Bluetooth Tracking without Discoverability

Simon Hay and Robert Harle

University of Cambridge Computer Laboratory
{sjeh3,rkh23}@cam.ac.uk

Abstract. Outdoor location-based services are now prevalent due to advances in mobile technology and GPS. Indoors, however, even coarse location remains unavailable. Bluetooth has been identified as a potential location technology that mobile consumer devices already support, easing deployment and maintenance. However, Bluetooth tracking systems to date have relied on the Bluetooth inquiry mode to constantly scan for devices. This process is very slow and can be a security and privacy risk. In this paper we investigate an alternative: connection-based tracking. This permits tracking of a previously identified handset within a field of fixed base stations. Proximity is determined by creating and monitoring low-level Bluetooth connections that do not require authorisation. We investigate the properties of the low-level connections both theoretically and in practice, and show how to construct a building-wide tracking system based on this technique. We conclude that the technique is a viable alternative to inquiry-based Bluetooth tracking.

1 Introduction

Location tracking is fast becoming an essential service for mobile devices. Underpinning this trend is the ubiquity and accuracy of GPS. However, GPS continues to struggle indoors due to the failure of satellite signals to penetrate buildings. To address this shortcoming, there have been many attempts at indoor location tracking, achieving a wide range of accuracies [1]. None of the proposed solutions have been widely adopted, primarily due to the cost of deploying and maintaining building-wide location technology. We have therefore become interested in finding a reliable location technology that is viable for quick deployment across entire buildings using standard infrastructure. This rules out fine-grained tracking systems such as the Bat system [2,3] since these require the retro-fit of dedicated infrastructure and the need to issue building occupants with custom tracking devices.[1]

From our experiences with indoor location in many forms over many years, we know that even coarse location can provide useful location-aware applications. The Active Badge project [4] was adopted by many at the University of Cambridge because the room-level location it provided was sufficient to enable a key

[1] Our experience with the Bat system is that users tend to forget to wear their Bats. Additionally, there is a constant maintenance task in keeping Bats operational (especially with respect to battery power) and keeping the model of the world up-to-date.

T. Choudhury et al. (Eds.): LoCA 2009, LNCS 5561, pp. 120–137, 2009.

application: finding co-workers quickly. This simple application demands only that users can be reliably located with room-level accuracy. The usage of that system waned because users forgot to carry their badges, didn't take responsibility for replacing batteries, and found that tracking failures were common (badges were easily obscured by clothing).

We are seeking a technique that will permit room-level tracking, both for colleague searching and for building optimisation. With the latter, we are particularly interested in dynamically optimising heating, cooling and computing devices based on room usage — see [5] for more details. To achieve this we must satisfy the following goals:

- The system must be able to locate users to a small area, ideally placing them in a single room.
- The system must be able to track users as they move to permit dynamic optimisation on timescales of the order of seconds.
- The system must be able to track multiple users simultaneously.
- The system must be easily adopted by a very high percentage of building users and it must avoid the issue of users forgetting to enable tracking in some way.

In order to meet the final criterion, we wish to derive indoor location from the mobile telephones that people carry. These devices are ubiquitous in many societies, feature an increasing array of communications technologies, are carried everywhere by their owners and responsibility for charging them lies with the user.

To obtain in-building tracking using unmodified mobile telephones, we wish to overlay location tracking on established indoor communications technologies. Such tracking has been demonstrated using signals from the mobile telephony networks, using WiFi and using Bluetooth. At present, positioning from the telephony networks is too coarse for in-building location [6]. WiFi-based tracking has received much attention in recent years [7,8], but it is not ideal for general tracking. It is associated with high power demands, difficulty in set up and maintenance, small market penetration of suitable (WiFi-enabled) handsets and difficulty in convincing users to leave handset WiFi turned on.

Bluetooth has more potential. It was designed for low power devices and almost all deployed handsets support it. For this reason, it has been a popular choice for other location tracking projects [9,10,11,12,13,14,15,16,17]. These projects all locate mobile devices based on the results of constantly scanning for mobile devices from fixed devices ('beacons'), or continually scanning for beacons from mobile devices. The disadvantages of tracking by continual scanning are studied in detail in the following section, but the key issues are the length of time each scan takes (up to 10.24s) and the requirement for the device being scanned to be discoverable (this is a serious privacy risk in the beacons-scan-handsets scheme [18] and is not even possible on many of the latest phones).

In this paper, we explore a novel way to track almost any Bluetooth device (following a one-off registration) without making it discoverable. The remainder of the paper looks at quantifying the properties of Bluetooth that we exploit

when tracking, before going on to consider the architecture of a large-scale tracking system based on the technique.

2 Bluetooth Tracking of Mobile Devices

Most Bluetooth tracking systems are proximity-based [19,20]. That is, if a user can be contacted by a base station, then the user is coarsely located to the base station position. Bluetooth was designed as a short-range communications system with range of comparable size to a room (class 2 Bluetooth devices have a nominal range of 10m in free space, less indoors), so proximity-based location is simple to implement and relatively reliable. Bluetooth tracking has also been demonstrated using the RSS measurement techniques first developed for WiFi location systems [16,21], either based on radio propagation models or 'fingerprinting' (matching current radio conditions to a previously-measured radio map).

