< Made in HP >

BlueTooth roaming proposals

Jean Tourrilhes, HPL

9 October 00

Just moving around...

1 Introduction

This document describes a roaming framework for BlueTooth, with lot of possible techniques to overcome some of the challenges specific to BlueTooth. Maybe what is written here make sense and is implementable, but the sole aim of this document is to stir discussion within the PAN-APR group ;-)

2 Goal

BlueTooth defines two type of networking support, the LAN Access Profile (using PPP) and the Personal Area Network (using BNEP). In both case, the assumption is that a specific BlueTooth device called an Access Point enable BlueTooth client devices to be connected to an infrastructure. This is similar to the definition of Access Points in IrDA and 802.11.

BlueTooth has a limited range (from 15 cm to 100 m depending on implementation). To be able to cover a large area, many Access Points need to be deployed. We would like them to be deployed in a regular cellular fashion, but people will put them where they want or where they can. There can also be multiple logical BlueTooth network co-located and owned by different entities.

Most BlueTooth devices are personal device, and therefore move around with their owner. There is not much we can do to prevent them to do that, so we better support this mobility. One good way to support mobility is to allow device to move freely from one Access Point to another without the need for user intervention and in a timely fashion. Even better would be to migrate transparently network connections.

3 Approach

In this document, we will define a basic roaming mechanism. This mechanism is the baseline, and we will add some other optional mechanisms to improve it (some optimisations).

The main goal of the basic mechanism is to be simple, robust and to work in every situations. We want it to work whatever the device, the implementation details, the conditions and the environment, and to be easy to implement and get to operate in all possible devices supporting LAN Access or PAN. However, this basic scheme will have some performance issues.

The optimisations described later will deal with a lot of frequent generic cases where it is possible to improve roaming and overcome some of those issues by making more assumptions on the capacity of the device, the infrastructure configuration or the operating environment. The aim of those optimisations is to be simple, fit on top of the basic scheme and offer some tangible value.

4 Basic mechanism

The basic mechanism doesn't assume much and is quite simple.

The BlueTooth device is connected with an Access Point, and move out of range. After some time (*supervisionTO*, see 10.11), the BlueTooth stack in the device declares the link to be dead and close the LMP connection. At this point, the L2CAP connection used by the network traffic receives an event (LP_DisconnectInd) and close down. The higher layer are kept in suspended mode until the roaming procedure is completed (TCP/IP may timeout if we take too much time).

At this point, the BlueTooth device performs an Inquiry. If the inquiry doesn't find any Access Point, the node continues with inquiry, up to the point where it timeout and close down the higher layer (either PPP or BNEP).

If the BlueTooth device finds an Access Point, it connects to it and query its SDP service record. If the "Service Name" of this Access Point is the same has the one of the previous Access Point, the device connect to it. If the "Service Name" is different, and if the device discovered other Access Points in the Inquiry process, the device should try to connect to those Access Points. In other words, we are using the "Service Name" as a logical network identifier, like the ESSID in 802.11.

If the Device can't find a Access Point with the same "Service Name", it can either try to connect to one of the other Access Point, continue doing Inquiry, or return a failure to the user. A Access Point with a different "Service Name" is managed by a different authority and may not offer the same connectivity options, so there is no guarantee that we can connect to it, have similar services and resume its suspended connections.

When the device connect to the Access Point, it registers to it and opens a L2CAP connection with this Access Point, and then reconnect the suspended higher layer on top of this L2CAP connection. For PPP, we need to establish a new PPP connection (LCP and IPCP) and redirect IP traffic on top of this new PPP instance. For BNEP, we should be able to reconnect the old BNEP instance on the new L2CAP connection (except if we change IP subnet, which should happen only if the "ServiceName" is different.

For the Access Point, it is as if the device would connect to it for the first time, and it just needs to pass the IP traffic back and forth. The old Access Point would also timeout, close the L2CAP connection and the higher layers, and flush its data buffers.

5 Challenges

Roaming is already something that other technology implement, for example 802.11 has strong support for roaming. However, BlueTooth has a certain number of features making roaming more challenging and interesting.

