Using Visual Tags to Bypass Bluetooth Device Discovery

David Scott[a] **Richard Sharp**[b] **Anil Madhavapeddy**[a] **Eben Upton**[b]
djs55@cl.cam.ac.uk *richard.sharp@intel.com* *avsm2@cl.cam.ac.uk* *eben.upton@intel.com*
[a]Computer Laboratory, University of Cambridge. 15, JJ Thomson Avenue, Cambridge, CB3 0FD, UK
[b]Intel Research Cambridge. 15, JJ Thomson Avenue, Cambridge, CB3 0FD, UK

*One factor that has limited the use of Bluetooth as a networking technology for publicly accessible mobile services is the way in which it handles Device Discovery. Establishing a Bluetooth connection between two devices that have not seen each other before is slow and, from a usability perspective, often awkward. In this paper we present the implementation of an end-to-end Bluetooth-based mobile service framework designed specifically to address this issue. Rather than using the standard Bluetooth Device Discovery model to detect nearby mobile services, our system relies on machine-readable visual tags for out-of-band device and service selection. Our work is motivated by the recent proliferation of cameraphones and PDAs with built-in cameras. We have implemented the described framework completely for Nokia Series 60 cameraphones and demonstrated that our tag-based connection-establishment technique (*i*) offers* order of magnitude *time improvements over the standard Bluetooth Device Discovery model; and (*ii*) is significantly easier to use in a variety of realistic scenarios. Our implementation is available for free download.*

I. Introduction

Bluetooth is increasingly becoming integrated into a variety of mobile devices, making its way into cell phones, laptop computers and PDAs. The rapid adoption of Bluetooth has led many to suggest that it will act as a key enabling technology, driving the deployment of mobile services. Although today Bluetooth is typically used in a limited fashion (e.g. to transfer files between a user's laptop and their PDA, or to connect their wireless headset to their mobile phone), it has the potential to do much more. For example, publicly accessible Bluetooth-enabled devices in shops could enable consumers to query whether a product is currently in stock using their mobile phone and Bluetooth-based active posters could distribute mobile content (e.g. 1-minute music samples of a band's latest album) to users' PDAs.

One factor that has limited the use of Bluetooth as an underlying networking technology for publicly accessible mobile services is that its *device discovery* model is inappropriate for this kind of interaction (see Section III). In this paper we describe the implementation of an end-to-end Bluetooth mobile service framework designed specifically to address this issue. Our implementation has a number of novel features:

1. Visual tags are used for out-of-band device selection, bypassing the standard Bluetooth device discovery model.
2. The same visual tags are also used for service *selection*, enabling users to choose between the multiple services or content downloads offered by a single Bluetooth device.
3. Unlike many research prototypes our system is built entirely on commodity hardware, demonstrating that our technique is immediately applicable to a variety of existing devices.

We demonstrate that our system for out-of-band Bluetooth connection establishment is (*i*) an *order of magnitude* faster than the conventional Bluetooth Device Discovery model; and (*ii*) that, in a variety of realistic usage scenarios, it is significantly easier to use.

Our work is motivated primarily by the recent proliferation of cameraphones. The Nokia 3660 cameraphone, for example, is a fully programmable device (running a general purpose OS) that incorporates a camera and has Bluetooth capability. Tens of millions of Bluetooth-enabled cameraphones have already been shipped globally, with market trends showing that the rapid adoption of such devices is set to continue for the foreseeable future [8]. By running our software on their cameraphone, a user can simply "point and click" at a tag representing a Bluetooth service. The connection is set up immediately, eliminating the need for device discovery entirely. Our implementation is freely downloadable over the web [14] allowing users to try it for themselves on their existing cameraphones. Note that, although this paper

specifically describes our cameraphone-based implementation, the work is also applicable to the increasing number of PDAs with Bluetooth capability and *built-in* cameras (e.g. Sony Clie, Palm Zire).

The remainder of this paper is structured as follows. Section II gives a brief outline of the parts of the Bluetooth specification that are relevant to this paper: *device discovery* is considered in Section II.A; *service discovery* is described in Section II.B. Section III considers using Bluetooth to connect two devices that have not seen each other before and argues that, in scenarios such as this, the current-generation Bluetooth device discovery model suffers from a number of usability problems. We describe the implementation of our visual-tag based Bluetooth mobile service framework in Section IV. Realistic usage scenarios, in which our device discovery technique offers significant advantages over the conventional Bluetooth Device Discovery model, are presented in Section V. A performance evaluation of our system running on Nokia 36x0 cameraphones is presented in Section VI. Security issues surrounding the use of visual tags to encode device names are considered in Section VII. Related work is described in Section VIII. We conclude and give directions for future work in Section IX.

II. A Bluetooth Primer

Bluetooth is an open standard for low-power, short-range networking that is based on a frequency-hopping spread-spectrum scheme in the 2.4 GHz band [1]. This section serves to briefly outline the parts of the Bluetooth specification that are relevant to the rest of this paper. Readers already familiar with the technical details of Bluetooth may wish to skip straight to Section III.

II.A. Inquiry and Device Discovery

The Bluetooth Baseband Specification [1] describes point-to-point connection establishment as a two step procedure. Firstly, *Inquiry* is performed to discover other Bluetooth devices in the neighborhood. During this phase, the inquiring device learns its neighbors' *Bluetooth Device Addresses* (BD_ADDRs) and frequency synchronization information. Secondly, *Paging* is performed to establish an actual connection with (some of) the discovered device(s).

Inquiry is an asymmetric process: a device wishing to perform inquiry enters the *Inquiry* state; devices waiting to be discovered must be in the *Inquiry Scan* state. The inquiring device repeatedly sends packets over a pre-defined *Inquiry [frequency] Hopping Sequence* in an attempt to locate other Bluetooth devices nearby. Devices in the Inquiry Scan state that hear these inquiry packets respond with a packet containing, amongst other information, their BD_ADDR. Whilst the precise technical details of Inquiry are beyond the scope of this paper, the Bluetooth specification requires the Inquiry Hopping Sequence to be traversed multiple times to ensure all inquiry responses have been collated. In an error-free environment, the inquiry phase must last for 10.24s if it is to discover all devices [1, 19]. In a noisy or error-prone environment, inquiry can last much longer than this (see Section VI).

