

# BT-Crowds: Crowds-style anonymity with Bluetooth and Java

Antti Vähä-Sipilä  
Nokia

Email: avs@iki.fi, antti.vaha-sipila@nokia.com

Teemupekka Virtanen

Helsinki University of Technology

Telecommunication software and Multimedia Laboratory

Email: teemupekka.virtanen@hut.fi

**Abstract**—This paper aims to describe a new idea on how privacy in a Crowds-style network [1], [2] could be enhanced by running it over Bluetooth personal area networking. A potential implementation is described using the Java Bluetooth programming interface. The proposal is called BT-Crowds and it describes a store-and-forward Crowds and remailer-style [3] service which uses the Bluetooth protocol stack and provides a system for privacy-enhanced communications that has some significant benefits over Crowds in fixed networks. It leverages the property of a single Bluetooth user being indistinguishable from a group of many, and the property of ad hoc networks being distributed in physical space.

**KEYWORDS:** PAN, Bluetooth, Anonymity, Privacy

## I. PERSONAL AREA NETWORKING

Networking (interconnection of computers) has been traditionally divided into various categories depending on how large the covered geographic area is. Established categories range from wide area networks (WANs) that cover an area that is typically larger than a city to MANs, metropolitan area networks, and further to LANs, local area networks. LANs are the most well-known of these network categories, and are typically implemented by dedicated cabling such as Ethernet, in contrast with larger networks that may be implemented by virtual private network tunnels.

Even though wired LANs are the norm at present, wireless LANs (WLANs or Wi-Fi networks) are growing in popularity. The leading WLAN technology is called IEEE 802.11. This standard is a member of the 802 family of standards, which also incorporates the standards which define Ethernet (IEEE 802.3) and Token Ring (IEEE 802.5).

A relatively new category in networking is personal area networking or PAN. Personal area networks usually span an area which is a few metres around a person, typically a sphere of a radius of 10 metres. This is called the sphere of proximity. PANs are used to interconnect the user's own electronics such as a portable computer, mobile phone, or some other gear such as a wristwatch or a headset. PANs can also connect to other people's PAN-enabled electronics within the sphere of proximity. Sometimes PANs are further divided to wireless (WPANs) and some sort of 'wired' personal networks; in the latter type, the wiring may actually involve using the human body as a conduit for data. In this paper, all PANs are WPANs.

Personal area networks differ from others in that connections through PANs tend to be transient, and must be created quickly and without any previous knowledge of other

network nodes. Sending of business card data is a good example of a transaction that is made possible by a PAN. PANs can also be used in ticketing (for example, instead of contactless smart cards that are currently deployed, although PAN connection speeds currently do not facilitate their use for public transport) and information kiosks. Because of this transient nature, PANs are often ad hoc networks, meaning that a node forms a network with other nodes on demand. Personal area networking also facilitates the distributed design of personal mobile devices. A monolithic device, where the display, keyboard or other input device and any output devices are combined, has its limits in usability. Consider, for example, a camera phone where the camera and the display would be embedded in a wristwatch, earphones and the loudspeaker would be embedded in a cordless headset, and the tranceiver part would be in the suitcase. This would provide freedom of movement which is not possible when holding the phone or when tethered by a cord.

## II. BLUETOOTH

Bluetooth is a wireless networking specification that operates on a licence-free area of radio spectrum located between 2.4000 and 2.4835 GHz. This band is called the industrial, scientific and medical (ISM) band and is shared by various applications such as radio amateur satellites, microwave ovens (around 2.45 GHz) and indeed many other wireless networking systems like IEEE 802.11 WLANs (Wi-Fi networks). The band is available in most countries throughout the world, except for some limitations in certain countries. [4]

Bluetooth is not a standard in the strict meaning of the word, but it is a product of an industry consortium Bluetooth SIG. Bluetooth is relatively new: systems that incorporate it are becoming more and more commonplace but still tend to be at the higher (more pricey) end of personal electronics. Bluetooth is better suited for personal area networking than IEEE 802.11 because it consumes less power and has been designed from ground up for ad hoc (on demand) networking. Most likely 802.11-based WLANs and Bluetooth PANs will coexist, IEEE 802.11 providing connectivity for computers and workstations in office environments and Bluetooth for mobile devices. Many devices will most likely contain both systems, modern laptops showing a good example.

Bluetooth protocol stack contains some layers that have been explicitly designed for Bluetooth and others that have

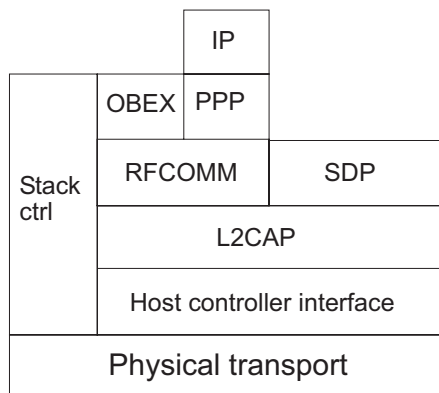


Fig. 1. The Bluetooth protocol stack

been adapted from other networking systems. One of the major ones to affect Bluetooth was IrDA, the infrared connectivity system that has been in use in mobile phones for several years. It is useful to describe the Bluetooth system by its protocol stack (figure 1).

A. The Bluetooth baseband

The lowest layers of the Bluetooth protocol stack are called the baseband layers. Baseband contains the physical radio layer and the layers for link control. Baseband layers have an abstraction layer called the host controller interface (HCI). It is possible to split the Bluetooth protocol stack in such a fashion that the layers below the host controller interface are handled by one module, which can be manufactured separately [5, p.407].

