Improving Connection Times for Bluetooth Devices in Mobile Environments

Erik Welsh, Patrick Murphy, J. Patrick Frantz
Rice University
6100 Main St – MS380
Houston, TX 77005
USA
{welsh, murphpo, jpfrantz}@rice.edu

***Abstract* – Communications devices in a highly mobile environment need to minimize connection setup times in order to maximize useful data transfer. In this paper we investigate - empirically and through simulations - the device discovery process of Bluetooth™, a technology that has potential in short-range, high-mobility applications. In order to improve Bluetooth's performance in a mobile environment, it is desirable to lessen the amount of time it takes to set up connections between two devices. We suggest three possible changes to the Bluetooth specification: eliminating or decreasing the random backoff delay in INQUIRY SCAN, using a single frequency train instead of two in INQUIRY, and a combination of the two. These simple changes can dramatically decrease connection setup times without having deleterious effects on overall system performance.**

I. INTRODUCTION

One frequently cited usage example for Bluetooth is the pedestrian equipped with a Bluetooth-enabled device such as a cell phone. In this scenario, the individual receives information about her surroundings via Bluetooth access points – e.g. nearby shopping, restaurants and other services. Since Bluetooth has both limited range and bandwidth, this mobility scenario raises some serious questions about the length of time it takes to establish a connection between two Bluetooth devices. If this process takes too long, then either no connection can be made or there will not be enough remaining time to do any useful data transfer. In our study, we are interested in an even more demanding application in which fast-moving vehicles would exchange information as they pass each other. In this environment, the Bluetooth devices are highly mobile. It is also likely a dense environment in which there could be tens or hundreds of devices in the vicinity.

TABLE 1
EFFECT OF SPEED ON CLASS 1 (20dBm) BLUETOOTH DEVICES

Speed of Vehicle		Max Time	Max 1-Way Data
km/hr	m/s	In Range (s)	Trans. (MBytes)
20	5.56	36.00	3.18
40	11.11	18.00	1.59
60	16.67	12.00	1.06
80	22.22	9.00	0.79
100	27.78	7.20	0.64

Table 1 shows some simple speed calculations for a Class 1 (20dBm) Bluetooth device. In these examples, one Bluetooth device is moving at a constant speed, and the other is stationary. Since our application involves outdoor transmissions occurring at high speeds, we assume that a Class 1 Bluetooth device (100m range) will be necessary.

Table 1 indicates that it is indeed possible to use Bluetooth in a mobile environment, provided that the expected amount of data transfer is fairly small. However, it is also clear from the calculations in Table 1 that a lengthy connection setup time will severely inhibit Bluetooth's usefulness in short-term ad hoc connections between fast-moving objects. These calculations assume that connections are instantaneous; therefore, any length of connection setup time will decrease the data transfer capability. Our objective in this paper is to characterize the device discovery process and introduce some simple methods to improve the performance of connection setup in Bluetooth devices in order to maximize the amount of data transfer. We will present three methods of speeding device discovery: reducing/eliminating the random backoff in the INQUIRY SCAN state, implementing a single frequency train in the INQUIRY state, and a combination of these two. We will also present data for each of these schemes that shows how the modifications affect collisions that can occur in the device discovery process.

In the rest of this paper, we will first give a brief overview of the Bluetooth device discovery process (Section II). Then, we will present some empirical studies of the discovery process with real Bluetooth hardware (Section III). Next, we will present some simulation results of modified inquiry processes that show improvements to the Bluetooth connection setup times (Section IV). Finally, we will provide some analysis and concluding comments (Section V).

II. THE INQUIRY PROCESS

The inquiry process is a means for one Bluetooth device to discover other Bluetooth devices. If a device wants to participate in Bluetooth piconets, then it will spend some of its time in an INQUIRY state (looking for other devices) and some of its time in an INQUIRY SCAN state (being looked for). It is important to remember that this process is only used to determine the existence of other Bluetooth devices and has no relation to the number of connections that a Bluetooth device can concurrently maintain. After discovering other devices or being discovered, an individual device can move into or resume other states.