As mentioned in the previous section, most of these methods locate a device using the inquiry, or scan, mode of Bluetooth. In this mode, a base station transmits a discovery packet on each of 32 radio channels. Devices set to 'discoverable' respond to this packet, identifying themselves. However, the response follows a random delay in order to minimise the chance of response collisions when multiple devices respond. The result of this protocol is that an inquiry must run for 10.24s to reliably detect all devices in range (and longer still if the radio conditions are unfavourable).

Initially, many systems tracked mobile devices by continually issuing inquiry packets from a network of fixed beacons. This has the advantage that no custom code need be deployed on the handset, but is often perceived as a privacy risk since anyone can track a handset by creating their own network of beacons. Additionally, handset manufacturers are increasing security by changing the 'discoverable' mode to be a time-limited handset state; this is the case for the iPhone and the G1. Thus more recent tracking attempts have concentrated on the mobile handset scanning for the fixed beacons. This is more secure since Bluetooth does not require that a scan packet identify its source address. However, it requires custom application code on the handset, will typically draw more power, and requires a data communications channel to publish positions. Regardless of these issues, both schemes suffer from a series of further problems:

High Tracking Latency. Since each scan can take 10.24s, these systems have a very slow update rate which does not support dynamic tracking.

Devices in the System must be Discoverable. In order for a scan to locate a nearby device, that device must be discoverable. It therefore announces itself to the world, and becomes a target for hackers regardless of whether it is a handset or a beacon.

Connection Disruption. Each scan must flood the Bluetooth channels in turn, disrupting normal Bluetooth communications whilst scanning a channel that is in use.

Ideally, then, Bluetooth tracking would not involve the inquiry process nor require any device to be permanently discoverable.

3 Inquiry-Free Tracking

Inquiry-based tracking detects proximity between handsets and fixed beacons by continually scanning for all devices. We introduce the alternative notion of *connection-based* tracking, whereby two specific devices are characterised as proximate if one can connect to the other i.e. if beacon B can connect to handset H then B and H are proximate.

In many senses this is notionally similar to an inquiry-based system where each inquiry targets a specific device rather than every device. This has the obvious disadvantage that multiple such inquiries would be needed to find a multitude of local devices. However, the specificity also allows devices to be targeted quickly rather than with extended broadcasts i.e. a single inquiry should complete faster, allowing for multiple inquiries.

The key motivation for investigating connection-based tracking is that its Bluetooth implementation can potentially address the inquiry-based shortcomings we have already identified. The remainder of this Section is devoted to characterising the relevant properties of Bluetooth that permit this connection-based tracking. This characterisation is done both theoretically and experimentally. For all testing we used a desktop Linux machine (running Ubuntu 8.04 with the BlueZ 3.26 Bluetooth stack) with an attached class 2 generic Bluetooth 2.0 USB dongle and a range of mobile telephones taken as representative of those available on the market (see Table 1).

3.1 Bluetooth Connections

The Bluetooth standard defines three different types of connection which are layered to form the Bluetooth stack. Only two of the three are appropriate for connection-based tracking. The most fundamental is the Asynchronous Connectionless Link (ACL). No more than one ACL can exist between any two Bluetooth devices at any given time, and it must be established before any other connection can be made. An ACL will disconnect only when no higher-layer connections have existed for a certain time (2s seems to be the preferred value of current stacks).

Directly above the ACL is the Logical Link Control and Adaptation Protocol (L2CAP) layer. This is a packet-based layer that provides guaranteed packet

Table 1. The test handsets used

Handset	Description
T-Mobile G1	Android
Apple iPhone 3G	iPhone OS 2.2
Nokia 6300	Series 40 3rd Edition, Feature Pack 2
Nokia N80	Series 60 3rd Edition (Symbian OS 9.1)

sequencing and a selectable degree of delivery reliability. Once established, an L2CAP connection remains open until either end explicitly closes it, or the Link Supervision Time Out (LSTO) expires. The LSTO is the time for which communicating devices are out of range before the L2CAP connection is destroyed. The default LSTO is 20s in many stacks, but can be configured dynamically per-ACL.

Above L2CAP sits the Radio Frequency Communications (RFCOMM) layer, which is the reliable stream-based protocol used by most Bluetooth applications. It represents the type of connection most people mean by 'Bluetooth connection'. It is almost never feasible to use this layer for connection-based tracking due to authorisation requirements (see the subsequent Section) and we do not consider it in any detail here.

3.2 Connection Authorisation

The biggest obstacle to connection-based tracking using RFCOMM connections is the requirement for explicit *pairing* of every handset with every fixed beacon. The pairing process is governed by the Bluetooth security manager and involves creating a shared secret (a passphrase or PIN) between the two devices. Because each pairing operation requires human input, it is not practical to pair every handset with a large deployment of beacons.

There are some communications services, however, that do not usually require this level of authentication. In particular, the creation of an ACL and a basic L2CAP connection is *almost* universally authorisation-free. Although the resultant connections are limited in use for communications (they support little more than low-level testing) they are sufficient for tracking usage because if they are successfully established, the two devices must be within range of each other. Additionally, the low-level tasks they do support, such as RSSI measurement and L2CAP echo requests (directly analogous to the familiar ICMP ping packet in IP), allow continual monitoring of the connection.