The link layers give the Bluetooth system some of its interesting capabilities for ad hoc networking. Bluetooth devices are capable of forming piconets, where there can be seven active devices at a time and 128 more in a passive mode. One piconet is formed entirely within the sphere of proximity of a master device (decided upon piconet formation). However, one device can take simultaneously part in several piconets that have partially overlapping spheres of proximity. This is called a scatternet. Link layers also handle different power saving operational states of Bluetooth devices. [5, pp.41-90]

On the link layer level, devices are distinguished by their Bluetooth device address. This address is a 48-bit IEEE EUI (Extended Unique Identifier) address (called the EUI-48), much like the one which is used on IEEE 802.3 Ethernets. Each manufacturer has its own numbering block (Organisationally Unique Identifier, OUI), and all Bluetooth devices are assigned a globally unique address within these numbering blocks. [5, p.42], [6]

It should be noted at this point that static Bluetooth device addresses may be a privacy risk. Consider a wireless kiosk which offers services over Bluetooth and is constantly scanning its sphere of proximity for Bluetooth devices. It can easily log all Bluetooth device addresses it sees, and if upper-level protocols identify the user of the device at any time, correlating

this user data with the Bluetooth device address makes it possible to track the user's geographical location. In addition, traffic analysis is helped by the fact that the communicating parties' device addresses stay the same. The Bluetooth SIG addressed this problem by promising to define an 'anonymity mode', the use of randomised addresses [7], but this feature failed to materialise in Bluetooth 1.2 Core Specification [8].

B. Host protocols layer

On top of the host controller interface, there is the logical link control and adaptation protocol (L2CAP). L2CAP provides data transfer services for upper layer protocols, most notably a connection-oriented service [9]. The two other host protocols are the RFCOMM layer [10, pp.393-424] and the service discovery protocol (SDP) [10, pp.331-392].

RFCOMM provides serial data transfer over the lower layers of the Bluetooth stack. To applications that are using it, RFCOMM services are very much like the services offered by a conventional serial port. Therefore, it is possible to run, for example, point-to-point protocol (PPP) over RFCOMM and the TCP/IP protocol stack on top of PPP. RFCOMM is based on an ETSI standard TS 07.10, and also exists for infrared as IrCOMM. There can be several concurrent sessions (like several serial ports) over RFCOMM.

The service discovery protocol (SDP) is used to find out whether the other devices in the sphere of proximity will be able to offer a certain service. Because Bluetooth can be used for a large variety of purposes, it is important for the system to know which services are available. Services are described by a service record which contains service attributes, which are key-value pairs. Each device that is able to provide a service has an SDP server, which can be queried by SDP clients on other devices. The client compares the advertised services to a search mask, and decides whether the services provided are useful based on its knowledge of advertised service attributes or the class of service. It is also possible to browse the services provided by the other devices. [11, pp.63-98], [10, pp.331-392], [9]

C. Application layers

The application layers in the Bluetooth stack consist of the protocols that are run over the host protocol layers. In many cases, this means the serial port emulation through RFCOMM. From our perspective, the most important service is however the object exchange protocol, OBEX, which is also run over RFCOMM. Like RFCOMM, OBEX has been inherited from the Infrared Data Association and is often called IrOBEX [12].

OBEX is of special importance in this chapter, because the BT-Crowds proposal is based on OBEX. OBEX provides a service which enables a quick exchange of data objects. As an example, the 'beaming' of business cards over infrared mentioned previously is done over OBEX: a business card object (a vCard object) is transferred to another device over the IrCOMM serial connection. A further example can be found in Microsoft Windows: an infrared-equipped Windows machine has a 'send to infrared recipient' option for files,

which transfers the selected files over OBEX. OBEX does not require any a priori knowledge of the other device. It has been designed for one-off transmissions of objects. OBEX is a relatively simple protocol. When an object is sent over OBEX, it also sends a fairly large amount of metadata on the object such as object name, a timestamp, a human-readable description and the object type. The object type is a MIME type: for example, a HTML text file would be text/html. OBEX also supports sending of HTTP headers with the object. The object usually ends up in the 'inbox' of the receiving device, but if the OBEX implementation is good enough, the object can be sent directly to the service that was discovered using Bluetooth service discovery protocol. Obviously, security risks have to be taken into account. [12]

#### D. Bluetooth profiles

Not all Bluetooth devices implement the same set of functionality. The way Bluetooth devices are implementing a certain usage scenario is called a Bluetooth profile. Some devices may support only some of the Bluetooth profiles, for example, a data access point (like a Bluetooth kiosk offering access to the Internet) rarely has a need for supporting telephony over Bluetooth. On the other hand, profiles are a way to guarantee that a certain level of interoperability has been reached. If a device implements a given usage scenario, it should do so according to a published Bluetooth profile. Profiles form a hierarchical structure: profiles inherit properties from other, more generic, profiles. Therefore, in order to create an interoperable Bluetooth device, the manufacturer needs to select which parts of which profiles to support, and the parts that are selected may require other parts of other profiles. This is the way in which Bluetooth devices are guided towards interoperability. [13, pp.209-222] For the purposes of this text, the following profiles should be mentioned:

1) *Service Discovery Application Profile (SDAP)*: SDAP uses the Service Discovery Protocol (SDP) to find services on other Bluetooth devices. Service discovery is very closely knit with device discovery, which is the time when devices are joining a Bluetooth piconet. This is because SDP needs a connection-oriented data channel between devices, and therefore two devices need to have a fully valid Bluetooth connection between each other before service discovery can be done. SDAP is primarily concerned with user-initiated service discovery, but applications without a user interface may use the same methods for finding services in an interoperable way. [13, pp.217-222], [9], [11, pp.63-98]

2) *Generic Object Exchange Profile (GOEP)*: GOEP defines the requirements for object exchange between Bluetooth devices. It defines an interaction between two devices, where one of the devices is a server containing files and storage space (actually, very much like a HTTP server) and the one is a client that can issue file requests and submissions. GOEP mainly defines how the OBEX protocol is used in an interoperable fashion. The BT-Crowds proposal is based on the GOEP. Therefore, in order for BT-Crowds to work, the device has to support the related parts of GOEP, and its parent profiles

(Serial Port Profile, which describes how serial data transfer is done over RFCOMM, and Generic Access Profile, which is the topmost profile in the profile hierarchy and mandatory for all Bluetooth devices). [13, pp.243-245], [11, pp.309-338]

3) *Object Push Profile (OPP)*: OPP builds on GOEP and it defines a simple, one-way object push, which very closely resembles the HTTP PUT method. [13, pp.245-250], [11, pp.339-364]

### III. UTILISING PAN PROPERTIES FOR PRIVACY

Crowds, in its fixed-network form, is vulnerable to an all-seeing attacker that can monitor all data exchanges. Wireless, short-range ad hoc communications has several benefits that make it harder to create an all-seeing attacker. Large numbers of devices, the transient nature of communications, ad hoc networking between previously known devices and the shared and interference-prone frequency band all contribute to the privacy of communications in a personal area network.