5.1 Discovery latency

The whole discovery process in BlueTooth takes a long time. For us, we define the discovery as the process by which a BlueTooth device get to know the presence of a device and connect to it. Usually, a discovery is composed of 3 phases, Inquiry, Paging and higher layer connection.

The Inquiry is usually the slowest. For example, a full Inquiry as defined in the spec last in excess of 10 s, and it doesn't guarantee that all the devices that need to be discovered are discovered (in case of fading or interferences). Such a long time has a high probability to interfer with the user experience and TCP/IP connections.

This time is dependant of the inquiry time itself (how many frequency scanned) and the time between inquiries for the Access Point. For the Access Point, there is clearly a tradeoff between throughput and discovery latency, if we increase the time between inquiries we allocate more time for data traffic but discovery will be slower.

The remaining part of the Discovery is faster. Paging, when done after a discovery, is pretty quick (below 100 ms). Connecting the higher layer is pretty quick as well ; establishing a L2CAP depends on how many L2CAP options are exchanged but should be well below 50 ms, for PPP it should be below 100 ms (assuming a good implementation), and for the PAN as well (in fact, in most cases PAN should be faster than PPP).

So, a good implementation should be able to go from Start of Paging to IP flowing in less than 250 ms. Of course, I can be wrong, and in a noisy environment, it could take much longer. But this is the type of delay that won't affect too much the user experience and TCP/IP.

To summarise, the discovery latency is mostly due to the Inquiry process. Doing the inquiry more efficiently or skipping it would improve the latency time.

5.2 Need to connect to identify AP

For protocols which are designed from the ground up as a cellular system, such as 802.11, the task of identifying an Access Point is pretty straightforward. The packets sent as part of the discovery process (called Beacons) identify if the node is an Access Point or not and advertise the logical network it is part of (i.e. in 802.11 the Beacon contains the ESSID of the logical network).

In BlueTooth, this is not the case. At the end of the Inquiry process, the BlueTooth device just has a list of devices that support the LAN Access profile or the PAN profile. By adding the correct information to the FHS packet (see 4.4.1.4), we should be able to narrow down to the list of Access Points in the vicinity.

But, this is not good enough, because we aim to support co-located logical networks and we want to favor handover within the same logical network and IPsubnet as much as possible. Moreover, Access Points may be set up with different access right and security settings, so the node may not be able to use all Access Points that he founds.

The information needed to select the right Access Point is part of the SDP record. And to access the SDP record, the device needs to connect to the relevant Access Point and make a SDP query.

This mean that when performing roaming, the device may need to connect to multiple Access Points, query their SDP, until it finds the right one. Each connection and query is quite fast, but they add up, so if there are lots of private Access Points or co-located networks, this may take some time.

To summarise, the problem is that we don't know which Access Point to connect to. If there are to many choices, finding the right one may take some time.

5.3 Lack of RSSI

The third issue is that many BlueTooth devices will lack RSSI (Received Signal Strength Indicator). The RSSI measures the power of the incoming radio signal, and is commonly used in CSMA systems to assess the state of the channel (free or busy). BlueTooth, as it uses a TDMA channel access mechanism, explicitly doesn't require a RSSI circuit in its implementation. Implementing a reliable and sufficiently accurate

RSSI in the radio is expensive, so it is our expectation that most BlueTooth device won't offer any RSSI (and it won't be good enough in most device that does).

The lack of RSSI has definitely strong implications on roaming. When a node connect to an Access Point, we would like to pick the best Access Point. There might be multiple Access Point offering the same service (effectively overlapping cells), and most often the best one is the one from which we receive the strongest signal (because we will get a lower BER and we will decrease the frequency of handovers).

Without a RSSI, we have very few way to select the best Access Point. BER (Bit Error Rate) would work only in the lower range of the RSSI, if there is no interferers and would be less reliable. And we don't want to rely on a location system...

In most wireless systems, RSSI is also used to detect when to perform a handover. Basically, when the RSSI is below a certain level, the device decides to search for another Access Point giving better service. Detecting a link failure without RSSI is tricky, because fading and interferer may make the link unavailable for some period of time. The only option is to trigger handover when the link has been inactive after a time long enough (*supervisionTO* in BlueTooth). But, this doesn't allow to do pro-active roaming and increase the time it takes to perform roaming.

