mobile device to discover WLAN NAP. The reading action needs to be invoked from within the iTouchConfig application.

Stimulus/response sequence:

- Start iTouch Server (Figure 9)
- Create the tag if necessary (see section 8.3)

- User presses the iTouch Hot Button (Figure 10)
- User device touches the tag to read it 
- A progress bar indicator shows the connection setup progress
- A dialogue box prompts the user that a NAP has been discovered (Figure 15)
- The user can then choose to manually launch an application that will use the newly discovered WLAN NAP (i.e. Internet Explorer).

Resulting state: The discovered WLAN NAP is added to the list of available access points in Control Panel ⇒ Connections ⇒ Internet Setup menu (Figure 16). The “W” icon appears in the left-hand side of the screen.

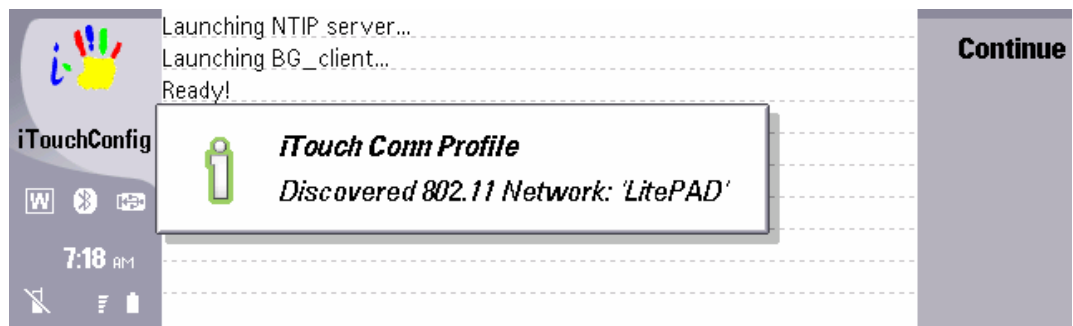


Figure 15: WLAN NAP has been discovered.

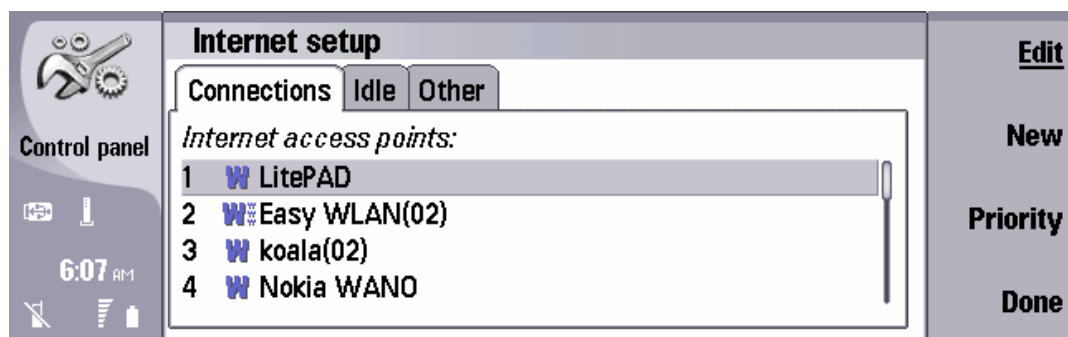



Figure 16: The discovered WLAN is added to the list of available NAPs.

8.6 Use case II: Touch to connect to WLAN hotspot & launch URL

Description: iTouch Server has been started. The tag contains the BRWLaunch record (see section 8.3). User touches tag with mobile device to launch a URL through a local WLAN NAP. A background client application is launched automatically upon starting the iTouch server, which registers for this record type. As a result, the reading action can be invoked either from within or outside of iTouchConfig. The background client receives and processes the BRWLaunch record. This demonstrates (a) the ability of iTouch to allow multiple clients to register for different record types and (b) the ability of the background client to launch an application (i.e. browser) based on the received record type.

Stimulus/response sequence:

- Start iTouch Server (Figure 9)
- Create the tag if necessary (see section 8.3)
- User presses the iTouch Hot Button (Figure 10)
- User device touches the tag to read it 
- Browser is launched pointing to URL included in the tag (Figure 17)
- Browser will prompt the user which access point to select, even though a NAP has been indicated in the tag. This dialogue box is controlled by the browser application and can not be disabled. The access point discovered through the tag is on the list of available access points presented to the user. Once the NAP selection is made, (a) an active WLAN connection is indicated by the appearance of the “W” icon on the left-hand side of the screen and (b) the browser loads the URL.