Because of the frequency band used by Bluetooth and the very small transmission power, it may be not straightforward to monitor devices from the distance. Recent developments have, however, cast doubt to the fact that this could be alone thought as a privacy measure. A device that was dubbed a 'BlueSniper', essentially a impressive-looking directional antenna, has been used to communicate with Bluetooth devices at the distance of 1.7 kilometres in open space [14], [15]. Noisiness of the ISM band may contribute to the security, as wireless LANs and Bluetooth transmitters become more commonplace. However, the most important part of additional security is the physically distributed and moving nature of communications. Building a centralised surveillance system cannot be done any more by installing the legal interception gateway box at operator premises. To enable blanket surveillance, there has been discussion about the feasibility of satellite monitoring of the ISM band, which might in the end make it possible to eavesdrop on a single device from the mass of devices on the ground.

Because PAN equipped devices are often personal and mobile devices, the users are usually carrying them and aware of the physical surroundings. As the sphere of proximity is fairly small, in most cases the users are aware of any other persons in their vicinity and possibly also about any other (at least visibly installed) electronics. Because of this, if the users want to make a transaction that will not be tracked, they can move to a location where there are no potential eavesdroppers, or better yet, move constantly. Also, when communicating to another person, the authentication of the other person can be readily done by conventional human means (looking at the face, talking to, and so on) - there is less need for an all-encompassing PKI, for example. Exchange of a shared secret could be done also visually (taking a picture of the other device's screen with an embedded camera), if the communicating applications would like to use this kind of system.

PAN-enabled devices will become more and more common. In some environments, one could find literally dozens of

devices that could connect to your piconet. If the devices do a series of transactions, it will be very hard to show later on which devices from that crowd in that physical location have participated in those transactions unless all transactions were logged.

Portable devices will start to have a large number of different data transfer methods. Usually, a computer connected to the Internet only has the TCP/IP stack and it must do all communications over it. As an example, the Nokia 6600 phone has a TCP/IP stack that can be used over GPRS or PPP over a data call; it has Bluetooth and infrared; it has GSM text messaging and multimedia messaging; it has a camera for one-way visual communications; and, of course, it has normal voice calls. Combining the different bearers for data transfer may offer extra protection against traffic analysis.

If the system is properly implemented, the mobility of the devices also offers resistance against denial of service. If an attacker of interference is active at one physical location, simply moving to another location should cure the problem in most cases. This option is usually not available in fixed networks, because it would need the machine to be moved to another place network-topologically, essentially requiring a multihomed machine.

There does not need to be a central registry of any sort. Bluetooth has a system for 'pairing' devices, where an authentication key is generated from a PIN code which is entered on both devices (for devices without a keyboard, a fixed PIN exists). This pairing system can be used to disallow, for example, rogue headsets from entering the user's piconet, but it does not need to be used in ad hoc-style transactions.

PAN does not require third party services such as mail boxes residing on a remote machine, a domain name system, backbone networks, or similar services. Therefore any attacks that are targeted towards such infrastructure (even if they are legislative attacks<sup>1</sup>) are not effective against PANs, as long as the transactions are kept local. Obviously, if a message needs to cross the boundary from the PAN networks to the big Internet, the laws of the Internet start to apply at the gateway.

#### IV. BT-CROWDS: PRIVACY THROUGH BLUETOOTH OBEX

##### A. Relationship to existing systems

Crowds [2], [1] is a system where privacy is enhanced by the principle of indistinguishability from a group. A message is passed from a node to node (nodes are called 'jondos', from 'John Doe') until one of the nodes decides to forward it to the final destination (see figure 2). As mentioned earlier, Crowds has the problem of being vulnerable to an all-seeing attacker, because the plain system does not use onion routing<sup>2</sup>

<sup>1</sup>For example, data retention by Internet service providers is already mandatory in some jurisdictions. Being operated by private individuals, Bluetooth networks are immune to this kind of data retention, at least until legislators notice it.

<sup>2</sup>Onion routing is a type of source routing where destination address of each hop is encrypted with the previous node's public key. Previous nodes cannot see where the message is going (except for the next hop) and as the layers are peeled off like onions, nodes further down the chain can not see where the message is originated.

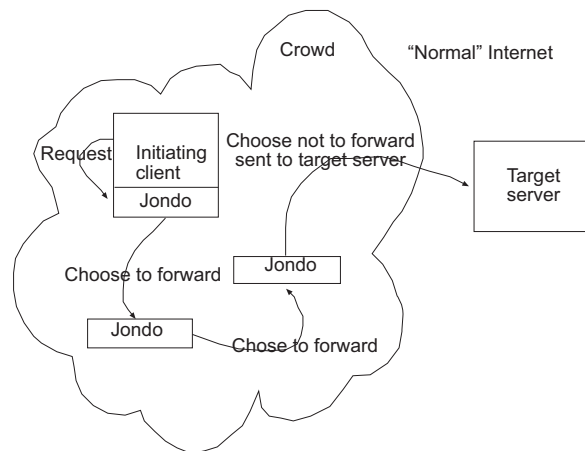


Fig. 2. Crowds operation

style systems or dummy traffic. Traffic analysis is therefore very easy. PANs offer a partial solution for this problem by distributing the message exchanges in physical space.

Crowds is adapted to web browsing with HTTP. However, just because PAN connections are transient and cannot last for long enough, there cannot be a return channel in a Crowds system based on PANs. Therefore the service offered by this proposal is closer to that offered by a remailer network. It cannot be a remailer network, however; the same transient nature of PANs makes it impossible to do source routing of messages through the privacy-preserving chain. In this sense, BT-Crowds is a privacy-enhancing object forwarding system which inherits some features from Crowds and some features from remailer networks.

##### B. BT-Crowds organisation

The internal organisation of BT-Crowds jondos is shown in figure 3. Each jondo contains the Bluetooth protocol stack, the Bluetooth API, and the BT-Crowds application.

A BT-Crowds jondo is very much like a HTTP server, as OBEX requests lend heavily from HTTP. Jondos accept OBEX PUT requests (not unlike the HTTP PUT request, which is usually used to transfer form data to HTTP servers), and accept a certain MIME object type as the payload. Most likely the object type message/rfc822, meaning a standard Internet mail message, is the best. This object type is a valid RFC 822 (nowadays RFC 2822) message, which can be sent over SMTP as-is. RFC 2822 messages may contain a variety of object types in their body. Multimedia messages can be expressed as message/multipart objects. Security can be employed by using S/MIME or PGP/MIME. Addressing can use any valid RFC 2822 address, which also enables addressing to recipients with only a phone number. RFC 2822 parsers can be found on many devices already, which should cut down the development cost.