Bluetooth *Device Discovery* starts by performing Inquiry and then contacts discovered devices to retrieve their *Device Names*. After an inquiry phase, paging is performed to establish a connection with each of the discovered devices; *Name Discovery* itself then consists of a simple request/response protocol. All this adds to the latency a user experiences when they select "discover Bluetooth devices" on their PDA or phone.

II.B. Service Discovery

The Service Discovery Protocol (SDP) provides a mechanism for applications to discover which Bluetooth services are provided by a remote device and to determine the attributes of those services. The attributes of a service include the type or class of service offered and the mechanism or protocol information needed to access the service. To perform service discovery, a device establishes an L2CAP connection[1] and uses this to talk to a remote device's SDP server.

A client will typically perform a two-stage process to obtain service information. Firstly, a *Service Search Request* is issued by the client, specifying which classes of service it is interested in (e.g. generic printing services). The client sends the Service Search Request to the SDP server, which responds by returning a list of *service handles* that identify the services available on the remote device that match the client's service search request. Secondly, the client may perform a *Service Attribute Transaction* to find out more about one of the available services. During this process, the client sends a *Service Attribute Request* containing a service handle and a list of the *service attributes* requested. The SDP server responds with the values of those attributes.

[1] L2CAP is Bluetooth's datagram-based Logical Link Control and Adaption Protocol.

An example of a service attribute is the *Protocol Descriptor List* which identifies the communications protocols over which the service is running and provides protocol-specific parameters. A client uses this information to connect to a service.

III. Usability Issues Regarding Device/Service Discovery

Having briefly introduced Bluetooth, we now continue by considering usability issues arising from its device discovery model. Consider the following common scenario: Alice wants to share a picture by transferring a JPEG file from her cameraphone to Bob's cameraphone; Alice and Bob's cameraphones have not seen each other before. In order to affect this transfer Alice must perform the following actions:

Select photo: Use the menu system on her phone to select the picture to transfer via Bluetooth.

Device Discovery: Wait whilst a list of Bluetooth device names appears on the phone's display.

Device Selection: Select the device name corresponding to Bob's cameraphone.

From a usability perspective, this scenario highlights three significant issues. Firstly, the device discovery phase takes a significant amount of time. In a realistic environment it is not uncommon for device inquiry and name discovery to take in excess of 30 seconds (see Section VI). This delay is by far the dominant time-cost for the whole transaction, dwarfing the time taking to transfer the small JPEG file.

Secondly, device selection requires Alice to know the device name of Bob's phone. If Bob has not thought carefully about giving his device a distinguishable name (or, more likely, has left the manufacturer's default device name unchanged), Alice is likely to see strings such as "My Phone" or "Nokia 3660". In an environment with many Bluetooth-enabled devices it may be unclear which name corresponds to Bob's phone.

Thirdly, the list of device names Alice has to choose from may be long. Even if Alice knows the name of Bob's phone it may take considerable time to scroll through their device name list and select it. Even today, in crowded places such as train stations, it is common for device discovery to find as many as 10-15 neighboring Bluetooth devices. As Bluetooth becomes more widely deployed, this problem will be exacerbated.

III.A. Context Awareness

The usability issues described above all indicate a lack of context-awareness in the standard Bluetooth device discovery model. There are a number of common scenarios where *a user* knows exactly which physical Bluetooth device they want to connect to. However, they have no way to communicate this contextual information to the devices themselves.

In the scenario above, for example, Alice is able to point to Bob's cameraphone but has no way of telling their own phone that "this is the physical device I would like to connect to". Instead, the connection has to be established by means of the Bluetooth Device Name associated with Bob's phone: a somewhat arbitrary string of characters which the users may not know (despite being able to identify the physical device itself).

III.B. Tag-Based Device Discovery

In contrast, by using visual tags to identify devices and services, Alice can simply point their cameraphone at Bob's phone and click in order to establish a connection with it. In our framework Bob's cameraphone has a tag attached to it: either physically printed on the device itself, or displayed on the device's screen (perhaps as the default wallpaper). Then, to transfer the photo, Alice performs the following sequence of actions:

Select photo: As before, Alice uses the menu system on their phone to select the picture to transfer via Bluetooth.

Select Physical Device: After selecting a picture to transfer, Alice aims their cameraphone at the tag on Bob's cameraphone and presses the select button on their phone's keypad.

The tag is decoded yielding the Device Address of Bob's cameraphone. A Bluetooth connection is initiated immediately and the selected picture is transferred. By encoding Device Addresses in visual tags, we provide two significant benefits to the user:

1. The time taken to establish the connection and initiate the picture transfer is an order of magnitude faster than performing Device Discovery (see Section VI).

2. The Bluetooth Device Name is abstracted from the user.

In many ways the use of visual tags in our system is analogous to the use of hyperlinks in hypertext documents. Just as hyperlinks hide URLs from readers,

Figure 1: [*Left*] A Visual Tag encoding a 48-bit BD_ADDR and 15 bits of application-specific data; [*Right*] Using the tag reader on a Nokia 3650 camera-phone.

we use tags to hide device naming information from users. Using our technology, a user can focus on performing the task in hand—connecting two physical devices to access a mobile service—rather than having to establish a connection via (arbitrarily chosen) device names.

IV. System Architecture

In this section we focus on technical details, describing our implementation of a tag-based service selection and connection establishment protocol. We start by describing the format of the visual tags themselves (Section IV.A) and outlining the implementation of our phone-based tag reading software (Section IV.B). In Section IV.D, we describe how visual tags are used to facilitate Bluetooth device and service selection.