3.3 Connection Time

A connection-based tracking system will constantly make connections between devices. The expected latencies in tracking will be dictated by the connection times, which we seek to quantify here.

Bluetooth uses frequency hopping for channel robustness. Every device continually retunes its transceiver to one of 79 Bluetooth channels. The precise sequence of changes is derived from its address and its local Bluetooth clock. For two devices to communicate they must have the same hopping sequence and phase at any given moment. To reach this state, Bluetooth defines a protocol that the two devices must follow. The protocol is known as *paging* and involves the master device sending search packets ('pages') addressed to the *slave* device until it replies. It is useful to consider the behaviour of the master and the slave separately, and Figure 1 may help in understanding the process.

Slave. The slave device periodically listens for page requests on a particular radio channel for 11.25ms. A total of 32 channels are used for paging, and the

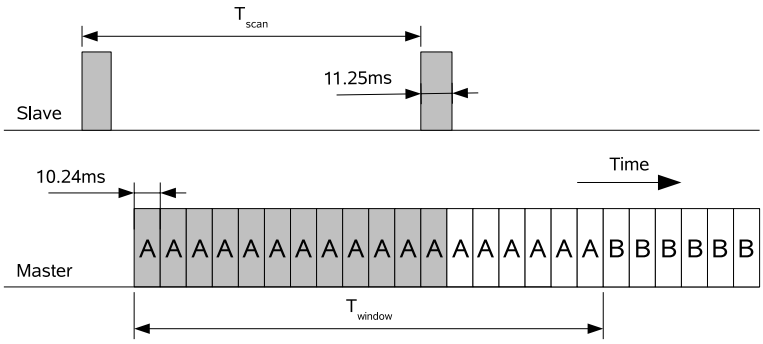

Fig. 1. The paging process. A slave periodically listens on a single frequency for 11.25ms. When paging, the master pages the 16 A frequencies in turn, each cycle taking 10.24ms. After T_{window} seconds, it repeats using the B frequencies. In this example, shaded cycles for the master indicate the cycles needed to connect to the slave shown.

frequency the slave listens on is changed every 1.28s according to a sequence also derived from its clock and its device address. If the slave does not detect a page, it sleeps for set period of time, T_{scan}. The Bluetooth specification defines three *SR* modes which a device can adopt and which provide a limit on T_{scan}. These modes are R0 ($T_{scan} = 0$), R1 ($T_{scan} \leq 1.28$s) and R2 ($T_{scan} \leq 2.56$s). The default mode is R1, but some mobile devices adopt R2 to save power.

Master. With each page sent out, the master needs to estimate the frequency that the slave will be listening on. The device's address tells it the hopping sequence, but it can only know the current sequence position by having an estimate of the Bluetooth clock on the slave. Assume temporarily that it has such an estimate. It then computes the most likely frequency to transmit on. However, in the 11.25ms that the handset listens for, it has time to send pages to 16 channels. Thus it chooses a set of 16 frequencies that are adjacent in the hopping sequence and centred on its best estimate of the listening frequency. This set of 16 frequencies is known as 'train A' and allows for a degree of error in the estimate of the slave's clock. It then cycles over this train of frequencies continuously for some for $T_{window} \geq T_{scan}$ seconds. Assuming the slave is listening on a train A frequency, it will hear the page and respond, implicitly syncing the two devices. On average this should take $\frac{1}{2}T_{scan}$ to complete.

How does the master get that important estimate of the slave's clock? Normally it gets this by completing a lengthy inquiry process. If it does not have an estimate (e.g. inquiry is avoided as in the connection-based technique) it is forced to centre train A on a randomly chosen paging frequency. There is now a 50% chance that the slave will not be listening on a train A frequency. After T_{window} seconds without a reply, the master repeats the entire process

using the other 16 frequencies; 'train B'. On average, the pager will have to use train B half of the time. Thus, the average connection time is

$$< T_{conn} > = \frac{1}{2}.\frac{1}{2}T_{scan} + \frac{1}{2}(T_{window} + \frac{1}{2}T_{scan}) \quad (1)$$

$$= \frac{1}{2}(T_{scan} + T_{window}) \quad (2)$$

and $T_{scan} \leq 2T_{window}$.

So, for default values of $T_{scan} = 1.28$s and $T_{window} = 2.56$s a successful page will take 0.64s on average if we know have a good clock offset estimate between master and slave, and 1.92s on average if not. In the worst possible case, a page should complete within 5.12s, which is still twice as fast as an inquiry.

It should be noted that once a connection is established, the master can cache the clock offset for the slave and use it for subsequent connections to the slave. Over time, however, the clocks drift apart and the estimate will degrade.

None of this analysis, however, incorporates any setup overheads at the master or slave once paging has succeeded. We expect there to be a systematic increase to the connection times. Figure 2 shows the connection times to an Apple iPhone. For each datum, a new connection was created and its creation time measured using the UNIX C function `gettimeofday()` on the connecting system. The start of each connection was spaced by 5000ms, chosen to be indivisible by 1280ms.

Since the iPhone has $T_{scan} = 1.28$s, each subsequent connection attempt would start at a different point in the page scan cycle of the iPhone[2]. Ignoring connection overheads, the connection time should move linearly between zero (connection starts just as the slave's page cycle does) and 1280ms (the connection starts just after the cycle ends). We observed that the experimental connection times incorporate an additional offset of approximately 200ms, which is the connection setup overhead. Additionally, one connection attempt failed altogether, resulting in a connection time of two page slots.