Resulting state: The discovered WLAN NAP is added to the list of available access points. A connection is established. The “W” icon appears on the screen. The browser loads the URL indicated on the tag.

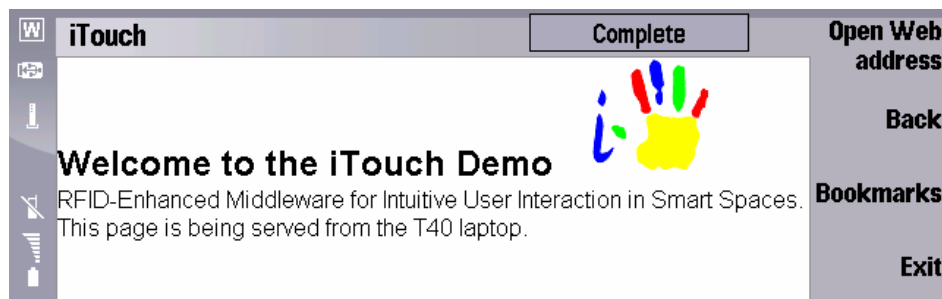



Figure 17: The browser is launched and the web page is loaded.

8.7 Use case III: Touch to discover UPnP devices - UPnP printing

Description: iTouch Server has been started. The tag contains the BRWLaunch record (see section 8.3). User touches tag with the mobile device to discover and connect to a local UPnP printer through a local WLAN NAP. The tag contains an SSDP1.0 record. The following needs to take place prior to reading this record: first, the Sample Browser application needs to be launched, secondly, the RFID option must be enabled and thirdly, the SSDP1.0 record needs to be written to the tag through iTouchConfig as indicated in section 8.3. This action, registers the UPnP Service Discovery Server to receive the SSDP1.0 record when it is read.

Stimulus/response sequence:

- Start iTouch Server (Figure 9)
- Launch the Sample Browser application & enable RFID Discovery (Figure 18)
- Create the tag; this is necessary in order to register the UPnP Service Discovery server for the SSDP1.0 record (see section 8.3)
- User presses the iTouch Hot Button (Figure 10)
- User device touches the tag to read it 
- The UPnP printer control point is launched showing the status of the device (Figure 19, Figure 20).
- Select the Print option from the menu to print a demo page to the newly discovered printer (Figure 21).

Resulting state: The WLAN NAP is added to the list of available access points. A connection is established. The “W” icon appears on the screen. The UPnP Control Point application is launched automatically and the user can interact with the printer.

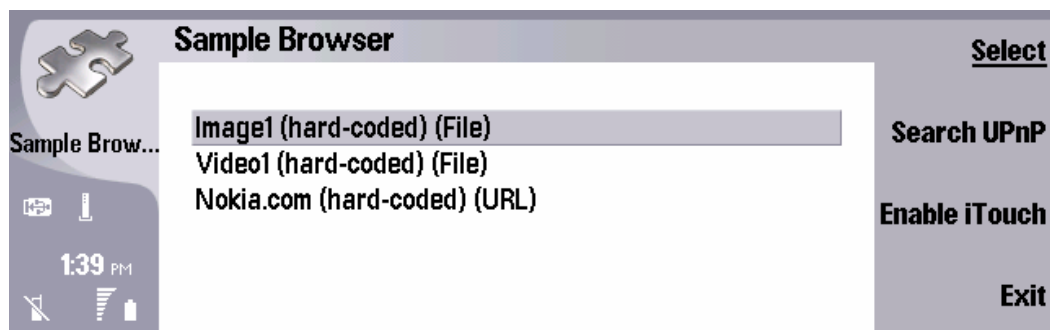


Figure 18: Sample Browser application.