To summarise, without RSSI we don't have a good way to select the best Access Point, to assess the link quality with the Access Point and to detect efficiently when we should switch Access Point.

6 Optimisations

The basic scheme for handover described above is very simple, but unfortunately not very efficient. In this section, we will propose a set of optimisations to improve the performance of handover.

Most of those optimisation are simple and bring some tangible benefits. Those optimisation will assume some common and usual network configuration, but may not work all the time. Those optimisations may be used alone or can be combined to improve the overall performance. Actually, some implementations depend on others.

The first optimisations will be generic and won't deal with the specificities of BlueTooth. On the other hand, the second set of optimisation will be dedicated to BlueTooth, by addressing BlueTooth specific issues and using BlueTooth specific features.

6.1 AP to AP communication

One of the most common optimisation of 802.11 networks is to have an Access Point to Access Point protocol. The goal of this protocol is to flush the ressource that were use by the roaming node in the old Access Point.

In the wireless environment, because of interferers and fading, a wireless link can be blocked for quite a bit of time before data can flow normally again. The same is true for node that use power saving and only exchange data with the Access Point infrequently. For this reason, the Access Point assumes a node is no longer associated with it only after a long timeout, to avoid disrupting a connection already potentially weak.

On the other hand, every node associated with an Access Point consumes some ressource in the Access Point. The Access Point has to keep a record of the association (node characteristic, security parameters, connection information) and buffer packets

going to this node. Even more critical in BlueTooth is the fact that there can be only 7 nodes associated with an Access Point, so we would like to know if these 7 link addresses are put to a good use.

When the node starts the roaming procedure, it doesn't know if it will be successful, so doesn't terminate its existing connection with the Access Point (in case it needs to come back). In most case, the node will have already lost contact with the Access Point when starting the roaming procedure. So, basically the node doesn't inform the Access Point when it is roaming away.

On the other hand, when the node has been associated with a new Access Point, the new Access Point can broadcast this information to the other Access Points on the wired backbone, so that the old Access Point can learn about it and flush all the resources associated with that node (registration record, packet buffers and link address).

If the time to perform handover is significantly faster than the timeout used by the AP to de-associate a node, this can result in significant ressource saving in the Access Points. We can increase the number of nodes active per Access Point, therefore reducing the number of Access Points needed. Our hope is that we will manage to get the handover time short enough that this optimisation is worth it ;-).

6.2 Optimised AP handover

A lot of people in the academia have been doing work on optimising generic handover. They have studied the interaction between the long time the link layer is blocked and TCP timers, and how to optimise the TCP recovery.

All those optimisation are of course applicable to the BlueTooth case. The first classical optimisation is to pass buffered packets from the old Access Point to the new Access Point.

While the node is performing its handover, the old Access Point still continue to receive and buffer packets for the roaming node, but can't deliver them. When the node is associated with the new Access Point, the new Access Point start to receive packets for that node at this point, but doesn't grab earlier packets. In other words, all the packets sent from the infrastructure while the node was performing the handover are not received by the node.

Those packets will need to be re-sent from the original source, which will take time and will reduce the TCP window, increasing the time it takes TCP to recover. On the other hand, the old Access Point has stored most of those packets, and they are just waiting in its buffer.

When the old Access Point receive the de-registration message from the new Access Point (see *section 6.1*), it can simply re-send all the packets in its buffer associated with this node. The new Access Point will pick them up and deliver them to the node.

In most cases, this should speed up significantly the TCP/IP recovery and overall increase performance and reliability of the higher layer through the handover.

6.3 Getting neighbour AP list

One of the challenge that we have identified is that when the node need to roam, it has no way to identify which Access Points available in its vicinity offer the same service as the one it was using (part of the same logical network) amongst all the Access Points available. This is explained in *section 5.2*.

There is a very simple solution for that. As part of the registration process, or on a regular basis, the Access Point could send to the node the list of Access Points in the vicinity that offer the same service. When the node perform the basic handover procedure, it simply compares the result of the inquiry with the list of Access Point and pick an Access Point in the common subset.