Ideally, connection would be near instantaneous for connection-based tracking. With unmodified handsets, it is realistic to expect connection times to be bounded by 1.28s. Until a connection is established or 1.28s has elapsed, a given beacon will not be able to verify whether or not a handset is in range. The equivalent figure for inquiry-based searching is 10.24s, although as noted previously, multiple handsets can be queried simultaneously.

3.4 Disconnection Time

Bluetooth connections can be either be shutdown in an orderly fashion by either end or severed when out of range. For the latter, a complete disconnection occurs after LSTO seconds have elapsed without communication. However, we have experimented with our test phones and found that when a connection was lost

[2] Note that BlueZ cached the clock offset so we expect all connections to occur in train A.

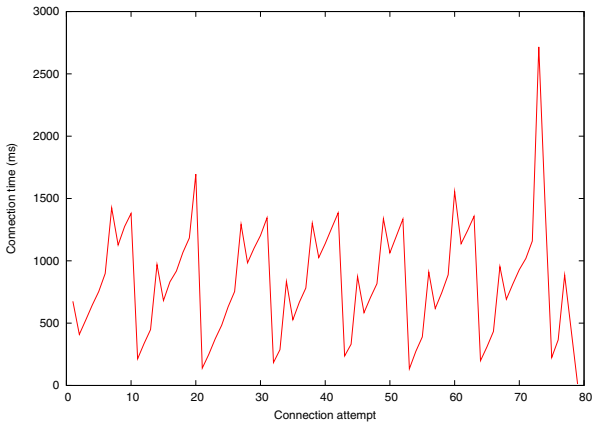

Fig. 2. Experimentally measured L2CAP connection times for the iPhone

due to range, the connection did *not* re-establish itself if the handset re-entered the radio range before the LSTO completed. This means that if a handset leaves communications range of a host and returns within the LSTO, we may not be able to 'see' it from the original host until the LSTO has expired. The LSTO value can be set dynamically and it is logical to set this to a small value since we expect frequent disconnection resulting from mobility.

3.5 Connection Monitoring

Once a handset and beacon are connected, a forced disconnection signals that they are no longer co-located. Whilst this is useful information, it may be possible to infer more by monitoring the connection quality.

In general there are three metrics used to monitor a Bluetooth connection: Received Signal Strength Indicator (RSSI); Link Quality (LQ); and echo response time. The standard does not require a Bluetooth chip to report the RSSI or LQ, although most do. We assume that the Bluetooth chip of the beacons, where we wish to measure the metrics, can be chosen to support these features.

The advantage of using the RSSI or LQ measurements are twofold. Firstly, they do not involve sending extra data wirelessly and so do not consume power at the mobile handset. Secondly, our experiences indicate that many manufacturers update the reported RSSI value at 1Hz or even faster, allowing for fast monitoring of a link if so desired.

An alternative method to monitor the connection is to use the round-trip time for an echo packet (c.f. ICMP ping). Table 2 shows the experimentally-measured maximum rate of echo requests for each of our test handsets when close to the sending host.

We have found the round-trip time to be of the order of 40ms for a strong connection, so update rates faster than 1Hz are easily achievable. However, we

Table 2. Experimentally measured ping rates for different handsets

Handset	Echo process rate (Hz)
T-Mobile G1	13.00
Apple iPhone 3G	20.14
Nokia 6300	21.97
Nokia N80	20.26

note that this approach does require the handset to be more active and therefore reduces its battery lifetime; we investigate this shortly.

RSSI/LQ And Proximity. Having established that the metrics can be read, there is value in asking how useful they are. There is a growing body of work on the use of RSSI values for location determination through fingerprinting techniques [19,20,22]. We note that ideas described thus far could be used to gather RSSI readings at a higher rate than the usual approach (using inquiries) and so may improve those methods. However, fingerprinting requires detailed surveying (and continual re-surveying) which does not meet our goals of minimal maintenance and we do not pursue fingerprinting here.

There have also been many attempts to relate the RSSI or LQ values to absolute distances with which to derive position estimates [21]. We feel that such approaches have demonstrated only limited success to date and opt for much coarser use of the RSSI.

We surveyed a series of offices using an iPhone to measure RSSI and a co-located Bat [3] to simultaneously measure locations to within a few centimetres. The traces from three hosts are shown in Figure 3. Figure 4 shows the observed relationship between the reported RSSI and the distance from the fixed Bluetooth master, based on data collected for 10 different masters. Note that an RSSI of -13 is actually a disconnection. Madhavapeddy and Tse performed further studies of Bluetooth propagation in our offices using a similar technique [23]. It is clear from Figure 4 that there is no deterministic relationship between distance and RSSI, but that there is a qualitative trend (larger distances are associated with more negative RSSI values). We simply use this trend to associate a falling RSSI reading with a handset leaving the area, which is generally sufficient for our purposes.