IV.A. Tag Format

Initially based on TRIP [4] and SpotCodes [13], our visual tags are circular barcodes with 4 data-rings and 21 sectors. A visual tag, large enough to be detected by our cameraphone tag reading software, is shown in Figure 1[*left*]. Bits in the data rings are encoded by using a white block to indicate a value of 1 and a black block to indicate a value 0. The outermost ring contains a *sync-marker* (the bit pattern 11), used to specify the orientation of the tag, and a 5-bit checksum which is used to detect decoding errors. The checksum is encoded using the bit pattern 00 to represent a 0 and the bit pattern 10 to represent a 1. This ensures that the sync marker is uniquely identifiable by preventing the bit pattern 11 from occurring anywhere else in the outer ring. The checksum is encoded in a clockwise direction with the most significant bit located directly after the sync-marker. In the 3 inner data-rings, bits are assigned to sectors radially, outside-in and in a clockwise direction. The most significant bit of data is located in the outer-most data ring of the sector below the first bit of the sync-marker. Of the 63 bits of data encoded in each tag, 48 bits specify a Bluetooth Device Address (BD_ADDR) and the remaining 15 bits are used for application-specific purposes.

Our circular tags are designed to work well with the low-resolution fixed-focal-depth cameras found on today's cameraphones. Crucially, the individual bits are large enough to be reliably decoded from distances at which the camera's lens gives a sharp image. In contrast we found that familiar linear (e.g. UPC) barcodes are unsuitable for decoding on cameraphones: when the phone is positioned near to a linear barcode, the image is out of focus and cannot be decoded; at distances large enough to capture sharp images, the resolution is insufficient to make out individual bars.

IV.B. Tag Reader

We have implemented tag reading software for Nokia Series 60 cameraphones. When running, the reader turns the cameraphone's display into a *viewfinder*: a live video feed continually capturing frames from the phone's embedded camera. The phone performs real-time image processing, locating and decoding tags in every frame. Whenever a tag is visible in the viewfinder and has been decoded successfully, it is highlighted with a red cross-hair on the phone's display (see Figure 1[*right*]). If multiple tags are detected in a single frame, the tag nearest the center of the viewfinder is highlighted with a red cross-hair. We adopt standard GUI terminology, saying that the highlighted tag *has the focus*. Pressing the select button on the phone's keypad indicates that the user wishes to establish a connection to the mobile service specified by the highlighted tag.

The cameraphone-based tag reading software is implemented as a native Symbian-OS application, written in C++. When running on Nokia Series 60 cameraphones the tag reading software runs at 15 frames-per-second with very low latency. We have found that a high frame rate and low latency is essential to usability. Without these properties a user finds it difficult to aim the cameraphone at a particular tag.

IV.C. Connection Establishment

Once a tag containing a device's BD_ADDR has been decoded by our tag recognition software, Bluetooth connection establishment can be performed in the usual way. For example, in the image-transfer sce-

nario of Section III, the sending phone would proceed as usual by: (*i*) connecting to the device specified by the BD_ADDR (decoded from the tag); (*ii*) performing service discovery to find the receiving phone's OBEX server [1]; and (*iii*) connecting to the OBEX server and sending the picture via an OBEX PUT. Thus we see that tag-based device discovery is a component that can be seamlessly slotted into existing applications, replacing the standard Bluetooth device discovery model whilst leaving the rest of the application's Bluetooth interactions unchanged.

IV.D. Tag-Based Content Selection

In Section I we hinted at a publicly accessible mobile service that enables users to download music clips onto their PDA or phone. In this section we consider how applications such as this may be realized in practice.

The basic principle is that a single mobile service can expose multiple choices to a user by means of multiple tags. In the case of the music download service, for example, the user would be presented with a number of tags—one for each track. The user simply clicks on the tag corresponding to their preferred track and it is downloaded onto their phone.

Recall that, as well as encoding a 48-bit BD_ADDR, our visual tags also have room for 15-bits of application-specific data. To facilitate tag-based content selection, we use these application-specific bits to encode a unique identifier. In the case of the music download service, each tag encodes the same BD_ADDR (since all the tracks are accessed by means of a connection to the same physical device). However, each track's tag has a different bit pattern in its application-specific data-block. Once a connection has been established to the device specified by the decoded BD_ADDR, the application-specific bits are used to specify which track to download.

We have implemented a mobile phone application that facilitates tag-based content/service selection. The application incorporates the tag reading software and begins by prompting users to select a tag (as described in Section IV.B). Once a tag has been selected the application extracts the BD_ADDR and attempts to establish a Bluetooth connection with the mobile service.

The mobile service itself runs a simple HTTP daemon. Once the phone has connected to the remote device, it uses the standard Bluetooth Service Discovery Protocol (SDP) to locate and connect to the HTTP server: first a Service Search Request is performed, yielding the service-handle corresponding to the web-server; secondly a Service Attribute Transaction is used to retrieve the Protocol Descriptor List, providing the phone with the protocol and channel number over which the web-server is running. Our current implementation assumes that the web-server will be running over Bluetooth's RFCOMM layer (a transport protocol which emulates an RS232 serial port) [1].

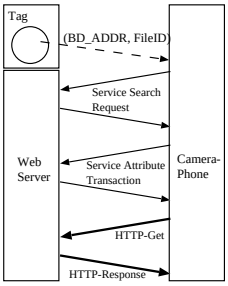


Figure 2: A diagrammatic view of the data-transfers involved in our tag-based content browsing application.

When the phone has established a Bluetooth connection to the web-server it sends an HTTP-GET message, specifying the application-specific bits decoded from the selected tag as part of the requested URL. In return, the web-server sends an HTTP-response containing the requested file. The phone stores the received file in the usual way and possibly performs an action on it (e.g. executes it) depending on the phone's security settings and the file's MIME type.