Another side of a BT-Crowds application is like a HTTP client (called "scheduler" in figure 3), which sends out OBEX PUT requests to other jondos. Between the server and client

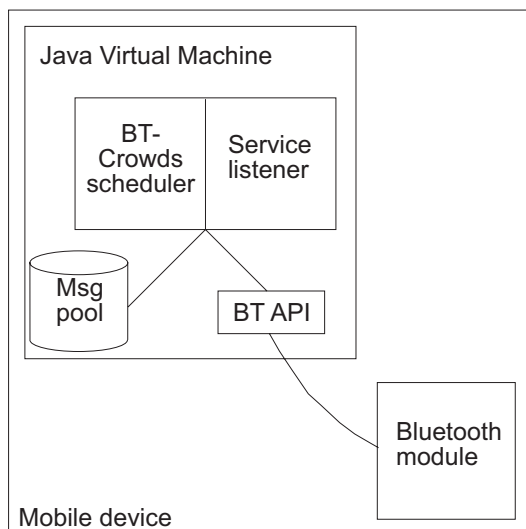


Fig. 3. Internal organisation of a BT-Crowds Jondo

parts there is a shared message pool, which holds the messages that are in transit. This message pool should be persistent, that is, it should survive the termination of the BT-Crowds application and device power-down. Messages are sent out from the pool in random order and only after a random time.

On a higher level, BT-Crowds is formed as seen in figure 4. In this model, there are three types of jondos: normal jondos, gateway jondos and kiosks. Normal jondos function like a Crowds jondo, passing messages along to other jondos. Gateway jondos are devices which are able to route the message to the final recipient, outside BT-Crowds. A suitably equipped phone may act as a gateway jondo. Kiosks are specially configured jondos that are placed in (physically) crowded places. These function as drop-boxes, where passing jondos can leave messages and others can pick them up. Typically a kiosk could be installed in a place where a lot of people pass past it, like in a corridor. In cities that have a sensible urban planning (where people still do walk in the city centre), someone could set up a kiosk in an office on the first or second floors. There it would communicate with jondos that people carry with them in the street.

It should be noted that BT-Crowds design is send-only. Because of the network properties, there is no easy way of routing replies back to the sender. However, potential systems for send-and-recv systems are discussed later.

### C. Requirements of a BT-Crowds jondo

The Bluetooth specification is luckily open in the sense that it does not restrict the personal area networking to any specific protocol. As TCP/IP can be sent over the Bluetooth stack, either over PPP and RFCOMM or using the Bluetooth PAN profile, any IP-based protocol can be used. In addition, 'native' Bluetooth protocols, most notably OBEX, are independent of any media types that are transferred over them. This makes it

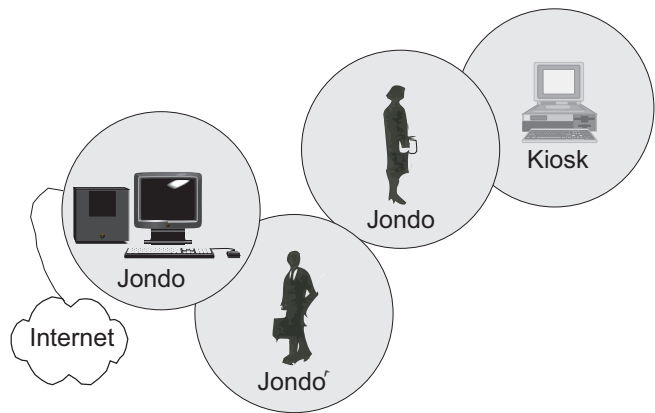


Fig. 4. BT-Crowds high level organisation

fairly easy to use the Bluetooth stack for realisation of privacy-enhancing services. The only problem is the openness of the platform and the application programming interfaces. Unless third party programmers get free access to Bluetooth APIs, and particularly APIs such as OBEX, it may not be possible to effectively implement privacy enhancing systems. This poses a major challenge to manufacturers of devices to open up enough APIs so that Bluetooth can be used to its fullest potential. Not only is this important for privacy applications, but also to many potential killer applications which combine the user's social life and the sphere of proximity. One such application is the automated matchmaker system, which scans potential matches for the user's personality and interest profile while the user is walking down the street. There will be undoubtedly a myriad of similar applications, and a third party developer scene is the only way to realise them all.

Each jondo in BT-Crowds is a Bluetooth-enabled device which supports the Generic Object Exchange Profile. The device is able to send and receive generic objects via OBEX and RFCOMM. In order to find devices that are supporting BT-Crowds, the systems also have to support Service Discovery Application Profile. In theory, BT-Crowds ideas can also be operated over infrared, because all the necessary systems (OBEX and serial port emulation) have been specified by IrDA. Interestingly, the Java Bluetooth API (JSR-82) [16] has the same requirements as BT-Crowds. This is fortunate as this should enable the creation of a single BT-Crowds client that can run on any platform that supports the full Java JSR-82. It is possible to distribute a BT-Crowds client via peer-to-peer sharing. New BT-Crowds installations can then immediately enter the BT-Crowd and start sending messages.

The Bluetooth programming interface has to fulfill the following requirements:

- It must be possible to build a BT-Crowds application which acts both as a server and client, and that the server and client parts share a persistent message pool between them. Problems may occur if waiting for an OBEX request is a so-called blocking request, which puts

the running process on hold until a request happens. In this case, the BT-Crowds jondo should spawn a separate process (or thread) to handle outgoing messages from the pool.