3.6 Battery Costs

Battery power is always a key issue for mobile devices. Providing any new functionality such as tracking ability is useless if users disable it to save power. It is, however, very difficult to accurately model the energy cost of the components of a generalised handset. Instead we created a custom rig for the T-Mobile G1 and Nokia N80 handsets that allowed us to accurately sample instantaneous power draw at 1kHz. The apparatus involved a replacement battery connected to the real one via circuitry to monitor the voltage and current, based on the design used by Hylick et al. [24] to analyse hard drive energy consumption. We used

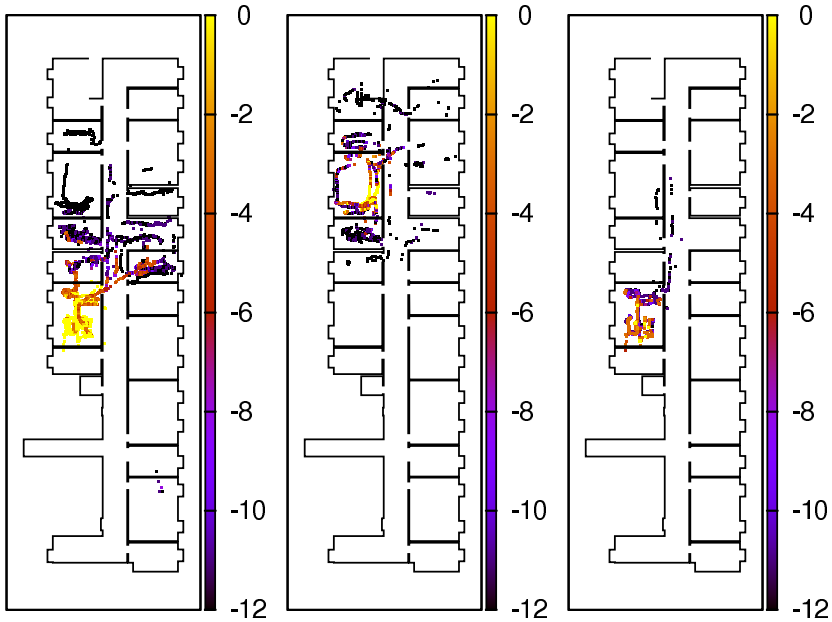

Fig. 3. Experimentally measured RSSI values from three hosts, plotted against locations recorded using the Bat system

this rig to investigate the power draw associated with monitoring a connection using either RSSI or echo response times.

The results are reported in Table 3, each given a unique test ID for reference here. Tests 1–3 provide baseline power draws; tests 4–6 provide power draws for continuous connection states; tests 7–15 provide power draws for discontinuous state (e.g. the connection was created, an RSSI taken and the connection shut down until the next update time). For all of these tests, we ensured that the handset screen was powered off, no SIM card was in place, no other connections were active and no applications (apart from any operating system requirements) were running on the devices. The power draws were computed by averaging over appropriate time periods (mostly 10 minutes).

The results are broadly as expected. A handset that is page scanning need only take action every T_{scan} seconds, and then only for 11.25ms. Thus we observe only a small associated cost. Saturating a permanent connection with echo requests (test 5) is very energy-intensive, whilst continually measuring its RSSI lies between the two extremes (and equates to the energy cost of maintaining an L2CAP connection).

When considering less frequent monitoring, instead of holding a connection open it may be more energy efficient to connect, take the reading, and then disconnect. Tests 7–15 concerned connection monitoring with such a model and we observe that, within error bounds, there is little to choose between using an

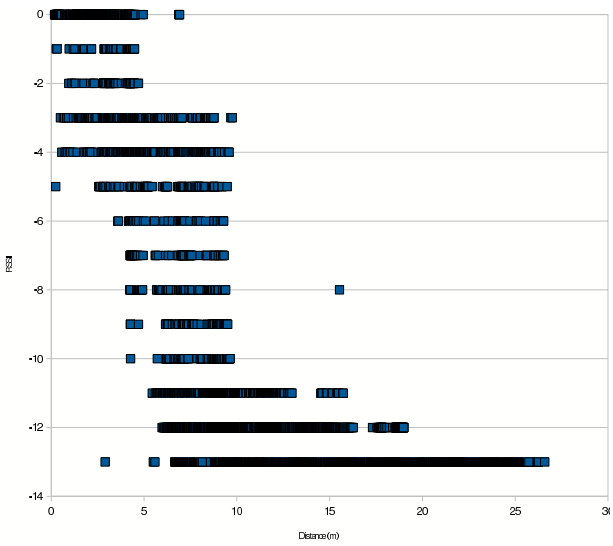

Fig. 4. Measured RSSI against euclidean distance from the host

echo packet or reading an RSSI. The main cost lies in creating and destroying a connection, which is relevant to both metrics. These data show that if the update period exceeds 5s, it is more efficient to break the connection and reform it when it is next needed than maintain the connection. For period less than

Table 3. Experimentally measured power draws

Test ID	Handset State	G1 (mW)	N80 (mW)
1	Idle	12.81	19.44
2	WiFi connected but idle	170.66	438.08
3	Bluetooth on, discoverable, unconnected	15.45	22.17
4	Bluetooth on, continually being scanned by host	16.07	31.80
5	Continuous Bluetooth echo at maximum rate	321.07	234.43
6	Continuous Bluetooth RSSI measurement	74.97	89.20
7	Bluetooth echo every 30s (with reconnect)	23.76	26.17
8	Bluetooth echo every 20s (with reconnect)	25.19	28.17
9	Bluetooth echo every 15s (with reconnect)	28.02	–
10	Bluetooth echo every 10s (with reconnect)	30.53	42.27
11	Bluetooth echo every 5s (with reconnect)	40.13	50.61
12	Bluetooth RSSI every 30s (with reconnect)	29.93	28.05
13	Bluetooth RSSI every every 20s (with reconnect)	35.86	29.93
14	Bluetooth RSSI every every 10s (with reconnect)	47.59	36.04
15	Bluetooth RSSI every every 5s (with reconnect)	75.72	51.88

5s it is unlikely to be possible to disconnect since the underlying ACL does not close with the L2CAP connection.

