Validation of Simulations of Bluetooth's Frequency Hopping Spread Spectrum Technique

Article · January 2004

CITATIONS
6

READS
175

3 authors, including:

Lennart E. Isaksson
Ericsson
20 PUBLICATIONS **119** CITATIONS
SEE PROFILE

Markus Fiedler
Blekinge Institute of Technology
201 PUBLICATIONS **2,517** CITATIONS
SEE PROFILE

Validation of Simulations of Bluetooth's Frequency Hopping Spread Spectrum Technique

Lennart Isaksson, Markus Fiedler and Arne A. Nilsson

Blekinge Institute of Technology, School of Engineering,
Campus Gräsvik, SE-371 79 Karlskrona, Sweden.
email: lennart.isaksson@bth.se, markus.fiedler@bth.se and arne.nilsson@bth.se

Keywords: Inquiry Procedure, Page Procedure, Bluetooth, Validation, Frequency Hopping Spread Spectrum.

Abstract

The need for performance studies avoiding expensive hardware installations and the lack of analytical models motivates the need of trustworthy simulation models for the cable-replacement technology Bluetooth. In this paper we show a detailed description of the Bluetooth technique and validation of different simulation models used today. To do this, a two-step approach for validation and credibility is necessary. First, a comprehensive empirical measurement was performed to establish a "status quo" of the behavior of today's Bluetooth technology. Second, a validation between empirical and simulation model is conducted to ensure the comparability. To accomplish these objectives we developed and validated a simulation model with frequency hopping algorithms and the whole state machine including collision detection. The results in form of histograms of different Bluetooth-specific durations indicate problems in case of already established simulation programs. So, finally, some changes and suggestions how to correct these defects are given.

INTRODUCTION

Bluetooth™ was designed for cable replacements and ad-hoc networking in form of wireless transmission with comparably low bit rate over short distances. In order to properly design Bluetooth networks and investigate the effects of new ideas, we need to carry out performance studies. However, using hardware in a real-life simulation is in general more costly and time-consuming than using a simulation model. In some cases, a simulation is the only way to verify and test certain situations or the only way to investigate the question of interest. But then, the question arises whether such a model is trustworthy, i.e. whether its behavior coincides with the corresponding standards.

"*If a model is not valid, then any conclusions derived from the model will be of doubtful value.*" [1], p. 298.

Thus, one issue of concern arises when evaluating performance data derived from a simulation model: Can the results from the simulation be trusted? Are they fairly accurate? One way to find out is to validate results from the simulation model with empirical measurements. Another way is to validate the code of open-source simulation with the specification.

When comparing measurement results with those generated by well-established simulation software, we got indications that something is not as it should be. Different shapes on our histograms of Bluetooth-specific durations gave us a hint that we should investigate even deeper how *Frequency Hopping Spread Spectrum* (FHSS) is implemented in Bluetooth simulation models.

Bluetooth has a set of interesting possibilities. One is the ability to send and receive data and/or speech with hardly any disturbances with help from the FHSS technique. Others are the abilities to have a secure channel based on authorization, authentication or encryption, or to provide a particular service, *Service Discovery Protocol* (SDP), connected to the product and at the same time being hidden to other Bluetooth devices. Furthermore, Bluetooth has the potential of creating hardly disturbed ad-hoc networks in an industrial environment thanks to the FHSS.

The FHSS technique avoids disturbance using a pseudo-random jump between frequencies over a wide bandwidth spectrum of 2.402–2.480 GHz or 79 frequencies (23 in some countries).

The FHSS is based on two steps. First, the selection box derives how to calculate the next frequency depending on the current (sub-) state. Second, the state machine derives which sequence to use between the (sub-) state. The first step is easy to validate because of the sample data in the specification. The second step is a lot harder to validate. In this case, one is pointed to [2, 3] to interpret the text into a state machine.