- It must be possible to create a server that can receive and process MIME objects over OBEX and RFCOMM (this implies support for GOEP, SPP and L2CAP), although only OBEX-level interface is required for applications.
- It must be possible to create a client that can send MIME objects, again over OBEX and RFCOMM.
- There must be an API that makes it possible to find and connect to devices that are identified as running BT-Crowds. This means that the BT-Crowds application must be able to insert a service record in the device's Service Discovery Database (SDDDB).
- As it is important for the usability of the system, the user should be able to grant the BT-Crowds application the possibility of transferring objects over OBEX without requiring user acceptance or intervention. Otherwise the user would need to acknowledge each transfer. This is most likely a device manufacturer policy decision<sup>3</sup>. The application may be able to set this policy on a per-service basis. Some device manufacturers may require that in this case, the origin of the application is authenticated (for example, in Java using a digitally signed JAR (Java Archive) file).
- Again, as an usability factor, it must be possible to send and receive MIME objects without authenticating<sup>4</sup> or pairing the devices with each other.
- The device must be able to continuously run BT-Crowds (client) in the background. A device which only can run one application at a time is not sufficient as BT-Crowds has to be able to relay message on an opportunistic basis. Luckily there are many other use cases where this kind of functionality is required, so it is foreseeable that devices will support this.

In addition to Bluetooth, any BT-Crowds jondos that want to operate as gateway jondos need to have some wider-area networking system installed. Gateway jondos are the jondos that can actually send a message to the recipient. Various systems are available for this purpose. OBEX can be used over TCP/IP, so BT-Crowds jondos could actually operate completely without Bluetooth. However, it is probable that the final recipient does not support OBEX as it is not very widely used outside small devices. Short, textual messages could be sent via GSM Short Message Service (SMS). Multimedia Message Service (MMS) might be an option for smallish pictures, video and audio, although some operators reportedly cap MMS sizes at 300 kilobytes at the moment. Longer

<sup>3</sup>In the light of the recent Bluetooth security scares, this may be challenging. However, device manufacturers should understand the vast business cases in exchanging information without explicit user acceptance. Obviously the implementation must be robust to mitigate attacks.

<sup>4</sup>Due to the fact that Bluetooth baseband encryption is coupled with authentication, this also means that BT-Crowds applications do not use encryption while communicating. If encryption is desired, it must be applied at BT-Crowds message level.

messages can be routed to their destination using SMTP over TCP/IP. The TCP/IP connection might be opened over packet data, such as GPRS, or in the case of a WLAN-equipped device, the gateway jondo could send the accrued mails when it joins a WLAN with an outbound connection. There could also be dedicated gateway jondos which are connected to a fixed network. Gateway jondos are in a special situation, because sending messages to the Internet usually costs money. Some methods, like connecting to a WLAN and sending the data, may have a negligible cost. SMS and MMS messages and SMTP transfer over circuit-switched data calls may however have such a large cost that it is hard to find gateway nodes. Suitable payment systems are discussed later.

#### *D. Message passing: high level description*

Everything begins when a message is created. As far as BT-Crowds is concerned, there is no restriction of what kind of message is being delivered. The size of the message may be an issue, though, and the data types used partly dictate what kind of protocol can be used to deliver this message to the final recipient.

The message is injected by a local system to the message pool of the local jondo. From this point onwards, the message is like just any other message in BT-Crowds. It does not receive any preferential treatment.

When a message enters the message pool, it is scheduled for forwarding based on several different factors. First, if there is much message traffic, the jondo may elect to randomise the order in which the messages are sent out. Most likely message volume is very small for most of the time. Therefore waiting for even two messages to arrive may be too much. Second, the messages are tagged with a latency time. After this latency time, the message is cleared for forwarding. More advanced jondos can analyse the number of other jondos they see, and if there is a large number of jondos around them, they can make the latency time longer without the fear of missing the next rendez-vous with another jondo.

When there are messages in the message pool that are cleared for forwarding, the jondo starts actively looking for other jondos. This is done using Bluetooth service discovery. When another jondo is found, an OBEX transfer is initiated and the message is transferred.

To avoid a situation where a message is passed back and forth between two devices, the receiving party stores the sending device's Bluetooth device address with the message when it is put in the pool. This is checked before sending the message, thus avoiding loops of less than three jondos. The address is deleted immediately after the message has been forwarded. Also, the sender of the message should retain the Bluetooth device address of (for example one or two) jondos it has last communicated with and avoid sending messages to these jondos. This way, all messages are not sent through the same route.

If the jondo is a gateway jondo, that is, it has a possibility of forwarding the messages to final recipients, it will determine whether to forward the message based on some probability. If

it elects to forward the message, it does so after the message in the pool would be cleared for forwarding.

To ensure that the system works more smoothly (for example, to avoid messages that go missing while being transferred through BT-Crowds), the originator of the message may send several copies of the message. Because the jondos should send each subsequent message to a different jondo, all of the copies can be injected to the originating message pool at the same time. This approach is similar to what is used by the Mixmaster anonymous/pseudonymous mail client.

#### *E. Message passing: single hop example*

Message passing between two jondos begins with the installation and start-up of the BT-Crowds application. The first thing that the application must do is to register itself to the Service Discovery Database (SDDB) of the device so that other Bluetooth devices can determine that this device is running a BT-Crowds jondo. This is done through the Bluetooth API supplied by the device. As an example, in the Java Bluetooth API, all BT-Crowds applications are running as Bluetooth servers. They must create a service record, add it to SDDB, set their security properties in Bluetooth Control Center (BCC), and then wait for OBEX PUT requests from other BT-Crowds jondos [16, p.15].

BT-Crowds applications have a double role in that they are also clients to other jondos and periodically should send out messages from their message pool using OBEX PUT. If there are messages in the message pool and their latency counters have elapsed, the application should start searching for other BT-Crowds services using Service Discovery Protocol. This is done in two phases. First, the device must be instructed to discover devices. When devices are found, a service discovery procedure is initiated. Services are distinguished from each other by their UUID (Universally Unique Identifier). If it turns out that a remote device has a BT-Crowds service, the message can be sent over OBEX to that device.

For the OBEX transmission, the client needs to determine the set of OBEX headers. OBEX headers closely resemble HTTP headers, and because of privacy issues, all unnecessary information should be weeded out. The actual data transfer is a sequence of an OBEX CONNECT request, followed by PUT, and finally DISCONNECT. Each of these requests might fail, and in that case the message should reside in the outbound queue. It may be possible to retry sending once, but if that also fails, the Bluetooth device identifier should be marked as "just seen" and not contacted any more (until other OBEX peers have been tried).

On the server (receiving jondo) side, the BT-Crowds application will get a notification of an incoming OBEX connection. It should examine the object that was transferred over OBEX and inject that to its own message pool if it is of correct type (typically message/rfc822).