3.7 Connection Saturation

The Bluetooth specification states that a Bluetooth device can maintain up to seven simultaneous connections to other devices. To reduce power consumption and complexity, however, there are reports that some handset manufacturers have reduced this limit in their handsets. We attempted to verify this, but were unable to find a handset that did not support the full seven connections. We speculate that handset capabilities are now sufficiently advanced to permit full specification compliance without adverse impact.

4 A Tracking System

Having characterised the fundamental properties of Bluetooth in the context of connection-based tracking, we now turn to the issue of exploiting these properties to build a wide-area tracking system.

4.1 System Architecture

We assume a large building, within which a large number of inexpensive Class 2 Bluetooth radios are distributed. Each radio must be connected to a host machine of fixed, known location and the host machines must be networked together in some way. In most scenarios, we envisage simply adding a Bluetooth dongle to desktop PCs, making deployment both quick and easy. We refer to these fixed machines with Bluetooth simply as *hosts*.

A central system that can connect to any host maintains a database of handset device IDs, their owners and any known properties of the handset (T_{scan} etc). Each entry in the database is the result of a one-off registration of the handset with the system.

The central system then maintains a model of where people are and uses this to instruct hosts to monitor associated handsets at appropriate times. For example, as a user left their office the system may instruct hosts at either end of the corridor to monitor the handset in order to infer the direction in which he or she went.

4.2 Distributed Clock Offsets

This system involves continually connecting and disconnecting hosts and handsets and so it is extremely important to minimise the connection times. From the analysis of Section 3.3, we seek to ensure that the host always incorporates the handset's current listening frequency within the paging train A. If only one host was involved, this could be achieved by accepting a lengthy first connection and thereafter caching the clock offset associated with that handset.

However, the connection/disconnection here involves many hosts, typically in sequence. We propose that, once a host discovers the offset between its local

Bluetooth clock and that of a handset, it distributes the offset to other hosts. For this to work, each host must be able to estimate the offset between its local Bluetooth clock and that of the handset. The estimate must be sufficiently accuracy that the host predicts a listening frequency for the handset that is within 8 frequency hops of the true value. This ensures that the true frequency will be in train A.

The handset advances its listening frequency every 1.28s. Thus, hosts other than the original may incorporate an error of up to $8 \times 1.28 = 10.24s$ in their clock offset and still connect using train A. Therefore distributing clock offsets can be trivially achieved by using NTP to synchronise the system clocks of all hosts, and having them report two offsets: the clock offset between their system clock and their Bluetooth clock, and the offset between their Bluetooth clock and that of the handset.

We verified this approach in principle. We first time-synchronised two desktop machines using NTP. We connected both machines in turn to a particular handset and measured the offsets between the local Bluetooth clock and both the handset clock and the local system clock. We were able to predict the offsets of one machine given only the data from the other to an accuracy far greater than was required to connect in train A. In practice, however, we have found that the BlueZ code that permits specifying the clock offset to use for a particular connection is not yet functional. We believe, therefore, that it is justifiable to assume that all connections take place within T_{scan} seconds, which gives an expected connection time of 0.64 seconds in the default case.

4.3 Search

At its simplest level, this setup can be used to emulate an inquiry-based system for a single user. A subset of hosts that represents the minimum set needed to cover the entire building would simultaneously be instructed to connect to the associated handset. Any successful host(s) then report back, allowing the central system to localise the user. Those devices that are out-of-range and still attempting to connect after localisation has completed can simply cancel their paging, allowing the user to be localised quickly and the next update to begin.

Whilst this algorithm will localise the user significantly faster than an equivalent inquiry-based system, it does not scale well with the number of users. An inquiry-based system may only provide position updates every 10.24s, but it handles multiple handsets simultaneously. We would expect a connection-based system used in the manner described to be able to localise around ten handsets in the time that an inquiry-based handset would localise almost all handsets.

We can start to address this shortcoming in two ways. The first is to use the connection monitoring techniques of Section 3.5 to identify those users are moving away from their nearest host (the current 'home' host) and who therefore need to be tracked. From our previous analysis of people's working habits we know that the vast majority of workers will be sedentary at any given time [5] and therefore the number of mobile users should be small enough to localise all users within reasonable time-frames.

The second is to limit the extent of search where possible. If, for example, a user's current home reported a very weak RSSI (indicating lost proximity) t seconds ago, the only hosts that can reasonably observe him now are those within vt of the reporting host, where v is the speed of the user. This type of information permits simultaneous searching for different handsets known to be in different building areas. This, combined with the fact there may be disjoint subsets of hosts that completely cover any given region, should permit tracking of many users simultaneously with update rates similar to that of inquiry-based tracking. However, the system will struggle to cope with many users moving at once or collected within a small area with few covering hosts. In such a scenario, the update rate for tracking will degrade, but the system will self-heal when the crowd disperses.

