Bluetooth Tagging 1.1

Lawrence Botley  
Monday, 17 September 2007

**This document contains the following sections:**

1. Device discovery thread
2. Outgoing Thread to initiate connection
3. Listener thread for incoming connections
4. Terminology

**Core functionality**

* Bluetooth communications
* Disk I/O for storing Bluetooth tags
* XML read/write for storing persistent data
* Use web service (SOAP) to send tags to the N2F server – This is an ASP.NET web service. The address of the server and access to the WSDL will be available at http://12.206.33.18:66/BTTagging.aspx

**Tagging process**

The Bluetooth tagging process from the device perspective can happen in two ways:

1. A device could discover a new device in its proximity and initiate the tagging process
2. A device could receive a request from another device and complete the tagging process

Therefore the Tagging process will have three threads:

1. Device discovery thread
2. Outgoing Thread to initiate connection
3. Listener thread for incoming connections

**1.0 Device discovery thread**

The device discovery thread will call the Bluetooth radio to look for other Bluetooth device its range. It should be possible to set the number of seconds that the radio will search for. It is important that two people walk past each other in the street should be sufficient time for the matching process to take place. Therefore the discovery time should be around 3-5 seconds. This time should be easily configurable so as to tune the application in beta stages.

Once the discovery has taken place a list of MAC addresses will be returned from the Bluetooth API. From here the list of MAC addresses will be added the ***OutgoingWaiting*** array where the ***Outgoing Thread initiators*** can access them. However, before this takes places a 2 tier validation process will happen.

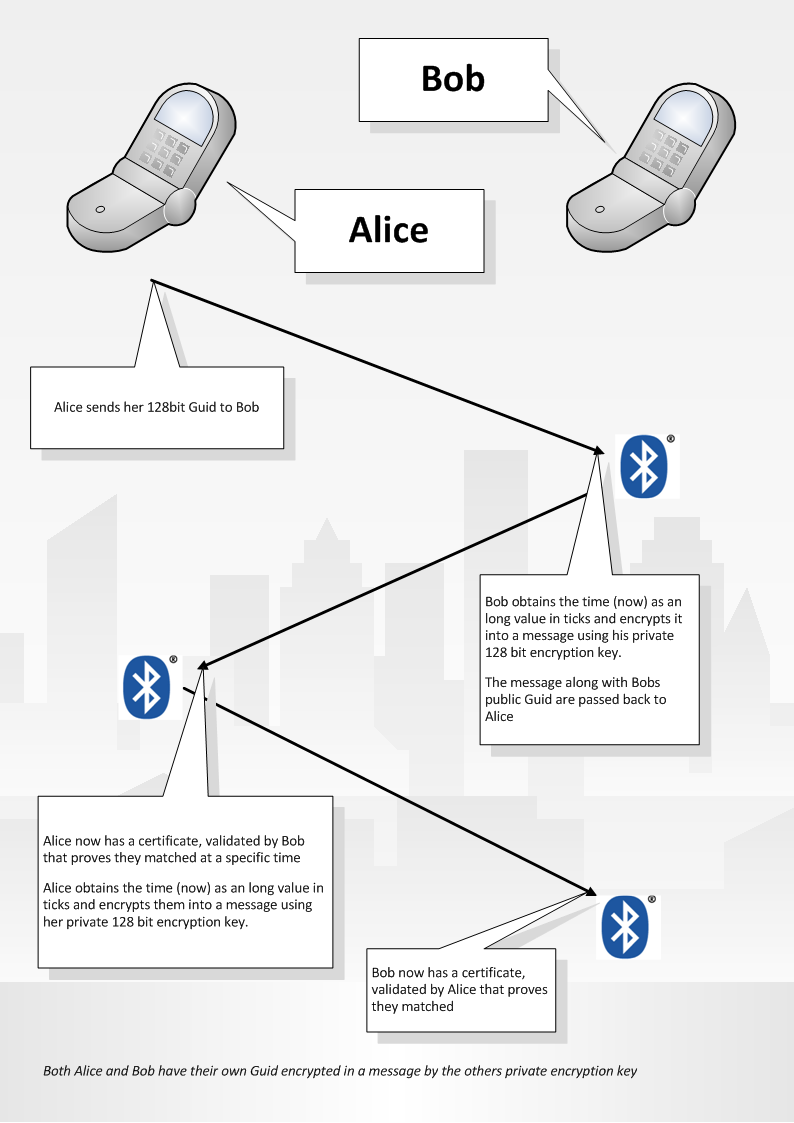
1. Check that the MAC address is not on a ***BlockList*** *[*4.1.3*]* (this is a list of devices that the individual member has chosen to ignore)
2. Check that if there is previous tag for this device, it was not in the last 30 seconds. This data can be stored and retrieved from the ***TagList*** [4.1.6] array (this time quantum should be configurable also). This will limit the number of tags and avoid ***TagThrashing*** [4.1.7]

As long as the MAC address passes the above 2 tests they can be added to the ***OutgoingWaiting****.* The device discovery thread will use and aging process to up the ***OutgoingWaiting***andon every completion of the discovery process it will scan the ***OutgoingWaiting***array and remove all items that are over 1 minute old. This will remove the number of MAC addresses that need to be check against each time, remove any clogged items that have not been process (and have probably physically moved away from proximity) and finally keep the memory footprint low.

Summary: The Discovery thread searches for new devices in range, checks them against the ***OutgoingWaiting***array to determine if it can be placed on the array, if it can then it is inserted to the back FIFO array and if it cannot then it is discarded. Finally a cleanup it performed on every iteration and removes all device references that were tagged over 1 minute ago.

**2.0 Outgoing Thread to initiate connection**

The Outgoing thread and incoming thread have similar functions. Diagram 2.1 depicts the Outgoing thread from the perspective of Alice. In this case, Alice initiates connection.