The state machine plays an important role in the process of creating an ad-hoc network. A connection process always follows a predetermined pattern: standby, inquiry, page and connected, see Figures 1 and 2. The inquiry procedure, initiated by the master device, could be explained with the question: *Is anybody out there?* This question is to be answered by any slave device in inquiry scan substate. The page procedure, also initiated by the master device, could be explained with the question: *Are you there?* This question is dedicated towards a particular slave device being in page scan substate. Also, the slave takes over the master's clock in order to yield a common timebase within the piconet. Finally, a connection is made between the master and that slave device.

In this paper the following issues for the Bluetooth FHSS are considered: The algorithm of the selection box are validated against the specification p. 962–974 [2], and the state machine of the inquiry, page and connection procedures are compared to different simulations.

The paper is organized as follows: Section 1 describes related works and research within Bluetooth. Section 2 outlines Bluetooth basics relevant to this work like inquiry, page and connection procedures and frequency hopping. In Section 3, the empirically measured data is presented together with results from our simulation program. Section 4 describes our simulation model for the inquiry procedure. Section 5 describes our extended simulation model including page and connection. Section 6 provides an analysis of and workarounds for different simulation programs. Finally, in Section 7, conclusions and outlook are presented.

RELATED WORK

Welsh *et al.* [4] carried out an empirical analysis of the connection time between one to four Bluetooth devices. Further, they suggested three possible changes to the Bluetooth specification, how to decrease the connection time by simulating the connection in the simulation program ns-2 [5] with the optional Bluehoc package [6]. This work was done to see whether Bluetooth units were suitable for moving objects regarding the time necessary to connect between two Bluetooth units.

Murphy *et al.* [7] focused on short-term ad hoc connections between moving vehicles. Other areas included are range and throughput of the capacity between two Bluetooth devices. When investigating some variables in the baseband layer, they had to emulate the behavior. They used a simulation program *Rice Bluetooth Baseband Inquiry Tester* (RIBBIT) [8] based on the open source code from *International Business Machines* (IBM) BlueHoc to simulate connection times.

Tae-Jin Lee *et al.* [9] conducted a performance evaluation with point-to-multipoint connections, including *Asynchronous Connection-Less* (ACL) and *Synchronous Connection Oriented* (SCO) connections in a piconet. When using the simulation they did use a different set of parameters compared to our simulations.

None of the works cited above have reported on the validation of their simulation models.

BLUETOOTH

In order to provide necessary basics for the validation, an introduction to how Bluetooth works is presented. At first, we give a thorough explanation of the inquiry, page and connection procedures. Finally, we describe how the selection box works.

Notations

The notation $Z_{a-b\ (,c)}$ refers to bit a to b (and bit c) are used out of the bit representation for Z.

In the following, the X input determines the phase according to (sub-) states. The inputs Y1 and Y2 switch between master-to-slave and slave-to-master transmission. The inputs A to F, derived from different parts of the Bluetooth address, determine the ordering within the (sub-) states.

The variable k_{offset} is used to toggle between the A_{train} and B_{train}, see Equation 3.

Time

Every Bluetooth device has a free running 28 bit clock and increments every 312.5 μs. This clock is referred as *Clock Native* (CLKN). A master device is responsible for the timebase in a piconet and this is referred as *Clock* (CLK). An offset used to calculate the difference between a slave and a piconet is referred to as *Master Clock Offset* (MCO). A master device estimates the time difference to a slave in the page procedure and this is referred as *Clock Estimated* (CLKE). An offset used to calculate the difference is referred to as *Slave Estimate Clock Offset* (SECO).

Inquiry, Page and Connection Procedure

Before one can use an ad-hoc network with Bluetooth devices, one or several connections must be established. This is done using an inquiry, page and finally a connection procedure, see Figures 1 and 2, to establish a connection between two or several Bluetooth devices.

Inquiry procedure

The master device is in inquiry substate when trying to send over an *Identity* or ID (ID1) packet (72 bits) to the slave device, see Figure 1. The slave device is in the inquiry scan substate. In that substate the slave device is listening, but only during a time window of at least 11.25 ms, see Table 3.

When the first ID packet has arrived, the slave device goes into a random backoff period of 0–1023 slots, used for collision avoidance.