4.4 Bootstrapping

The discussion so far has ignored how a registered handset returning to the building will get discovered before being tracked. We currently have two options:

Constant Round-Robin Polling. The obvious solution is to continually cycle through all the registered handsets that are not being tracked and poll them from a variety of hosts. Because a connection can take up to 5.12s to connect (worst case), cycling through 100 registered handsets can take over eight minutes to complete. Additionally, it places an extra load on hosts that may be in use for monitoring tracked handsets, reducing update rate.

Out-of-Band Events. Our preferred solution is to use out-of-band events to signal that a particular handset is likely to be in the building. These events can be easily generated from computer system events (e.g. the associated user logs in) and from building security (e.g. the user gains access to the building by swiping their ID card). It may also be possible to infer presence based on cell-ID from the mobile telephony network.

Additionally, we note that many people follow a daily routine which there is potential to learn autonomously. Doing so would allow prediction of when a user is likely to arrive, permitting for targeted polling of the subset of expected handsets rather than all handsets.

4.5 Inverted System

There is, of course, the possibility to invert the system and have the handset as the master, connecting to the hosts as slaves. This would require custom software on the handset and a wireless data channel to receive a model of the host positions and addresses. This has the advantage that no bootstrapping would be necessary since the handset localises itself. However, care would be needed to ensure that a given host was not saturated with connection requests from a series of nearby handsets. Hence a degree of centralised control is still likely to be needed.

We note that such inversion is common for the inquiry-based approach because it permits the handset to localise itself and still remain anonymous. Unfortunately, in the inverted connection-based model, whenever a handset connects to a host it must explicitly identify itself and anonymity is broken. Given also that paging is more power-hungry than page scanning, the inverted technique is not compelling for connection-based systems.

5 Discussion

We believe we have demonstrated that connection-based tracking is a viable alternative to inquiry-based tracking, although neither is ideal. We characterise the advantages of connection-based tracking as:

Faster Update Rate. Connections can generally be *established* faster than a scan can be completed. Connections can be *monitored* at a very fast rate.
Variable Update Rate. The rate can be adapted to suit the context of the mobile device and preserve battery life. If, for example, we believe the user to be sedentary, we can choose to reduce the query rate.
No Scanning. The lack of need for constant inquiries is a significant enhancement to Bluetooth usage, security and privacy.
Privacy. Users retain their right to opt out of being tracked by switching off Bluetooth on their handset.

There are, however, a number of drawbacks:

Devices must be Known. Connection establishment requires that at least one party knows of the existence of the other in order that it can issue a connection request; however, this convenience drawback is also a privacy advantage.
Security Policies are not Universal. Not all Bluetooth security policies permit L2CAP connections without authorisation. Similarly, not all Bluetooth chips permit querying the connection metrics such as RSSI. However, such queries are performed on the beacon's Bluetooth chip, which we are free to choose.
Connection Limits. Care must be taken to ensure Bluetooth limits on the number of connections are not reached.

Table 4 compares connection and inquiry based tracking side by side. Ultimately, Bluetooth was designed as a short-range wireless communications protocol and as such it is unreasonable to expect it to provide ideal location tracking. Nonetheless, both inquiry-based and connection-based tracking are at least feasible, and both can be implemented without modifying handset software or hardware, which is an important consideration.

In terms of security and privacy, we note that any handset with Bluetooth turned on is likely to be trackable by the techniques in this paper if an observer knows the Bluetooth address of it (this requires the handset to have trusted the

Table 4. Comparison of inquiry-based and connection-based tracking

	Inquiry-based Handset-scans-Host	Inquiry-based Host-scans-Handset	Connection-based
Works on Unmodified Handsets	Most	Some	Almost all
Requires Custom Handset Software	Yes	No	Yes
Deployability	Med	High	High
Requires Discoverable Handset	No	Yes	No
Requires Separate Data Channel	Yes	No	No
Location Update Frequency	∼0.1Hz	∼0.1Hz	Varies (0-20Hz)
Supports Dynamic Update Rates	<0.1Hz	<0.1Hz	Yes
Scalability	Simple	Simple	Complex
User Privacy	High	Low	Med–High
Handset Power Drain	High	Low	Varies (Med)

third party at some point in the past). This means a third party can potentially track specific handsets without their knowledge.

Looking forward, we believe that connection-based tracking will improve in the short-term as handsets advance. Already we see handsets that incorporate accelerometers (which could be used to infer movement or the lack of it, or to dynamically update the monitoring rate). Similarly, advanced development platforms for mobile devices are emerging and these will hopefully provide low-level access to handset subsystems such as Bluetooth. This in turn will permit handsets to be more active in the location process since APIs and behaviours will standardise and applications will be easier to deploy.

6 Conclusions and Further Work

In this paper we have identified and evaluated an alternative to inquiry-based Bluetooth tracking in the form of connection-based tracking. This form of tracking does not demand that devices be permanently discoverable (something which is no longer possible on many mobile devices), nor that the mobile handsets are modified in any way.