For our characterization of device discovery we are primarily concerned with some of the timing parameters associated with the inquiry process, as well as the behavior

of devices in INQUIRY SCAN state. We will not embark on a thorough explanation of the inquiry process here because it described fully in the baseband specification [3]. We will, therefore, only briefly describe the details that concern our experiments. To ease the understanding of this process, it is useful to think of one device being in a perpetual state of INQUIRY and the rest being in INQUIRY SCAN. Much of our study assumes this case for ease of analysis.

Fig. 1 shows a broad overview of some of the timing parameters we are interested in for this study. Bluetooth implements frequency hopping at 1600 hops per second with a corresponding time slot length of 625μs. In the inquiry process, Bluetooth devices hop through a set of 32 common frequencies.[1] The potential master (in INQUIRY STATE) breaks this set into two 16-hop trains, A and B. It hops through the frequencies in each train at twice the normal rate, repeating the train at least 256 times (2.56s) before switching to the next train. During this process, the device is sending inquiry packets on every frequency. To find all devices in an error-free environment, the length of time a device spends in inquiry, $T_{w\_inquiry}$, must consist of at least three train switches (10.24s) [3].

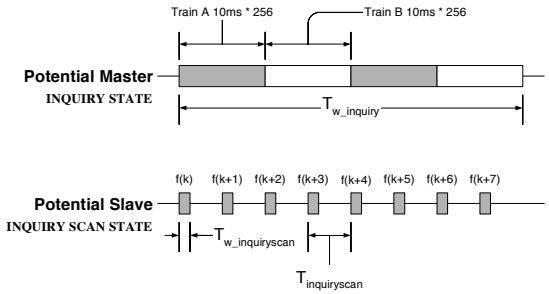

Fig. 1- Bluetooth Inquiry Timing Parameters

The potential slave (in INQUIRY SCAN STATE) also hops through the 32 common frequencies, but at a much slower rate. The phase of the hopping sequence changes every 1.28s. Periodically, the device listens for inquiry packets on the current frequency. This interval, $T_{w\_inquiryscan}$, lasts at minimum of 10ms – long enough to listen for all 16 frequencies in train A or B. The default length, however, is 11.25ms to compensate for minor offset in clock values between devices. The parameter $T_{inquiryscan}$ defines the amount of time between active scans and must be between 1.28 and 2.56s.

Fig. 2 shows the state diagram for a device entering INQUIRY SCAN state. This state can be entered from STANDBY or CONNECT at an interval defined by $T_{inquiryscan}$. In this state, the device will actively listen for inquiry packets for $T_{w\_inquiryscan}$. If no packet is received, the device can reenter INQUIRY SCAN or go back to STANDBY or CONNECT. If a packet is received, the device implements a random backoff between 0 and 1024 timeslots (0 to 640ms). This random backoff helps reduce the number of collisions among responding devices. In addition, the device can also enter the PAGE SCAN state before continuing. After the random backoff, the device reenters INQUIRY SCAN to listen for another inquiry packet on the same frequency. If no packet is received, the device returns to STANDBY or CONNECT. If a packet is received, then the device sends an identifying FHS packet to the inquirer and adds an offset to the INQUIRY SCAN hop sequence. It then continues scanning, but on a different hop frequency. By responding in this manner, a potential slave will respond on average 4 times during a 1.28s probing window, but on different frequencies and at different times [3].