The complete transaction, from clicking on a tag to receiving the requested file on the phone, is shown diagrammatically in Figure 2. Data transmitted visually by means of a tag is indicated with a dashed line; solid thin lines show Bluetooth Service Discovery (SDP over L2CAP); solid thick lines show the content transfer itself (HTTP over RFCOMM). Note that using HTTP to transfer files to the phone is by no means the only the option. We could, for example, have used Bluetooth's Object EXchange (OBEX) protocol [1] to equally good effect.

V. Usage Scenarios

In this section we describe two realistic usage scenarios that highlight the advantages of using visual tags to access Bluetooth mobile services. The applications described have both been implemented.

V.A. Stock Querying

The simple stock querying mobile service is designed for deployment in a large shop (e.g. a DIY store or supermarket). Figure 3 shows the architecture of the

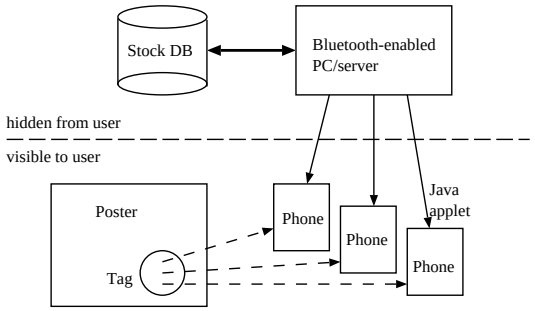


Figure 3: Architecture of the stock-query mobile service.

system. On entry to the shop, the user sees a large poster advertising the mobile service. They run our tag-based content browser on their cameraphone and click on the tag displayed on the poster. Using the connection establishment and content delivery mechanism described in Section IV.D, a Java applet[2] is downloaded to the phone (from a nearby Bluetooth-enabled web-server) and executed. The Java applet displays a combo-box on the phone's display with which a user can enter/select an item in the stock database. On selecting an item, the applet displays how many are currently in stock and the number of the aisle in which the product is located.

In our current implementation the Java applet is self-contained, incorporating a snapshot of the stock database; with careful compression, thousands of products can be stored in the applet. Every five minutes or so, the PC recompiles a new applet containing up-to-date information derived from the stock database. The main advantage of adopting this mobile-agent approach is that once the Java applet is downloaded onto the phone there is no need for further network connectivity; queries are performed entirely on the phone. This allows the user to walk freely around the shop, performing product queries on their phone without having to worry about remaining within Bluetooth range of the mobile service.

The benefit of our visual-tag technology in this application is the speed and simplicity with which a Bluetooth connection may be established. If the mobile service was accessed via the standard Bluetooth device discovery model (presumably by printing a Bluetooth Device Name on the poster and providing instructions describing how to connect to it) the user would be exposed to the issues described in Section III. The key is that, in this scenario, a user knows exactly which service they want to connect to—that is, the one advertised on the poster. By clicking on a visual tag, the user is able to communicate this contextual information directly to their phone, saving them time (by eliminating slow Bluetooth Device Inquiry) and making the system as a whole more usable (since the user is spared from having to navigate a potentially long list of Bluetooth Device Names on their phone's small display).

[2] Java pedants may point out that, technically, we should really use the term "Midlet" here.

V.B. Mobile-Game Vending Machine

The mobile-game vending machine allows users to browse a selection of Java games and download selected games onto their cameraphone. The vending machine consists of a Bluetooth-equipped PC connected to a flat panel LCD display. In a realistic deployment, the PC would be hidden from the user with the display mounted in a prominent position and shielded from damage behind a sheet of glass.

The LCD display shows a selection of animated clips of mobile phone games with a tag next to each one. To select a game and download it onto their phone the user runs our tag-based content browser and uses their cameraphone to click on the appropriate tag.

The Bluetooth-equipped PC that runs the vending machine application may be connected to the Internet. This allows the company responsible for maintaining the machine to connect remotely and dynamically update the selection of games offered.

Whilst the vending machine example raises the issue of payment, this is not something we want to discuss at length in this paper. There have been a great many proposals for secure mobile payment schemes, any of which could be integrated into our vending machine application. The purpose of this case study is twofold: firstly, it demonstrates the advantages of using tags for content selection (as distinct from device selection); secondly, by using tags on an active display we make it very easy for the vending machine company to update the machine with the latest games.

To expand further on the first of these points, consider designing a mobile-game vending machine without using tags. In this case the user would have to (*i*) connect to the service using the standard Bluetooth Device Discovery model (exposing the user to the issues described in Section III); and (*ii*) select a specific game to download[3]. Thus we see that, in this scenario, the tags offer a dual benefit. Not only do they elimi-

[3] There are a number of ways in which a user could select a game. If a touchscreen is available, for example, a user could select a game by pressing the screen with their finger. If no input devices are available, however, a user could select from a list of game titles displayed on their cameraphone's screen. This second scenario is explored further in Section VI.