2.1 – The Bluetooth tagging process

In diagram 2.1 Alice sends her 128 Guid to Bob. Bob gets the time of NOW from the device OS (represented as a long data type) and concatenates it with Alice’s 128 Guid into a ***Tag validation String***

The unencrypted ***Tag validation String*** might look something like this:

64EAF825-ED86-4152-B1F2-BB7E4BFCC99512316848148948418489

Each member will be given a unique encryption key when they sign up to N2F which is stored in the device configuration file, this key is used to encrypt the *Tag validation String*. This *Tag validation String* is used for validation by the server to determine that the Tag was definatly made by the user and no foul play has occurred.

The encryption algorithm used will be a DES and the source is fully available is both the J2ME and Symbian version of Movino.

*[For your information: The server will validate the tag by making a lookup on the users encryption key and attempting to decrypt the string. If it is successfully decrypted then the string will contain a 36 character 128 bit Guid and a long number representing the time that the tag took place.]*

Once the *Tag validation String* has been encrypted it is returned to Alice along with Bobs 128bit guid. This information makes up the ***TagDataPacket*** *[4.1.8]* will contain the following:

* RTCE86RT7GV16T77ERT86V1Q7R3783783117865V18247C187518C785T 🡨128bit Guid and DateTime encrypted into a Unicode string
* 6743EBDF-5D6C-4a05-A768-F34BEBD0D522 🡨 Bobs Unique 128bit Guid

After the string is sent to Bob, he will store it and then encrypt Alice’s string in exactly the same way that Alice encrypted his. Once the encryption process has taken place, Bob sends a ***TagDataPacket***back to Alice. At this point the Tagging process is complete.

There can be up to 5 ***OutgoingThread*** instances alive at any point. Once the thread has completed (successfully on unsuccessfully) it is terminated so that another thread can be created for another connection. The number of threads (5) should be configurable to allow tuning for the beta stage.

**3.0 Listener thread for incoming connections**

The *listener thread* involves almost exactly the same process at the *Outgoing Thread initiators* except instead of initiating the connection it listens for incoming. We will use the same Alice and Bob example except this time we are working from Bob’s perspective.

Once Bob has detected an incoming connection from Alice a lookup is performed on the *block list* to determine if the MAC is allowed to communicate. If it is disallowed then the connection is denied and the listener will listen for other connections. If the connection is allowed then the byte stream will be received from Alice and the *Tag data packet* will be returned. The encrypted ***TagValidationString*** *is extracted* from the ***TagDataPacket***andstored in an XML file next to Alice’s 128 Guid. See XML Document for example.

Before the Guid is concatenated with the DateTime of now and encrypted, a small check is done to make sure that the 128 Guid has in fact been allocated officially by N2F and the user hasn’t injected their own Guid inside the application. If a user had done this then they could perform a denial of service on a device and pretend to be multiple devices. This would clog a user device and stop it from making valid tags.

The mini check is to be kept a secret from the outside world and involves the following process.

Add the single hexadecimal values of the two **RED** characters. In this case A (10) + 9(9) = 19. And add the values of the three **BLUE** characters at position 10, 20 and 25. In this case 2 + B (11) + 6. Therefore the magic number is 19 and if these values do not equate then the 128bit Guid has not been assigned by the N2F server and should be ignored

**A9**B6FC1F-**2**C22-43d1-**B**D96-**6**058F6619ECE

**(A** + **9)** = (**2** + **B** + **6) = Valid MemberID**

If the 128 bit Guid passes this test then it may proceed to the next 2 tests which are exactly the same as in section 2.0 of this document. To reiterate, here they are again.

1. Check that the MAC address is not on a ***BlockList*** *[*4.1.3*]* (this is a list of devices that the individual member has chosen to ignore)
2. Check that if there is previous tag for this device, it was not in the last 30 seconds. This data can be stored and found in the ***TagList*** [4.1.6] array (this time quantum should be configurable also). This will limit the number of tags and avoid ***TagThrashing*** [4.1.7]

*[Finally: there will be a process that transmits the* ***TagValidationString’s*** *to the server later on. This will be a simple web service and once the server has confirmed that it has received the* ***TagValidationString*** *then it is flagged as “ConfirmedByServer” in the XML file and is removed after the web service has finished being called. This server synchronisation process is to be addressed as another component of the system]*

1. **Terminology**

**4.1.1 Tag validation String –** This is the encrypted string containing the members Unique ID and DateTime of tag

**4.1.2 OutgoingWaiting –** An queue (array)of MAC addresses of devices that are to be tagged next by the ThreadInitiator

**4.1.3 Block List –** An array of objects (**BlockListItem**) that contains the MAC address and MemberID of a user who has been blocked specifically by the user.

**4.1.4 BlockListItem –** An data structure item in the block list (MAC address as string and MemberID as String)

**4.1.5 OutgoingThread –** The Thread that makes the outgoing connection to another device. There can be up to 5 OutgoingThreads alive at any one time.

**4.1.6 TagList –** The list of devices/members that have been tagged. This list is mirrored on disk in a XML config file and serves two purposes. 1. Is used when informing the server of new tags. 2. Is checked before an incoming or outgoing connection is made to ensure that there has been at least 30 seconds since the last tag. (this avoids tag thrashing of BT communications or **TagThrashing**)

**4.1.7 TagThrashing –** Couldoccur when devices are not properly configured and repeated tagging occurs between devices as quickly as BT communication and CPU can handle.. FYI: this is bad… 8o)

**4.1.8 TagDataPacket *–***A packet data structure sent over Bluetooth between devices that contains the encrypted Unique ID, date & time of the tag and the unencrypted Unique ID of sender.