This scheme would save the node all the trial and errors of finding the right Access Point, which would decrease both handover time and power consumption. While roaming, the node doesn't need to query the SDP of all Access Points around, because it knows in advance which Access Points to connect to.

Now, we need to define how this list sent from the Access Point to the node. I would suggest to sent 2 levels of the hierarchical tree of neighbouring Access Points. In other words, the Access Point would send the list of its immediate neighbours, and for each neighbour the list of its immediate neighbours.

Having neighbouring Access Point one hop and two hops away should be good enough to cover most roaming speed and environment (most often you would roam one hop away, and most remaining cases 2 hop away). Limiting to 2 hops will allow to limit the amount of data exchanged (this increase exponentially with the number of hops). The fact that the data is presented in a hierarchical way duplicates some information (some 2 hops APs will appear twice), but allows to favor roaming one hop away and will be used in a later scheme (see *section 6.6*).

For each Access Point, we also want to transmit timing information related to paging and inquiry. We will also use that later (see *section 6.5* ;-).

To summarise, by having the Access Point sending the node the list of its neighbouring Access Points, the node doesn't need to query each Access Point, which significantly improve the performance of roaming.

6.4 Building neighbour AP list

One of the problem of the previous scheme (see *section 6.3*) is to generate the list of neighbouring Access Points. Some setups will be fixed and managed, so we can expect a network administrator to enter the list of neighbours as part of the configuration of the Access Point. But, there are many cases where the network is not tightly managed or is dynamic, and this solution is not good enough.

On the other hand, when the network is not tightly managed or is dynamic, cells tend to be widely overlapping. This is because there is no strict network design, so the placement of Access Point and distance between them has to be conservative. Also, overlapping cells increase reliability and can be used for load balancing.

If we have widely overlapping cells, Access Points are within range of their neighbours, so Access Points can discover each other through BlueTooth inquiry. By querying the SDP record of the Access Points it can see, an Access Point can build automatically the list of neighbours.

Then, using the AP to AP protocol on the wired medium, the Access Point can interrogate its neighbours for their neighbours. This way, the list of Access Point to be passed to the mobile node can be automatically constructed and dynamically updated without any human intervention.

The advantage of this scheme is that more network deployment and configurations can use the previous optimisation (see *section 6.3*).

6.5 AP presence checking (AP probing)

The previous optimisation (see *section 6.3*) was dealing mostly with the second phase of the basic roaming procedure (querying APs). However, the first phase (Inquiry) is also a major hurdle.

As part of the Access Point list (see *section 6.3*), we pass timing information for each Access Point. Basically, for each Access Point, we want to know when it performs inquiry, inquiry scan, page and page scan, what is its clock offset and its scanning pattern.

Then, using that information, the node can regularly check if it can see any of its neighbouring Access Point. As the node has all the necessary information, it can probe the Access Point only when it knows it is performing inquiry and paging, and should find it very quickly. In most case, it doesn't need to connect, but just receive a single FHS packet sent by the Access Point (we may want to connect to do PiggyBack RSSI - see *section 6.7*). By doing some regular probing of the various Access Points synchronised to their inquiry or page patterns (i.e. probing at the right time), the node can determine which Access Point it can reach and which it can not with minimal overhead.

One of the issue is for the node to hold its link to the Current Access point while it's doing this fast probing of the other Access Points. Hopefully, this time can be made very short, and the IPSS subgroup will tell us how to do it ;-). The frequency of the probes can be adjusted depending in various parameters (such as roaming history, data load and bit error rate).

The big advantage of building the list of reachable Access Points is that when the node loose its link with the current Access Point, it knows in advance which Access Points are reachable, so doesn't need to perform a full inquiry and can connect directly with the Access Point it desires. This connection is faster because the node already knows the timing of the Access Point (paging and co). This is also more reliable, because during a single inquiry procedure we may not see some nodes due to fading or interference, whereas this procedure aggregate data over a longer period of time.

In other word, we use the information sent to the node by the Access Point to perform continuously some targeted fast inquiries, so that we don't have to perform a full inquiry later.