As mentioned previously, it is also possible to run the system over infrared. However, service discovery does not work without user intervention. Typically the user needs to point the devices at each other and then start infrared receiving

on the other device, and sending on the other. Only then will service discovery start.

#### *F. High level pseudocode for Java BT-Crowds application*

The reason why a Java implementation of the BT-Crowds system is interesting is twofold. First, Java is platform-neutral. Almost all mobile devices are expected to run Java, specifically Java MIDP. Therefore anyone could download the same BT-Crowds application and superdistribution of the application between different device brands and models should be easy and straightforward (just beam the JAR/JAD file over using infrared or Bluetooth). Second, Java provides in many cases the minimal API for programmers. It is probable that any native APIs will provide at least the same functionality as Java APIs. Therefore, if the system is implementable in Java, it is most probably implementable also natively on most platforms.

Figure 5 describes a potential BT-Crowds implementation in high level pseudocode, building on possibilities offered by the Java Bluetooth API [16].

#### *G. Analysis of privacy properties*

When thinking BT-Crowds from traffic analysis point of view, it can be seen that cover traffic and other types of traffic shaping are perhaps not that much required. BT-Crowds transactions being distributed not only in time but also in physical space, it is harder for an attacker to conduct useful traffic analysis. If the jondos are immobile and only live in a constrained space, traffic analysis could be done by logging all traffic. Because of this, passing on the messages should be done rather slowly or by using location services as a hint of the change in geographical position (GPS and GSM cell identifiers are possible alternatives, the latter being used by many third party applications already on existing phones). If required, ideas from remailer networks can be employed in BT-Crowds to provide cover traffic. BT-Crowds does not offer path preselection (source routing) because of its opportunistic approach. This may be considered a problem in the light of fundamentals of privacy enhancing technologies. However, as the path will be chosen according to the physical surroundings of the sender's jondo, the original sender has control at least over the first hop.

The notion of a 'crowd' becomes very tangible in BT-Crowds, because the system works best in a real, human crowd. Therefore the system is expected to work best in areas which have a lot of people about, such as in city centres, cafés, public transport, shops and malls. Because many of these places are also public or semi-public places, this offers resistance against any attempts to single out a person from a crowd. On the other hand, as Bluetooth devices send out their device address, one potential attack could try to create incidence profiles of all Bluetooth devices in a given area, and determine which of these are running a BT-Crowds service. Unless BT-Crowds service becomes sufficiently widespread, this may pose an Achilles' heel for the system.

Repudiation properties of BT-Crowds depend mainly on the data object which is being routed through the crowd. The

```

Initialise message pool
Register BT-Crowds service
  Create a Bluetooth service record
  Add service record to Service Record Database
  Register security measures with Bluetooth Control Center
Start BT-Crowds service listener
  Accept connections from clients
Wait for any of the following events
Event: TIMER EVENT
  Is message pool empty?
  Yes:
    If inquiry is on:
      Cancel all pending device and service searches
      Inquiry = off
  No:
    If any messages should be cleared for sending:
      Mark messages cleared for sending
    If inquiry is off:
      Check sending policy (time of day, etc.)
    If ok:
      Inquiry = on
      Start device search
    Schedule new timer event
Event: DEVICE FOUND
  Is the device of suitable type and not on "last seen" list?
  Yes:
    Add device to list of found devices
    Start service search on the found device
  No:
    Disregard device
Event: DEVICE SEARCH CONCLUDED
  If there is an error condition:
    Process error condition
  If the device list is empty (no found devices):
    Inquiry = off
Event: OBEX CONNECTION ATTEMPT
  Receive message
  Schedule message in message pool
Event: BT-CROWDS SERVICE FOUND IN SERVICE SEARCH
  Register security measures with Bluetooth Control Center
  Initiate OBEX object exchange
  If successful:
    Update the "last seen" list of device IDs
    Remove message from message pool
  Still messages scheduled for sending?
  No:
    Inquiry = off
    Stop device and service searches
  Yes:
    Start service search on next found device
Event: SERVICE SEARCH CONCLUDED
  If there is an error condition:
    Process error condition
  If there are still more devices in found devices list:
    Start service search on next device
Event: LOCAL MESSAGE INJECTION
  Put the message in message pool
Loop repeat
    
```

Fig. 5. BT Crowds pseudocode

OBEX transactions do not convey origin data as such except for the last hop information. Objects should apply encryption to shroud the contents. As it is depicted here, BT-Crowds does not employ hop-by-hop encryption. This is because there is no infrastructure for spreading the keys, and as routing is opportunistic, the idea of source routing which is essential for onion routing is not possible. In addition, the “d  j   vu” list of Bluetooth device addresses may be backtracked if a number of devices are compromised. Therefore it would be beneficial to have the “d  j   vu” list cleaned periodically or at least have a finite list which cycles in round-robin fashion.

Luckily mixing can still be used to confuse any traffic

analysis, and the fact that any Bluetooth device that is running a BT-Crowds client can be a relay should guarantee some resilience to traffic analysis based on traffic flow patterns.

#### H. Extending BT-Crowds to send/receive system

As specified here, BT-Crowds is a one-way system. Messages can be sent, but due to the fact that routing is opportunistic and the message path cannot be guaranteed to be valid at any future point in time, bidirectionality is very hard to attain. Obviously the reply address may use a normal onion routed email system such as Mixmaster, or a broadcast system where the reply is published in a newsgroup encrypted with the recipient’s public key.

If we want to use BT-Crowds for the return path, one solution for bidirectional communications employs so-called drop boxes. Drop boxes are BT-crowds kiosks (and usually also gateway jondos) that store messages for predetermined time until a BT-Crowds client comes to pick the message up. The system requires the original sender to be aware of a selection of email addresses of potential drop boxes. The number of drop boxes can be fairly large, and geographically widespread. They would optimally placed in areas where people can hang about and use their mobile phones without looking suspect. By using directional antennas, a “drop-box” could be implemented as an area on the street, without any physical box to be seen. The list of email addresses of the selected drop boxes will be added to the message as a Reply-To: header. Each email address will be prefixed with a hash of the public key of a recipient jondo (e.g. hash%dropbox@domain). Drop boxes will hand over these messages when presented by a public key that hashes to the correct result and after authenticating the device with the public key (the recipient has the corresponding private key).