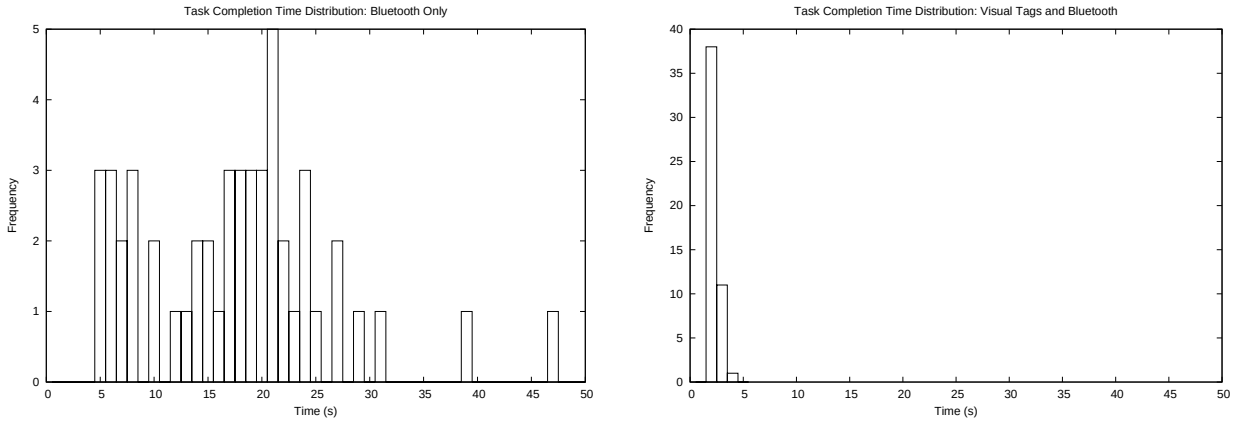


Figure 4: [*Left*]: Distribution of times taken to complete experimental task using Bluetooth alone (50 trials); [*Right*]: Distribution of times taken to complete experimental task using visual tags (50 trials). Both figures are histograms with a bin-size of 1s.

nate the time-consuming Bluetooth device discovery and selection process, they also provide the user with an easy way to tell the vending machine which game they require.

VI. Performance Evaluation

We compared the performance of our tag-based device/service/content selection technique against the standard Bluetooth Device Discovery model. As a test application we designed two versions of a Bluetooth-based active poster designed to distribute mobile content to a user's cameraphone/PDA. The first version of the poster contains six tags, each advertising a different piece of content[4]. Clicking on a tag initiates a Bluetooth connection and downloads the requested content onto the user's device.

The second version of the poster does not make use of visual tags; instead, it contains instructions explaining how to select and download content using the standard Bluetooth Device Discovery model. To facilitate interaction with this poster we implemented a tagless version of our mobile service/content browser. This application (*i*) performs standard Bluetooth Device Discovery, prompting the user to select which device they want to connect to; (*ii*) connects to the specified device and queries it for a list of available services/content; (*iii*) displays the list of services/content on the phone's display, prompting the user to select one; and (*iv*) downloads the selected content from the mobile service onto the user's device. The name of the Bluetooth device providing the service ("test-service-01") is displayed clearly on the poster so the user knows which of the discovered devices to connect to.

To ensure that our experiment is a fair comparison it is essential that we use "best known methods" for Bluetooth Device Discovery in our tagless service browser. For this reason we implemented the application directly on top of the cameraphone's native OS, Symbian, and leveraged Symbian's existing Bluetooth Device Discovery dialog. The Symbian dialog incorporates a number of features designed specifically to reduce the device discovery latency experienced by the user: (*i*) Name Discovery is performed together with Inquiry, with names being resolved whilst other BD_ADDRs are still being discovered; (*ii*) device names are displayed on the phone's screen *as soon as they are discovered*—if the device the user is looking for happens to be found early on in the discovery procedure they have the option to abort device discovery immediately and select their required device.

The experiments were conducted using Nokia 3650 and 3660 cameraphones and a Bluetooth-enabled desktop PC (Linux 2.4.20-28.9 running the Bluez Bluetooth stack) to simulate the mobile service. Using both tag-based and tagless versions of the poster each piece of content was downloaded 50 times.

In the case of the tag-based poster we measured (*i*) the time between running the tag-reading application and the user selecting the tag; (*ii*) the time to initiate an L2CAP connection and perform service discovery (using standard Bluetooth SDP); and (*iii*) the time taken to establish an RFCOMM connection with the mobile service (once SDP has revealed the channel on which to connect).

In the case of the tagless poster we measured (*i*) the time between starting Bluetooth device discovery and

[4]For the purposes of the experiment it doesn't matter what the content is. In a real-life deployment content may be Java applets, music clips, HTML pages etc.

the user selecting the correct device from the resulting list; (*ii*) the time taken to establish an L2CAP connection and perform SDP; (*iii*) the time taken to establish an RFCOMM connection; and (*iv*) the time taken to display the list of available downloads on the phone's screen and for the user to select one.

A group of four users performed the content downloads. We made sure that the users were familiar with both Bluetooth and the UI of the Nokia cameraphones before starting the experiment. Between each run of the experiment we performed a hard reset of the cameraphone. This ensured that no ⟨BD_ADDR, device name⟩ pairs had been cached by the phone, simulating the scenario where the devices had not seen each other before (as is often the case when accessing public mobile services). The experiments were all conducted in an office environment containing 6 discoverable Bluetooth devices (mobile phones and PCs) and several 802.11-enabled PCs. We believe that this environment accurately models places were mobile services are likely to be deployed (e.g. airports, coffee shops).

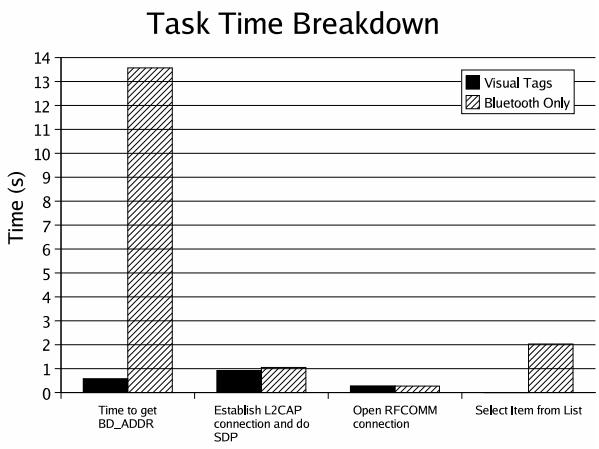
In the remainder of this section we present the results of our experiments, using them to evaluate our framework against three separate criteria: speed, usability and scalability.

VI.A. Speed

Figure 4 compares the distributions of the times taken to establish a Bluetooth connection using both our visual-tag framework Fig. 4[*Right*] and the standard Bluetooth version of the service browser Fig. 4[*Left*]. It is clear to see that using visual tags to perform out-of-band device and service selection offers a significant benefit to users. With the tag-based content browser users completed the task in an average of 1.77s (standard deviation 0.47s). Using the standard Bluetooth version of the app the task was completed in an average of 17.12s (with a standard deviation of 8.6s).