6.6 Position estimation

The only challenge that we haven't tackled so far is the fact that we don't know which Access Point gives us the best quality of service (see *section 5.3*). This prevent us to choose the best Access Point and to know when to perform pro-active handover.

On the other hand, in the previous optimisation (see *section 6.5*), the node check which are the Access Points it can reach. And, the node also knows the topology of the Access Point deployment (which Access Points are close to which - see *section 6.3*).

By correlating those two informations, the node can decide which is the best Access Point. For each Access Point, the node calculate how many of the Access Point neighbour it can see. The Access Point for which it can see the most neighbours will usually be the closest Access Point. If this Access Point is not the current Access Point, the node should start the handover procedure.

In fact, if the list of Access Point is build automatically (see *section 6.4*), what we are trying to do is to find the Access Point which radio environment is most similar to ours, and in most case this is the closest and best Access Point.

In other words, by correlating the list of Access Points the node can reach with the hierarchical list of Access Points, the node may be able to estimate its position relative to the Access Points and select the best one. However, this optimisation is rather crude, does an approximate job and will not work reliably in a certain number of cases.

6.7 PiggyBack RSSI

As we have mentioned earlier (see *section 5.3*), the RSSI is still the best way to determine the quality of the link with the Access Point. The problem is that we can't assume that BlueTooth devices will be equipped with it. On the other hand, the Access point is a more expensive device, is a dedicated ressource shared by many users and handover is an essential feature of it.

So, we may imagine that some class of Access Points will be equipped with RSSI, and they will be able to assess the quality of the link for each node they support. Then, we can imagine a simple protocol where the Access Point piggyback the RSSI measure to the node. Knowing its current transmitted power, the node can then estimate the quality of the link (and can do power control if needed).

This piggyback RSSI information can also be used to detect when to roam, as in 802.11. When the measure goes below a certain threshold, the node trigger handover. However, this piggyback RSSI will be rather infrequent (to avoid overhead), so it can be used only for relatively slow roaming.

A more interesting use of PiggyBack RSSI would be for the probing procedure, when the node is checking presence for Access Points (see *section 6.5*). The idea is that when the node probe for the presence of an Access Point, the Access Point return to it the RSSI measure of the signal it receives from the node. If the node collect those PiggyBack RSSI from various Access Points, it could rank the various Access Points it can connect to based on it, and could choose the best when times come to roam.

Then, we can also benefit from that improved probing procedure to estimate the node position (see *section 6.6*). By comparing the PiggyBack RSSI of the various Access Points probed, the node can see if any offer better service than the current one and can also derive a more accurate relative position. This way, the node has a much better idea of when to roam, and can do pro-active roaming (instead of waiting until it's too late).

By having the Access Point returning a PiggyBack RSSI as part of the probing procedure, the node can rank the various Access Points in term of quality of the link and decide more accurately when to perform handover.

6.8 Fast AP connect

Using the previous optimisations, we can now do pro-active roaming based on link quality (see *section 6.7*) and our estimated position relative to the Access Points (see *section 6.6*). We may also need to do roaming for load balancing reasons (because an Access Point can accept only 7 nodes).

Connecting to the new Access Point is easier, because we know its timing information (see *section 6.5*). However, after we have decided to connect to an Access Point, we have still to wait until this Access Point is ready to accept our connection, that is to wait until the Access Point is in inquiry scan or paging scan mode.

But, in all the cases where we do pro-active roaming, we still have a working connection to the old Access Point at the point we want to connect to the new Access Point, and this allow us to do a simple optimisation to bypass this waiting time.

At the point when the node wants to connect to the new Access Point, it sends a specific message to the new Access Point via the old Access Point and the wired backbone. When the new Access Point receives this message, it sends back a confirmation via the wired backbone and the old Access Point and put itself in page scan mode. Then the node put itself in page mode and connect immediately to the Access Point. The advantage is that the node doesn't have to wait until the Access Point put itself in the right mode and it can bypass inquiry, which save both time and power.

By contacting the new Access Point through the old Access Point and the backbone, the node can trigger page scan mode on the new Access Point and connect immediately to it instead of having to wait.