We have studied the properties of Bluetooth that enable connection-based tracking both theoretically and experimentally and concluded that there is potential for the approach. There is little doubt that inquiry-based and connection-based tracking have their advantages and disadvantages (many of which do not overlap). We recognise that neither makes the ideal tracking system, but both have very little in the way of deployment costs, allowing the construction of large testbeds. We intend to develop the tracking algorithms and evaluate them by

deploying a connection-based tracking system across our building and encouraging many users of it.

References

1. Hightower, J., Borriello, G.: Location systems for ubiquitous computing. Computer (January 2001)
2. Addlesee, M., Curwen, R., Hodges, S., Newman, J.F., Steggles, P., Ward, A., Hopper, A.: Implementing a sentient computing system. IEEE Computer 34(8), 50–56 (2001)
3. Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P.: The anatomy of a context-aware application. In: MobiCom 1999: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pp. 59–68 (1999)
4. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The Active Badge location system. ACM Transactions on Information Systems 10(1), 91–102 (1992)
5. Harle, R., Hopper, A.: The potential for location-aware power management. In: UbiComp 2008: Proceedings of the 10th International Conference on Ubiquitous Computing (September 2008)
6. Rehman, W., Lara, E., Saroiu, S.: Cilos: a CDMA indoor localization system. In: UbiComp 2008: Proceedings of the 10th International Conference on Ubiquitous Computing (September 2008)
7. Bahl, P., Padmanabhan, V.: RADAR: an in-building RF-based user location and tracking system. In: INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, vol. 2, pp. 775–784. IEEE, Los Alamitos (2000)
8. Youssef, M., Agrawala, A.: The Horus WLAN location determination system. In: MobiSys 2005: Proceedings of the 3rd international conference on Mobile systems, applications, and services (June 2005)
9. Anastasi, G., Bandelloni, R., Conti, M., Delmastro, F., Gregori, E., Mainetto, G.: Experimenting an indoor Bluetooth-based positioning service. In: Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops, April 2003, pp. 480–483 (2003)
10. Cheung, K., Intille, S., Larson, K.: An inexpensive Bluetooth-based indoor positioning hack. In: UbiComp 2006: Proceedings of the 8th International Conference on Ubiquitous Computing Extended Abstracts (2006)
11. Bargh, M., Groote, R.: Indoor localization based on response rate of bluetooth inquiries. In: MELT 2008: Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments (September 2008)
12. Jevring, M., de Groote, R., Hesselman, C.: Dynamic optimization of Bluetooth networks for indoor localization. In: AASN 2008: First International Workshop on Automated and Autonomous Sensor Networks (2008)
13. Huang, A.: The use of Bluetooth in Linux and location aware computing. Master of Science dissertation
14. Bruno, R., Delmastro, F.: Design and analysis of a Bluetooth-based indoor localization system. PersonalWireless Communications, 711–725 (2003)
15. Hallberg, J., Nilsson, M., Synnes, K.: Positioning with Bluetooth. In: ICT 2003: Proceedings of the 10th International Conference on Telecommunications, vol. 2(23), pp. 954–958 (2003)

16. Pandya, D., Jain, R., Lupu, E.: Indoor location estimation using multiple wireless technologies. In: PIMRC 2003: 14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications, August 2003, vol. 3, pp. 2208–2212 (2003)
17. Naya, F., Noma, H., Ohmura, R., Kogure, K.: Bluetooth-based indoor proximity sensing for nursing context awareness. In: Proceedings of the 9th IEEE International Symposium on Wearable Computers, September 2005, pp. 212–213 (2005)
18. Jakobsson, M., Wetzel, S.: Security weaknesses in Bluetooth. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 176–191. Springer, Heidelberg (2001)
19. LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert, J., Powledge, P., Borriello, G., Schilit, B.N.: Place lab: Device positioning using radio beacons in the wild. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) PERVASIVE 2005. LNCS, vol. 3468, pp. 116–133. Springer, Heidelberg (2005)
20. Graumann, D., Lara, W., Hightower, J., Borriello, G.: Real-world implementation of the location stack: the universal location framework. In: WMCSA 2004: Proceedings of the 5th IEEE Workshop on Mobile Computing Systems and Applications, pp. 122–128 (2003)
21. Gwon, Y., Jain, R., Kawahara, T.: Robust indoor location estimation of stationary and mobile users. In: INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, vol. 2, pp. 1032–1043 (2004)
22. Schilit, B.N., LaMarca, A., Borriello, G., Griswold, W., McDonald, D., Lazowska, E., Balachandran, A., Hong, J., Iverson, V.: Challenge: Ubiquitous location-aware computing and the place lab initiative. In: WMASH 2003: Proceedings of the First ACM International Workshop on Wireless Mobile Applications and Services on WLAN (2003)
23. Madhavapeddy, A., Tse, A.: A study of Bluetooth propagation using accurate indoor location mapping. In: Beigl, M., Intille, S.S., Rekimoto, J., Tokuda, H. (eds.) UbiComp 2005. LNCS, vol. 3660, pp. 105–122. Springer, Heidelberg (2005)
24. Hylick, A., Sohan, R., Rice, A., Jones, B.: An analysis of hard drive energy consumption. In: MASCOTS 2008: Proceedings of the 16th Annual IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (2008)