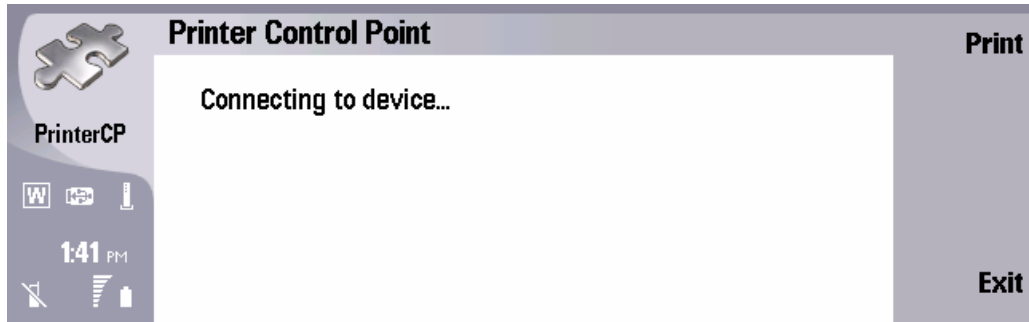


Figure 19: UPnP Printer Control Point is launched.

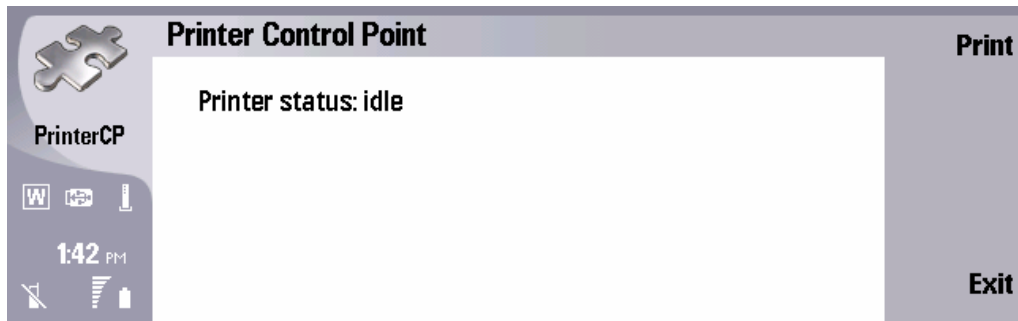


Figure 20: Control Point application shows device status.

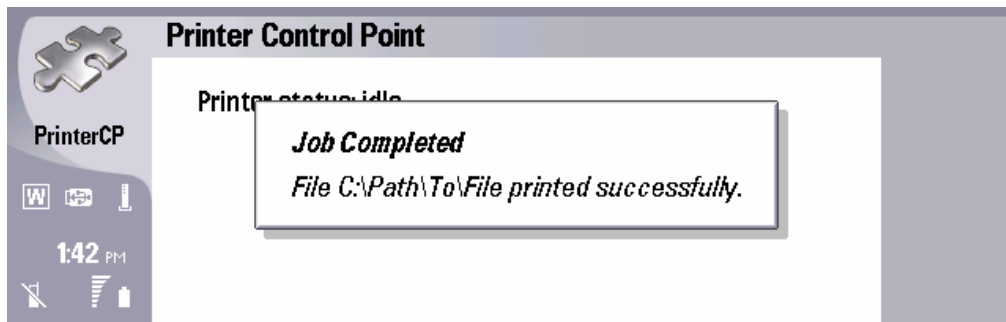


Figure 21: Print a demo page.

8.8 Ending the demo

To end the demo, stop the iTouch Server (Figure 22).