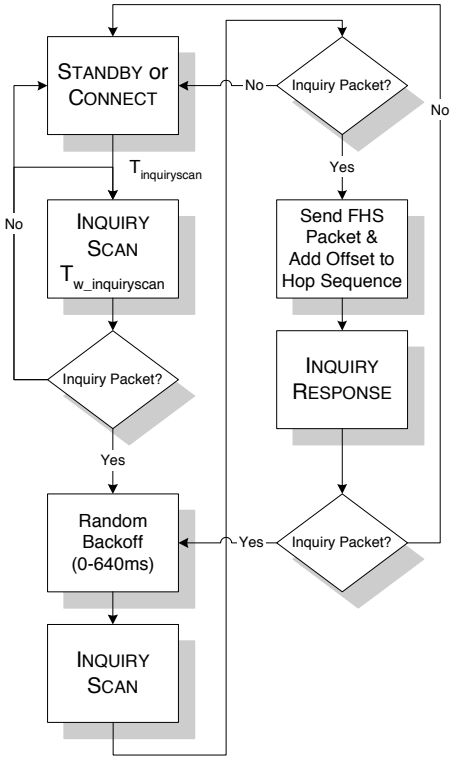

Fig. 2 - State Diagram for INQUIRY SCAN

We briefly mentioned collisions in the previous description of INQUIRY SCAN and more needs to be said on this topic. A collision is defined by two devices in INQUIRY SCAN transmitting their FHS packets at the same time and on the exact same frequency in response to an inquiry packet from an inquirer. When this happens the inquirer will get no useful data since the two packets will interfere with each other. Since there are 32 distinct frequencies in the inquiry process, and since each device in INQIURY SCAN is only actively listening for a short period of time, there is only a small chance that a collision will take place. To make the chances even smaller, the specification calls for the implementation of a random backoff delay as described above.

[1] In the 79-hop system there are 32 frequencies. The 23-hop system uses 16. We will concentrate on the 79-hop system in this paper because it is far more common.

III. EMPIRICAL ANALYSIS

Previous investigations into Bluetooth links have suggested 2 seconds as a typical setup time between two unknown devices [1], although some experiments have shown much higher values of 10s and greater (for unknown reasons) [2]. A quick read of the Bluetooth specification [3] also indicates that 2s is a reasonable number, and that 10s should be an absolute maximum.

For this study, we first decided to empirically measure the connection setup times of various Bluetooth devices. We did this to verify the numbers quoted in other sources and also to gauge the promise of Bluetooth in a mobile and dense environment. To do this we set up two experiments. The first measures the latency in receiving the first reply to an inquiry. The second provides a distribution of connection times for individual devices.

For our empirical tests we used five Bluetooth devices from various manufacturers:

- Two Ericsson Bluetooth modules with Class 3 radios each hosted by a Linux-based embedded systems development board made by Axis Communications
- Two Texas Instruments Bluetooth modules with Class 1 radios each hosted by Windows 2000 PC's
- One Toshiba Bluetooth PC Card with a Class 1 radio hosted by a Windows 98 laptop

We chose one of the Ericsson modules, hosted by the Axis development board, to act as the inquirer in these experiments. The software used in this system is an open-source implementation of the Bluetooth stack [4]. We were able to use this stack to interactively issue device discovery requests and to accurately time the responses.

In our first experiment, we set the inquirer to stop after it had received the first reply. Table 2 shows the results of this test. When there are multiple Bluetooth devices to be discovered – as is the case in this test - the first discovery occurs quite rapidly (0.79s on average). In addition, we found that the connections were distributed evenly among the four devices, such that no device was more likely than another to be the first device discovered. This results of this experiment were encouraging, because in a very dense or highly-mobile Bluetooth environment, one may not want to communicate with all Bluetooth devices, but rather just the first few that respond.

TABLE 2
LENGTH OF FIRST CONNECTION ESTABLISHMENT
(1 MASTER, 4 SLAVES)

Number of Tests	Mean Inq. Time (s)	Median Time (s)
1142	0.79	0.53

For our second test, we measured the connection time to each individual device. We did this because the connection time can theoretically vary between devices if the clocks are not initially well synchronized, as can be the case in devices from different manufacturers. Table 3 shows the results. We can see from this experiment that the average connection time to an individual Bluetooth device is, indeed, around 2s, although the median is much lower. Furthermore, there are not tremendous variations between different Bluetooth devices. These empirical results are quite similar to other studies [7], which suggest that these numbers make a good baseline for comparison.

TABLE 3
CONNECTION TIME TO ONE BLUETOOTH DEVICE

	Number of Tests	Mean Inq. Time (s)	Median Time (s)
Device 1	600	2.26	1.72
Device 2	505	2.63	1.95
Device 3	610	2.11	1.35
Device 4	499	2.10	1.36
All Devices	2214	2.27	1.44

IV. SIMULATION RESULTS

While using real Bluetooth devices is informative, it is also somewhat limiting. In general, we do not have access to the inner workings of the baseband protocol in real Bluetooth devices. In order to better characterize the device discovery process, we decided to use a simulator. We started with IBM's open-source Bluetooth simulator, BlueHoc [8]. We found, however, that BlueHoc did not yet implement certain features we were interested in, such as collision detection. Therefore, we developed our own simulator, RIBBIT[2] (RIce Bluetooth Baseband Inquiry Tester), which uses a small part of the BlueHoc source as a base.