This result can be explained with reference to Figure 5, which shows a breakdown of the individual subtasks involved in performing the content download task. As expected, the major difference between the two systems is in the time taken to retrieve the BD_ADDR of the mobile service: here the visual-tag based system, which retrieves its BD_ADDR by decoding a tag, outperformed the standard Bluetooth Device Discovery model on average by a factor of 24.

Recall that the Bluetooth specification quotes the *worst case latency* of inquiry time as 10.24s (see Section II). In reality our experiments show that users



	Visual Tags		Bluetooth Only	
	$\bar{x}$ (s)	σ (s)	$\bar{x}$ (s)	σ (s)
Get BD_ADDR	0.56	0.19	13.57	8.21
L2CAP/SDP	0.92	0.39	1.05	0.60
RFCOMM	0.28	0.10	0.27	0.13
List Selection	0.00	0.00	2.03	0.45

Figure 5: Comparing the time breakdown of the subtasks performed across the two systems. (The table and graph present two different views of the same data.)

experience an *average latency* greater than this. The reasons for this disparity are twofold. Firstly, the specification's worst case latency is a theoretical result quoted assuming an error-free environment; secondly, it does not include the time to perform name discovery (required to display Bluetooth device names on the phone's screen).

In contrast, the average time to click on a tag using the cameraphone is very small: 0.56s (standard deviation 0.19s). The reason this time is so small can be explained by noting that the phone takes about 3s to load and initialize our service browser applications. (This setup time, deliberately excluded from our measurements, is the same for both the tag-based and tagless versions of the software; indeed, all applications running on the phone incur similar start-up latency.) During this 3s setup delay the user has ample time to aim the camera at their chosen tag. As soon as the tag-reader starts up, the tag is immediately detected (highlighted with a red cross-hair) and the user clicks[5].

To gauge the magnitude of this effect we ran another experiment in which users were instructed not to start aiming the cameraphone at a tag until the

[5]We refer the interested reader to our downloadable implementation [14] in case they would like to verify this result for themselves.

application had initialised. Under these conditions the average time taken to click on a tag (and for the BD_ADDR to be decoded) increased to 1.5s. On this basis we conclude that, even on devices with negligable application initialisation delay, our results still hold.

The times taken for the two systems to establish an L2CAP connection are very similar. Given that, as part of Inquiry, a device learns clock-synchronization information from discovered devices [1], this result is surprising. One would expect the Bluetooth-only version of the service browser to perform better in this respect (due to the fact that it performs Inquiry immediately before connection establishment). It could be that the Symbian Bluetooth stack does not use clock-synchronization information learned during Inquiry as efficiently as it could. However, since the time to establish an L2CAP connection (in both cases) is small compared to the time to perform Device Discovery, our main result (order of magnitude improvement in task completion time) continues to hold even if we drop the L2CAP connection time in the "Bluetooth Only" case to 0.

The other major difference between the two systems is that the tag-based version of the application does not require the user to select the content to download from a list. Recall that service selection is performed by clicking on 1 of the 6 different tags on the poster. The single tag click serves both to initiate the Bluetooth connection and specify which content is requested.

VI.B. Usability

For the purposes of this section, we take the average number of button presses required to complete the task as a crude measure of usability [17]. The tag-based system requires only one button press (to notify the system that the required tag has been acquired). Once a tag has been selected there is no further input required from the user. In contrast, the standard Bluetooth version of the application requires an average of 10 button presses to navigate two separate list controls: one containing discovered Bluetooth Devices, the other containing a list of items that can be downloaded.

We have conducted an in-depth study of usability issues surrounding visual tag-based applications. This work has been written up in a separate publication [18].

VI.C. Scalability

The standard Bluetooth version of the application scales badly both as a function of the number of discoverable Bluetooth devices in the neighborhood and also as a function of the number of options provided by a mobile service. Increasing both these parameters leads to longer lists for users to select from, increasing the number of button presses required to make selections and thus decreasing the usability of the system.

Furthermore, increasing the number of Bluetooth Devices in the neighborhood causes Device Discovery time to increase as $O(n)$. This is not due to the inquiry phase itself but to the fact that name discovery must be performed for each of the BD_ADDRs discovered by inquiry.

VII. Security Issues

Although security is not the main thrust of this paper, in this section we briefly consider the two main security issues arising from accessing mobile services using visual tags and Bluetooth.

VII.A. Authentication

A common concern regarding the use visual tags to establish connections to mobile services is the "Man-in-the-Middle attack". In this scenario, an attacker somehow replaces a visual tag on (say) an active poster with a new tag that points to their malicious device. Users that click on the tag will then unwittingly connect to the malicious service rather than the genuine one.

The Man-in-the-Middle attack is a well studied problem that must always be considered when developing connection-establishment protocols. There has been a great deal of work on using digital signatures for secure authentication, preventing Man-in-the-Middle attacks [16]. We propose using similar schemes when establishing Bluetooth connections to mobile services. For example, on connection, a mobile service could send a message containing a declaration of their identity and their public key. This message would be signed by a trusted party (e.g. VeriSign). When the phone receives this message from a mobile service, the signature would be verified on the phone (using the public key of the signing authority) and a message displayed on the phone screen displaying the name of the service provider—e.g. "This mobile service is provided by ServerCorp, do you wish to continue?". The user could then decide whether to continue to access the service, depending on whether they trust ServerCorp. This is exactly

the same model used to provide secure authentication for all e-commerce transactions that take place on the web.

VII.B. Privacy and Encryption

There are a number of reasons why one may want a user's interactions with a mobile service to remain private. Consider, for example, the issue of downloading paid content (e.g. the mobile-game vending machine example of Section V.B). In this instance the suppliers of the vending machine want to ensure that an eavesdropper that has not paid for the content cannot obtain a game for free by snooping another customer's Bluetooth transaction.