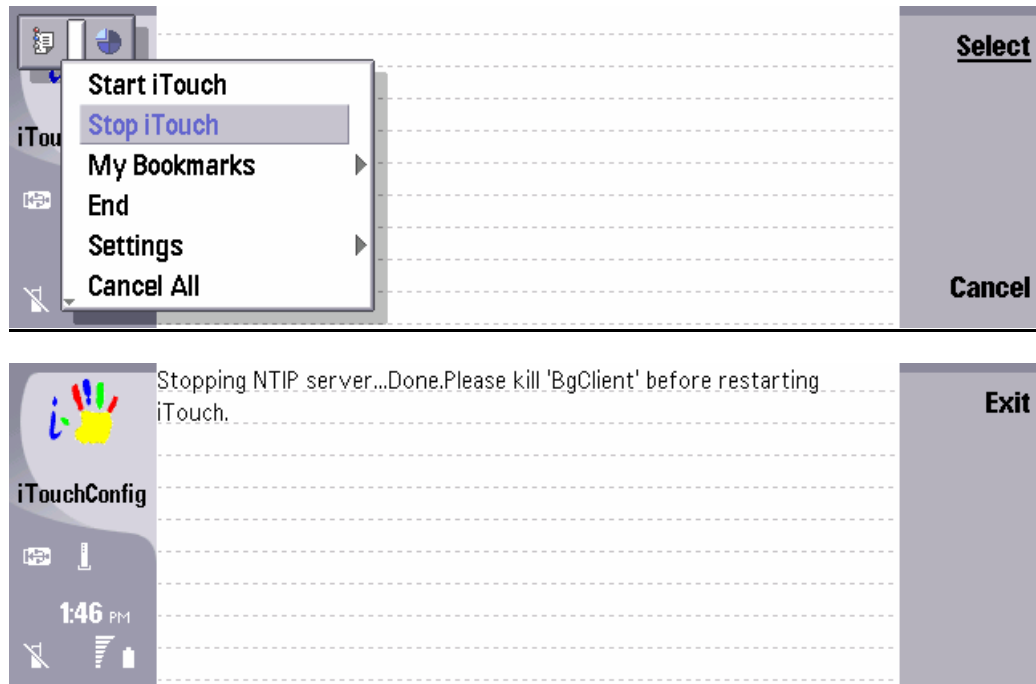


Figure 22: Stopping the iTouch Server.

REFERENCES

- [1] UPnP Forum, "UPnP Device Architecture 1.0.1", December 2003. <http://www.upnp.org>
- [2] NFC Forum, www.nfc-forum.org.
- [3] "Printer:1" Device Template Version 1.0, Approved Standard, UPnP Version 1.0, August 2002.
- [4] "WLANAccessPointDevice:1", Device Template Version 1.01, Standardised DCP, UPnP Version 1.0, October 2003.
- [5] "WLANConfiguration:1", Service Template Version 1.01, Standardised DCP, UPnP Version 1.0, October 2003
- [6] Kalofonos D. and Shakhshir S., "IntuiSec: a Framework for Intuitive User Interaction with Security in Smart Spaces". Nokia Technical Report, NRCC-TR-2006-003.pdf.
- [7] Z. Antoniou and D.N. Kalofonos, "NFC-Based Mobile Middleware for Intuitive User Interaction with Security in Smart Homes", submitted for review.
- [8] J. Heidermann, R. Govindan and D. Estrin, "Configuration Challenges for Smart Spaces", University of Southern California, USC Technical Report, July 1998.
- [9] A. Misra, S. Das and A. MaAuley, "Autoconfiguration, Registration and Mobility Management for Pervasive Computing", IEEE Personal Communications, August 2001.
- [10] Z. Antoniou, G. Krishnamurthi and F. Reynolds, Intuitive service discovery in RFID-enhanced networks, *Proc. of IEEE COMSWARE Conference*, India, 2006.
- [11] C. Bettstetter and C. Renner, "A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol", Proceedings Sixth EUNICE Open European Summer School, 2000.
- [12] J. Riekkki, T. Salminen and I. Alakarppa, "Requesting Pervasive Services by Touching RFID Tags", *Pervasive Computing Magazine*, January-March 2006.
- [13] T. Pering, R. Ballagas and R. Want, "Spontaneous Marriages of Mobile Devices and Interactive Spaces", *Communications of the ACM*, Vol. 48, No. 9, September 2005.
- [14] P-G Raverdy et. al, "The MSDA Multi-Protocol Approach to Service Discovery and Access in Pervasive Environments", *WSProceedings*, http://middleware05.objectweb.org/WSProceedings/demos/d10_Raverdy.pdf
- [15] E. Bajic, "Ambient Services Modeling Framework for Intelligent Products", Research Centre for Automatic Control, University Henri Poincare, Nancy, Technical Report, CRAN – CNRS UMR 7039
- [16] R. Want et. al., "Bridging Physical and Virtual Worlds with Electronic Tags", Proceedings of CHI'99, ACM Press, April, 1999.

- [17] C. Kerer et al, "Presence-Aware Infrastructure using Web Service and RFID technologies", Technical Paper, Vienna University of Technology, <http://www.infosys.tuwien.ac.at/staff/sd/papers/PresenceAwareInfrastructure.pdf>
- [18] F. Siegemund and C. Florkemeier, "Interaction in Pervasive Computing Settings using Bluetooth-enabled Active Tags and Passive RFID Technology together with Mobile Phones", Proc. of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), 2003.
- [19] K. Romer et al, "Smart Identification Frameworks for Ubiquitous Computing Applications", Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), 2003.