To improve connection setup performance, we decided to modify two inquiry parameters: eliminating the differentiation between trains A and B (i.e. we implemented just one train of 32 hops and increased the duration of $T_{w\_inquiryscan}$ to 21.25ms), and eliminating or reducing the amount of time that a device performs random backoff in INQUIRY SCAN. We also test the combination of the two. Several factors influenced our decision. First, we felt that these changes were relatively easy to implement in the baseband, and would therefore not impact the cost or complexity of hardware or software. Second, the changes could be easily incorporated into future Bluetooth specifications while retaining a measure of backward compatibility. Third, we felt that these changes would offer better connection times without degrading performance (e.g. increasing the number or rate of collisions).

The RIBBIT simulator runs in Linux and uses the GNU C library function `random()` to generate pseudo-random numbers for the simulations based on an initial seed. In order to assure that the simulations used a different set of psuedo-random numbers, a random number from 1 to 1000 was passed in as the seed to `random()`. In addition, to eliminate individual variation within each trial we ran each simulation 5000 times and averaged the data returned by the simulator.

[2] For more information about our simulator, visit the RIBBIT simulator home page at http://koala.ece.rice.edu/bluetooth/ribbit/

We ran the same tests in our simulator that we ran in our empirical study. The connection setup time numbers we received from the simulator were slightly higher but were comparable to the numbers we received from our empirical analysis, leading us to believe that our simulator was providing accurate data. However, we did notice some unusual behavior with regard to collisions when there were very few devices in the simulation. The simulator does not account for two device clocks being unsynchronized, which is the case in a real environment. Because of this, there is a higher chance that two devices clocks will be very highly synchronized. This will result in an unusually high number of collisions, especially in the case of no random backoff.

Fig. 3 shows the results of our control case. The purpose of this was to show the decrease in connection time to one device as the number of devices increases. This was also the case against which we measured the performance of our three schemes. This plot shows that as the Bluetooth environment grows denser the rate of collisions increases, although it still remains quite low even when there are 100 devices. In this dense environment, only about 0.002% of all inquiry packets result in a collision.

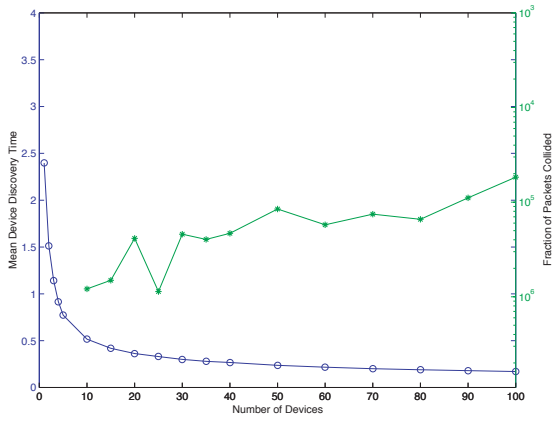

Fig. 3 - Connection Time and Collisions vs. Number of Devices

Fig. 4 shows a normalized plot of three connection time improvement schemes without regard to collisions. The control gives a baseline with which to compare. This is the case with no changes to the baseband algorithm. The average time to connect to 1 device has been normalized to 1 for each data point. As can be seen, all three schemes offer dramatically better improvement than the control case. Using a single hopping train instead of two separate trains offers best performance when there are just a small number of devices present, and eliminating the random backoff performs better as the environment grows denser. Combining the two schemes results in the best improvement, and offers at least an 70% reduction in the time it takes to connect to the first Bluetooth device.