Drop boxes, however, are very vulnerable. It is easy to monitor the physical vicinity of a drop box. Therefore, it is very probable that the main benefit of BT-Crowds is in anonymous sending of messages, so it can effectively be used for one-way communication and publication.

#### I. Billing of transmitted data

Sending messages over Bluetooth is effectively free. The amount of data sent from a gateway node using a fixed Internet connection is most likely also negligible, unless there is a need for traffic shaping with a large amount of bogus data. Problems arise when the connection to the Internet is wireless. Unless gateway nodes get paid for data there will not be very many of them, and on the other hand, if gateways can make money by gatewaying messages, there is a risk of misuse.

This is a very hard problem because fair billing usually requires identification of the sender, and BT-Crowds specialises in obscuring the sender information. Even if there was an anonymous digital cash available, there is the question of who will get access to these coins. If more than one jondo will get copies of the coins, the risk of stealing these electronic vouchers will be increased.

One option would be a server where one would be able to buy vouchers in advance. The originating jondo would submit a blinded<sup>5</sup> hash value of the message to a payment server for digital signing before sending the message, and pay for its transit through the system by some out-of-band method such as a credit card. Blinding ensures that the payment server cannot correlate the originating jondo and the signed hash afterwards. The sender then sends the message, incorporating the signed hash. After being propagated through BT-Crowds, the gateway jondo sends the message to a special server, which collects the hashes and credits the gateway jondo if the hash is properly signed by a trusted payment server, has not been doubly used, and the hash corresponds to the message in transit.

Problematic areas in this payment system are that the identity of the sender (but not which message it sent) becomes known to the payment server, the identity of the gateway jondo becomes known to the gateway server, and the gateway jondos can only get paid if they forward the messages through the special gateway server. However, the message cannot be correlated with the message (except by traffic analysis, if there are not many paid-for messages), and the gateway jondo is usually identified to the target network by its IP address or telephone number anyway.

Adding some kind payment system may not only alleviate the problem of expenses but it may also guard against misuse. If, for example, sending a message involves a calculation of a hard problem (and takes time), flooding of the BT-Crowd becomes harder. This idea has been proposed in hashcash [17]. In hashcash, a communicating party has to perform an expensive (in terms of processing power) calculation, which is easy and quick to verify. The results of the calculation are sent with the message, and the recipient can discard any messages that do not contain a verifiable hashcash token. This system has been used to protect anonymous remailers from flooding.

#### *J. Reputation systems and hardening against message loss*

BT-Crowds is vulnerable to message loss. It is essentially a fire-and-forget, connectionless system that cannot guarantee message delivery. Because it is a one-way system, there cannot be any acknowledgements flowing back towards the sender. Therefore it is beneficial to shortly discuss ways in which the system can be hardened against message loss and malicious jondos.

First option is to have multiple copies of the same message flowing through the crowd. BT-Crowds is designed in such a fashion that several copies of the same message, injected to the local message pool, take different opportunistic routes beginning with the first hop. Having more copies of the same message tries to ensure that at least one of them reaches its destination. The destination will be able to do duplicate removal for example by filtering messages by duplicate Message-Id RFC 2822 headers. More messages also means more traffic, which is always a benefit for combating traffic analysis.

<sup>5</sup>Blinding is an operation where the data to be signed is shrouded and the signer does not see what it is signing.

Second option is to have a reputation system in place. In BT-Crowds, good reputation means ability to forward a message that has been received from someone else. Sending out messages that you yourself have generated is not considered a reputation bonus, because you could accrue a lot of good reputation even when dropping everyone else's messages. Unfortunately, this definition is in direct opposition of the fundamental privacy properties. You must be able to repudiate your own messages, that is, for any given message, you must be able to deny that you yourself sent it, thus meaning that someone else sent it, and therefore meaning that you should get a reputation bonus for this message. Alternatively, we could consider bad reputation. Bad reputation means inability to forward messages that one has received. We can, however, argue that having even this kind of a reverse reputation system in place adversely affects the privacy properties. Bad reputation must be imposed by someone else than the subject (a jondo which is being evaluated). This someone else would need to know both that a message has been received by a jondo, and that it is being forwarded. This is exactly the combination of information that an anonymous forwarding system tries to avoid.

A potential approach towards a reputation system can be done if we give in for the reputation definition somewhat. We could argue that if a jondo has a lot of outgoing messages and a lot of incoming messages, it might be a well behaving crowd member. (We cannot be sure, though - the jondo could just drop all incoming messages and create its own messages in their place.) The reputation system could then function as follows:

- Each jondo has a pseudonymous identity, which is a private, public key pair.
- There is a (network of) trusted reputation servers, which use some peer evaluation technique to weed out potential malicious servers.
- Each receiving jondo gives its public key to the sending jondo in a transaction, and vice versa.
- Each jondo periodically sends all public keys (of other jondos) it has accrued to the reputation servers (for example, over GPRS connection), which calculate a "reputation index" and sign it with their private key. The reputation index is essentially a public key certificate issued for the jondo's public key. The public key of the reputation server is known to all jondos, for example, embedded in the jondo's software.
- The reputation index is placed in a public directory. Each jondo may retrieve its own index (or all indices) and extract the one that belongs to it.
- In subsequent transactions, jondos can present this signed reputation certificate during the OBEX handshake, plus authenticate themselves by signing their own Bluetooth ID with their secret key. The sender of the message can then verify that the potential recipient jondo is in fact possession with the correct private key that belongs to the presented reputation index certificate, and based on the reputation index, determine whether the reputation is

good enough to warrant sending of the message.

This system is vulnerable to malicious jondos that drop incoming messages and generate one of their own. However, even though this cannot protect against this sort of attack, it can protect against jondos that never are able to send anything forward.

The scheme is also vulnerable to a group of jondos that decide to circulate message and thereby increase the 'good karma'. Protection against this system would also be difficult as due to the privacy properties, one should not be able to deduce whether the message has already passed through a given jondo.

If a reputation system is in place, it should be coupled with the multiple copies system so that some copies are also passed through systems which lack reputation. The number of copies would depend on the reputation (more copies if there are less trusted recipients). This gives prospective BT-Crowds jondos a possibility to join in the system, and also ensures that potential malicious reputation-gathering systems are not the only ones used for transactions. Essentially, the reputation system is useful for helping to determine the number of copies required, and not for weeding out bad jondos.