The obvious solution is to apply Bluetooth's connection-level encryption [1]. If the system designer does not feel that this is satisfactory we note that there is a huge body of literature on session-key establishment and encryption [16]. These standard techniques can be applied to our HTTP/RFCOMM Bluetooth sessions. In a realistic environment, tried and tested protocols such as HTTPS could be employed to provide both authentication and privacy guarantees. Most of today's mobile phones and PDAs already have built-in support for HTTPS.

VIII. Discussion and Related Work

We are not the first to realize that there are a number of scenarios in which the Bluetooth Device Discovery model is less than ideal. Motivated by a desire to eliminate the time- and power-costs associated with Bluetooth Device Discovery, Hall *et al* propose using RFID tags to "wake-up" a sleeping Bluetooth radio and transmit a device's BD_ADDR [7]. Our implementation achieves the same time and power benefits as Hall but also also offers a number of distinct advantages. Firstly, since we rely entirely on commodity hardware, our work is immediately applicable: there is no need to augment existing devices with RFID readers[6], instead our implementation can be downloaded and immediately deployed on the hundreds of millions of cameraphones and camera-enabled PDAs that have already shipped globally. Secondly, the combination of visual tags on an active display provides a functionality that cannot be duplicated by RFID—a PC can generate tags on-the-fly, *dynamically* updating the list of services offered to users (see Section V.B). Thirdly, our work considers both Bluetooth device discovery and service selection whereas Hall *et al* consider only the former.

At this point it is worth drawing one further comparison between RFID technology and visual tags that we believe to be important. Whereas RFID tags are typically small embedded devices hidden from human view, optical tags are, by their very nature, visible to users. This raises an important question regarding RFID-based device and service discovery: how does a user know to hold their mobile-device against an (invisible) RFID tag? Clearly the RFID solution also requires some kind of visual symbol or printed text in order to alert the user to the existence of the mobile service. In contrast, visual tags are a single technology serving the dual purposes of (*i*) encoding machine-readable data; and (*ii*) advertising the existence of mobile services to a human users.

Woodings *et al* have proposed using IrDA to bypass Bluetooth Device Discovery [19]. In their system, an IrDA connection is established and used to transmit the BD_ADDR of a remote device. Once the BD_ADDR has been transferred, the IrDA connection is closed and a Bluetooth connection is initiated. Woodings shows that the resulting time taken to establish a Bluetooth connection when using IrDA in this fashion, is several times faster than relying on the standard Bluetooth Device Discovery model. In many ways, the comparison between our work and this research is similar to the comparison between our work and RFID-based Bluetooth connection establishment procedure outlined above. Firstly, we have shown that visual tags provide a convenient mechanism for both device discovery and service selection. Whilst IrDA allows Bluetooth device discovery to be bypassed, it is unclear how the technology could also be used for service selection. Secondly, as with RFID tags, some kind of visual symbol or text would be required anyway to alert the user to the presence of a Bluetooth/IrDA mobile service. This is something that visual tags provide for free.

The performance evaluation sections of both Hall's [7] and Woodings' [19] papers are weak, relying on numbers taken directly from the Bluetooth specification, combined with results derived by experimentally testing subcomponents of the proposed systems *in isolation*. To our knowledge, our paper is the first attempt to quantify the benefits of out-of-band Bluetooth connection-establishment as measured by real users working with a real end-to-end systems im-

[6] Whilst some manufacturers are incorporating RFID readers into mobile devices, there are currently no such products in the *consumer* market space. Current offerings are targetted at specific industrial applications (e.g. ruggedized phones for field engineers).

plementation in a realistic environment.

Many other researchers have noticed the potential of the camera as an appealing input medium for interaction. Much of the work in this area involves fixed cameras trained on users [6]. This approach, pioneered by VideoPlace [12], has proved fruitful in areas such as gesture recognition [5] and location-aware computing [9]. In contrast, cameraphones are more suited to the approach often taken by the augmented reality community in which hand-held cameras are pointed at objects in the world. A variety of augmented reality systems based on optical tags have already been developed [10]. These systems influenced our design to a large extent.

Near Field Communication (NFC) [2] is a wireless networking technology that operates over ranges of a few centimeters. Like RFID tags, NFC has also been proposed as a mechanism for quickly establishing other types of wireless communication between devices (e.g. 802.11 or Bluetooth). Although there is a great deal of industrial interest in NFC, the technology is still in its infancy and is yet to achieve acceptance in the marketplace.

A number of projects have explored the use of tags to associate physical objects with URLs. A pioneering example is HP's CoolTown project [11] which considers a variety of different tagging mechanisms, including barcodes. The contribution of our work in this area is (*i*) the application of the technique to Bluetooth device discovery and service selection; and (*ii*) a robust tag-based connection-establishment implementation that runs on commodity hardware.

The circular tags used in this paper were initially inspired by TRIP [4] tags. For the purposes of this research we made a number of changes from the original TRIP specification. In particular: (*i*) we increased the number of bits encoded in a single tag (by adding extra data rings); and (*ii*) added an explicit *sync-marker*, rather than taking the TRIP approach of reserving particular bit-patterns within the data bits themselves to specify tag orientation. Our 4-ring tags are also closely based on the authors' previous work on SpotCodes [13]. Indeed, the cameraphone reader for the 4-ring tags described in this paper exploits many of the same image-processing techniques as the 2-ring SpotCode reader.

Semacodes are square 2D barcodes designed for encoding URLs that can be read by cameraphones [3]. A cameraphone-based application is available that reads a Semacode, extracts the encoded URL and downloads the corresponding web-page using the phone's built-in web browser. Again, the novelty of our work here is the application of visual tags to a different domain area—namely, Bluetooth device discovery and service selection. In the early stages of this research, we evaluated the Semacode platform as a possible underlying tag-technology for Bluetooth connection-establishment and found it unsuitable, primarily due to the unreliability of the reader implementation. Furthermore, we noticed that, due to a lack of real-time feedback, users found it difficult to click on Semacodes—the reader software simply reported a "decode failure" for more than 30% of attempted tag decodes. Whilst we recognize that other tag formats could be used to facilitate Bluetooth connection establishment, the lack of reliability and robustness of existing offerings forced us to implement our own custom solution.