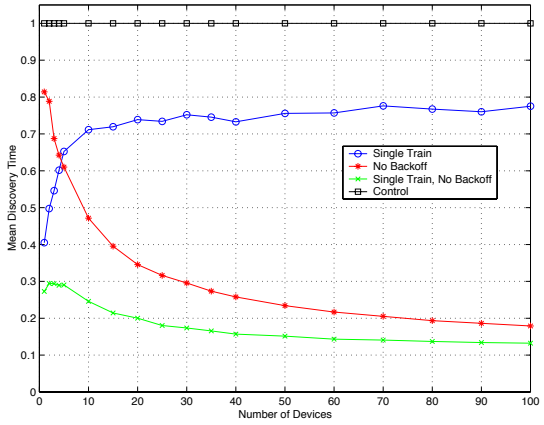

Fig. 4 – Discovery Time for First Bluetooth Device (Normalized)

We discovered during our simulations that our three schemes not only decreased the time it takes to discover Bluetooth devices, but that it also increased the number of devices found in a given inquiry. Fig. 5 shows a normalized plot of the number of devices found in a one second search. Regardless of the number of devices, each of our schemes consistently results in 1.5 to 3 times more devices being found. Similar data were gathered for longer search lengths.

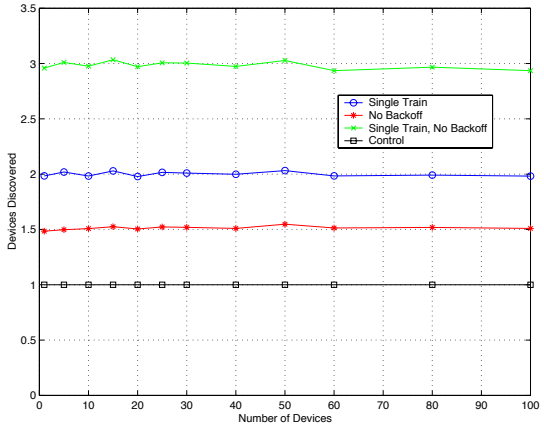

Fig. 5 - Number of Devices Found in a 1 Second Search (Normalized)

We know that we can improve connection setup times and also increase the relative number of devices found in a given inquiry. It is important to look at how our suggested changes affect collisions that can occur when two devices in INQUIRY SCAN transmit their FHS response packets at exactly the same time and on the same frequency.

Fig. 6 shows the average rate of collision for each of the test cases. The rate of collision is expressed as the fraction of inquiry packets sent that result in a collision. As expected, the control case has the fewest collisions. As in Fig. 4 the Single Train and No Backoff cases offer different qualities of performance that depend on the density of the Bluetooth environment. The combined case offers the worst performance, which was expected. However, this is still very few collisions, less than 0.03%

of all inquiry packets will result in collisions with this scheme.

Because the random backoff case had such good performance improvement, we decided to explore this case further. It would probably not be wise to totally remove the random backoff since two devices could still conceivably have closely matched clock values that would result in very high numbers of collisions. Fig. 7 shows the results of varying the random backoff from 4 to 1024 timeslots (each timeslot is 625μs) with 100 slaves present.