If the reader would like to get more information about potential reputation schemes and what incentives users could be offered to behave constructively, the reasons why users would like to operate honest nodes are discussed in [18] and reputation systems are discussed in [19].

#### K. Hopping between crowds

Jondos that either go for a long time without seeing other jondos might decide that their messages are going stale, for example due to the jondo being at the border of the crowd with less devices to talk with. If the owner of this jondo decides to absorb the (negotiable) cost that this would entail, the jondo could send the message to a bulletin board over the cellular network or wireless LAN. Other jondos could pick up these messages and forward to their own BT-Crowds them if they feel more connected. This way, the system can bridge geographically disconnected BT-Crowds. This kind of hopping exposes both of the jondos that communicate with the remote bulletin board.

#### L. Practical implementation issues

One of the main questions in implementation feasibility is that are jondos capable of operating without user intervention. As said many times before, BT-Crowds is opportunistic: messages are relayed when the opportunity knocks. This capability was already described in the requirements section, where it was divided in three parts: first, the transfer of messages must be allowed without needing the user to acknowledge each transfer. Second, there should be no need to pair or authenticate the devices to each other. Third, it should be possible to run the BT-Crowds application in the background. This implies support for running multiple (Bluetooth) applications simultaneously.

Java does not mandate that applications are running simultaneously or in the background, but BT-Crowds servers can be left dormant until an incoming OBEX connection wakes them up. For the client side implementation, the issue is more complex, because it needs periodically check if any new messages have been cleared for sending. Of course, this could be left to the user, but if the device is only capable of running one Java application at a time, the service discovery phase will effectively block any other use of the device for this time. Because concurrent Java applications are useful for many other purposes as well, it is foreseeable that many future devices will handle this.

### V. DISCUSSION

Finally, the big question: who will be running BT-Crowds? Privacy-obsessed nerds, for sure. Enlightened mobile enthusiasts, maybe. But how large a section do these groups cover? In order to be usable, BT-Crowds applications would need to exist on many, many devices. Hypothesis of crowd density could be built from some observations such as the size of public transport vehicles and compartments (dozens of people within the sphere of proximity) and by analysing real-life crowd patterns. The required density is expected to strongly depend on the type of physical crowds that exist. Cities with mass transport could probably do with tens or hundreds of times smaller density than those where people tend to use private vehicles.

Luckily, BT-Crowds could be used in conjunction with other interesting opportunistic or ad hoc services. All it needs is a freely available implementation of a killer application that contains a BT-Crowds jondo. What would this killer application be?

Could it be an automated dating system, where the user's own and preference profiles have been stored in the device, and the device scans neighbouring devices in order to find the perfect match? Possibly. Dating services to require pseudonymous communication, which could be offered by the BT-Crowds jondo. Games would also be a suitable carrier. Game concepts that are based on the ad hoc player group that just happens to surround a single player could be very interesting. Yet another idea of an application where a jondo would fit quite well would be a rumour client. You could enter a rumour into the rumour client, and it would propagate from device to device. Rumours would create virtual bulletin boards (or toilet walls). Rumours could be transferred only between devices that share a similar interest profile. Or, even more ideas: what about a Tamagotchi-like virtual pet, that lives from contacts with other users, and likes to talk with them? Bluetooth networking makes possible many new ways of socialising that are completely outside the current norms and customs. It is just an issue which ones strike a chord with the public.

### REFERENCES

- [1] M. Reiter and A. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security*, pp. 66–92, 1998.
- [2] M. K. Reiter and A. D. Rubin, "Anonymous web transactions with crowds," *CACM*, pp. 32–38, 1999.

- [3] L. Cottrell, "Frequently asked questions about Mixmaster remailers," 1996. [Online]. Available: <http://www.obscura.com/%7eloki/remailer/mixmaster-faq.html>
- [4] J. Geier, *Wireless LANs: Implementing Interoperable Networks*. Macmillan Technical Publishing, 1999.
- [5] J. Bray and C. F. Sturman, *Bluetooth: Connect Without Cables*. Prentice Hall PTR, 2001.
- [6] "IEEE OUI and Company-id assignments," 2002, IEEE. [Online]. Available: <http://standards.ieee.org/regauth/oui/oui.txt>
- [7] "Bluetooth SIG presents new Specification, and two Implementation Guides," 2003, Bluetooth SIG Press release. [Online]. Available: <http://www.bluetooth.com/news/sigreleases.asp?A=2&PID=870&ARC=1&ofs=>
- [8] "Specification of the Bluetooth System. Version 1.2," 2003, Bluetooth SIG.
- [9] K. Rantala, *Implementation of Bluetooth Service Discovery Protocol*. Tampere University of Technology, 2000.
- [10] "Specification of the Bluetooth System: Core. Version 1.1." 2001, Bluetooth SIG.
- [11] "Specification of the Bluetooth System: Profiles. Version 1.1." 2001, Bluetooth SIG.
- [12] P. Megowan, D. Suvak, and D. Kogan, "IrDA Object Exchange Protocol OBEX," 1999.
- [13] B. A. Miller and C. Bisdikian, *Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communications*. Prentice Hall PTR., 2001.
- [14] "BlueSniper Rifle and More Fun Bluetooth Exploits," 2004, Gizmodo. [Online]. Available: <http://www.gizmodo.com/archives/bluesniper-rifle-and-more-fun-bluetooth-exploits-019037.php>
- [15] "IMterview With Bluetooth Hacking Flexilis's John Hering," 2004, Gizmodo. [Online]. Available: <http://www.gizmodo.com/archives/interview-with-bluetooth-hacking-flexiliss-john-hering-019057.php>
- [16] "Java APIs for Bluetooth Wireless Technology (JSR-82): Java 2 Platform, 'Micro Edition'," 2002, Motorola.
- [17] A. Back, "Hashcash - A Denial of Service Counter-Measure," 2002. [Online]. Available: <http://www.cyberspace.org/%7eadam/hashcash/hashcash.pdf>
- [18] A. Acquisti, R. Dingledine, and P. Syverson, "On the Economics of Anonymity," in *Financial Cryptography (FC '03), LNCS*. Springer Verlag, 2003.
- [19] R. Dingledine, N. Mathewson, and P. Syverson, "Reputation in P2P Anonymity Systems," in *Workshop on Economics of Peer-to-Peer Systems*, 2003.