Patel and Abowd have proposed augmenting handheld devices (e.g. PDAs and mobile phones) with lasers [15]; data modulated over the laser is used to trigger a photosensitive (active) tag in the environment. The advantages of our work over this are (*i*) we use commodity hardware; (*ii*) we can interact with both active tags and passive tags; and (*iii*) unlike photosensitive tags, which require physical hardware, computers can generate an arbitrary number of visual tags dynamically, writing them to a display.

We do not claim that our tag-based connection-establishment procedure makes the standard Bluetooth Device Discovery procedure completely redundant. Indeed, there are a large class of applications where a user wants their phone to periodically perform device discovery, sometimes connecting to Bluetooth services without even leaving their pocket (e.g. automatically synchronizing email when a user is in proximity of their PC). In this scenario, Bluetooth's Device Discovery and SDP provide just the right functionality. Our work specifically targets one particular use-case: the user-driven connection of a mobile device to a publicly accessible mobile service. In this scenario we believe our approach delivers real benefits.

IX. Conclusions and Further Work

We have shown that visual tags can be used to initiate Bluetooth connections with mobile services, providing order-of-magnitude time improvements over the standard Bluetooth Device Discovery model. Furthermore, we have shown that tags can be used for service selection, providing users with a convenient way to choose from a selection of content downloads or mobile services.

One of the strengths of our approach is that it

relies entirely on commodity hardware. Hundreds of millions of Bluetooth-enabled cameraphones and camera-enabled PDAs have already been shipped globally [8], with market trends showing that the rapid adoption of such devices is set to continue for the foreseeable future. Our Nokia Series 60 cameraphone implementation is available for download [14].

We do not believe that printed tags are competing with RFID and NFC for the prize of "universally accepted connection establishment technology". Instead we observe that each offers *complementary* tradeoffs in terms of cost, data capacity, interaction distance, client-device compatibility and visibility. We predict that all three of these technologies will ultimately be integrated into mobile applications to provide consumers with the flexibility and functionality they require.

The research issues surrounding the development of mobile services draw on a variety of computer science disciplines including networking, security, human-computer-interaction and ubiquitous computing. We strongly believe that, when developing frameworks for mobile services, the computer science researcher should "design-in" relevant ideas from each of these areas from the outset; only then will we develop systems that are secure, reliable, practical and, equally importantly, usable. We have attempted to embody this principle in our research by combining traditional network-based service discovery protocols (Bluetooth's SDP) with concepts inspired by human-computer-interaction research and context-aware computing (visual tags).

References

[1] Bluetooth Specification version 1.1. `http://www.bluetooth.com/`.

[2] NFC white paper. ECMA International. Available from `http://www.nfc-forum.org/`.

[3] Semacode website. `http://semacode.org/`.

[4] Diego López de Ipiña, Paulo Mendonça, and Andy Hopper. Trip: a low-cost vision-based location system for ubiquitous computing. *Personal and Ubiquitous Computing*, 6(3):206–219, May 2002.

[5] K. Dorfmueller-Ulhaas and D. Schmalstieg. Finger tracking for interaction in augmented environments. In *Proc. IEEE International Symposium on Augmented Reality*. IEEE Press, 2001.

[6] Jerry Fails and Dan Olsen. A design tool for camera-based interaction. In *Proc. CHI*, pages 449–456. ACM Press, 2003.

[7] Eric S. Hall, David K. Vawdrey, and Charles D. Knutson. RF Rendez-Blue. In *Proceedings of the 11th International Conference on Computer Communications and Networks*. IEEE, 2002.

[8] IDC Research. Moving pictures 2003: Worldwide camera phone survey, forecast, and analysis, 2003-2007.

[9] Diego Lopez de Ipina, Paulo Mendonca, and Andy Hopper. Trip: a low-cost vision-based location system for ubiquitous computing. *Personal and Ubiquitous Computing*, 6(3):206–219, 2002.

[10] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *International Workshop on Augmented Reality (IWAR)*, pages 85–94, October 1999.

[11] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, John Schettino, Bill Serra, and Mirjana Spasojevic. People, Places, Things: Web Presence for the Real World. In *Proceedings of the third IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'00)*. IEEE Press.

[12] Myron W. Krueger, Thomas Gionfriddo, and Katrin Hinrichsen. Videoplace—an artificial reality. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press, 1985.

[13] High Energy Magic Ltd. The spotcode framework. Available for download from: `http://www.highenergymagic.com/`.

[14] High Energy Magic Ltd. Visual tag-based bluetooth connection software download. Available from: `http://www.highenergymagic.com/btTag/`.

[15] Shwetak N. Patel and Gregory D. Abowd. A 2-way laser-assisted selection scheme for handhelds in a physical environment. In *Proceedings of The Fifth International Conference on Ubiquitous Computing (UbiComp 2003)*. Springer-Verlag.

[16] Bruce Schneier. *Applied cryptography: protocols, algorithms, and sourcecode in C*. John Wiley and Sons, New York, 1994.

[17] Harold Thimbleby. Formulating usability. *SIGCHI Bulletin*, 26(2):59–64, 1994.

[18] Eleanor Toye, Anil Madhavapeddy, Richard Sharp, David Scott, and Eben Upton. Using cameraphones to interact with context-aware mobile services. 2004. To appear.

[19] R.W. Woodings, D.D. Joos, T. Clifton, and C.D. Knutson. Rapid heterogeneous ad hoc connection establishment: accelerating bluetooth inquiry using IRdA. In *Proceedings of the Wireless Communications and Networking Conference (WCNC2002)*, pages 342–349. IEEE.