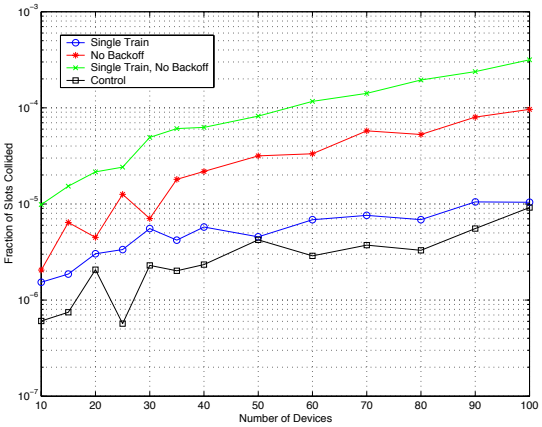

Fig. 6 – Average Rate of Collision

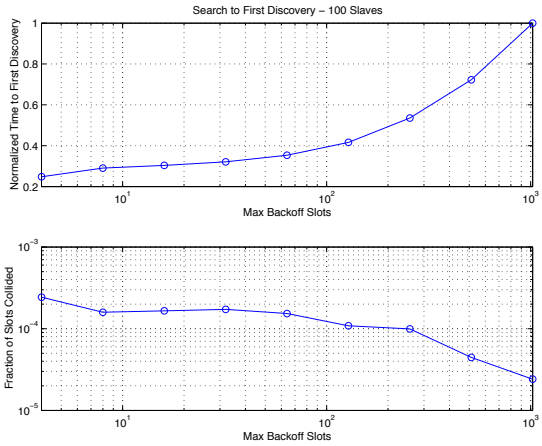

Fig. 7 - Varying the Random Backoff

V. DISCUSSION

Our empirical analysis and simulations demonstrate that Bluetooth is a viable candidate for use in highly mobile, ad hoc networks. This is particularly true in very dense environments - such as a large group of automobiles - where the application may not require communication with all devices, but rather just the first few to respond.

In a mobile environment, however, consuming even one-half to one second to setup a connection can impact the amount of data that can be transferred. To maximize data transfer we propose extending the Bluetooth baseband specification by adding a "mobility mode" that can be used in situations when devices are in motion. Our three proposed schemes - reducing random backoff, using a single inquiry train and a combined scheme - each offer significant performance benefits to connection setup without drastically increasing the number of failures due to collisions. In a highly mobile environment, one might not care that there are a few more collisions because it is not imperative to discover and communicate with every device. It becomes more like a best-effort model. The easiest scheme to implement, from a hardware and software standpoint, would be a modification of the random backoff time that a device implements in INQUIRY SCAN. We do not, however, suggest completely eliminating the backoff because there would still be a very small chance that two Bluetooth devices would have matching clock values and would always collide. A much smaller value, perhaps on the order of 0 to 16 timeslots (0 to 10ms), would likely prevent this from happening yet still offer superior connection setup times (about 75% less).

Decreasing or eliminating the random backoff does result in a larger number of collisions, however. An alternative to this would be to implement a single train sequence in the INQUIRE state. This offers a much lower improvement in connection times, but it does not increase the number of collisions from the control case.

This study of the Bluetooth discovery process has encouraged us to continue work along this path. As stated before, our test scenarios assumed that one device was constantly in the INQUIRY state while the other devices were always in INQUIRY SCAN. In the real world, this will not necessarily be the case, and this will impact device discovery. If a device cannot spend all of its time in either state, then this will naturally increase discovery times. Other researchers have developed efficient algorithms for switching between these two states to minimize connection setup times [5, 9], and we will incorporate such algorithms into future versions of our simulator. In addition, we plan incorporate channel models to reflect the presence of noise, fading, Doppler shift and other channel conditions.

VI. REFERENCES

[1] R. Nüsser and R. Pelz, "Bluetooth-based Wireless Connectivity in an Automotive Environment," *IEEE Vehicular Technology Conference*, Fall 2000, pp. 1935-1942.

[2] D. Groton and J.R. Schmidt, "Bluetooth-based Mobile Ad Hoc Networks: Opportunities and Challenges for a Telecommunications Operator," Vehicular Technology Conference, Spring 2001, pp. 1134-1138.

[3] H. Hedlund, *Bluetooth Baseband Specification, Version 1.1*, www.bluetooth.com.

[4] Axis OpenBT Stack, developer.axis.com/software/bluetooth/

[5] T. Salonidis, P. Bhagwat and L. Tassiulas, "Proximity Awareness and Fast Connection Establishment in

Bluetooth," *Mobile and Ad Hoc Networking and Computing*, 2000, pp. 141-142.

[6] G.F. Pedersen and P. Eggers, "Initial Investigations of the Bluetooth Link," *Vehicular Technology Conference*, Fall 2000, pp. 64-69.

[7] O. Kasten and M. Langheinrich, "First Experiences with Bluetooth in the *Smart-Its* Distributed Sensor Network," *Workshop on Ubiquitous Computing and Communications*, PACT 2001, October 2001.

[8] IBM Open-Source Bluetooth Simulator, BlueHoc. oss.software.ibm.com/bluehoc/

[9] F. Siegemund and M. Rohs, "Rendezvous Layer Protocols for Bluetooth-Enabled Smart Devices," *International Conference on Architecture of Computing Systems*, October 2001.