The slave device wakes up from the random backoff period and goes into the inquiry scan for the second time. This time there is a time limit to consider (inqrespTO). When the second ID (ID2) packet has arrived, the slave device goes into inquiry response substate exactly one slot later (625 μs). If the ID packet has not arrived in time, a timer is triggered and the slave device goes back to state standby.

In inquiry response substate, the slave device sends an *Frequency Hop Synchronisation* (FHS) packet to the master device with information

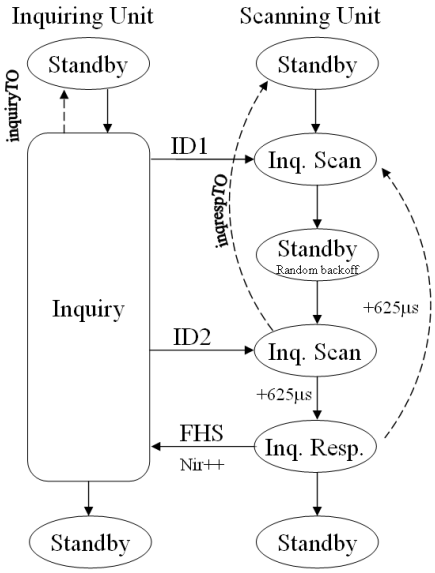


Figure 1: Inquiry procedure. The left column addresses the Master device, the right column addresses the Slave device.

about the slave, e.g. the *Non-significant Address Part* (NAP) of *Bluetooth Device Address* $(\text{BD\_ADDR})_{47-32}$, *Upper Address Part* (UAP) BD_ADDR_{31-24}, *Lower Address Part* (LAP) BD_ADDR_{23-0} and CLK_{27-2}. The Master's *Device Access Code* (DAC) is used for synchronization of its lower part of the time, CLK_{1-0}. The slave device only tries to send the FHS packet once, goes back to the first inquiry scan substate and starts all over again. At the same time the counter Nir is increased by one. The master device has a time limit to consider (inquiryTO). If the master device got the FHS packet from the slave device, it uses the slave's address by using DAC instead of the *General Inquiry Access Code* (GIAC). GIAC is only used during the inquiry substate because of the lack of knowledge of the potential slave of this point in time. DAC is used during the page substate dedicated to a known device.

Page procedure

When the master device has sufficient information (FHS packets from the slave device), it continues by sending an DAC (ID1) packet in the page substate using the slaves Bluetooth address

and a SECO, see Figure 2. The slave device recognizes its address when the DAC packet has arrived, and goes into the next substate which is the page response substate. At the same time, a part CLK_{16-12} of the clock is frozen. In this substate, there is also a time out period (pageRespTO). The value of Nprs, used by the slave device for synchronization, is increased by one. This is conducted every time $CKLN_1$ is set to zero. The master device has a time limit to consider (pageTO). In the page scan substate, the slave device is listening only during a time window of at least 11.25 ms, see Table 3.

In the page response slave substate, the device is trying to send over an ID (ID2) packet to the master device. When the master device has received the ID packet, it goes into the page master response substate. At the same time, a part of the clock and the parameter k_{offset} are frozen (see Equations 5 and 6).

In the page master response substate, the device is trying to send over an FHS packet to the slave device. When the slave device has sufficient information (FHS packets from the master device), it continues by sending an DAC (ID3) packet using the master's Bluetooth address and a MCO. The value of Nprm, used by the master device for synchronization, is increased by one. This is conducted every time $CKLN_1$ is set to zero. Both master and slave device have a time limit to consider (pageRespTO).

Connection procedure

Finally, after having sent the ID (ID3) packet the slave device goes into the connection state and waits for the POLL packet to arrive from the master device, see Figure 2. When the master device has received the ID (ID3) packet, it enters to the connection state and sends a POLL packet to the slave device. An acknowledgment from the slave device is conducted by sending a NULL packet. When the master device has received the packet, a connection has been established. In the connection state there is a time limit of 32 slots in case something goes wrong, e.g. POLL or NULL packets collide during the connection phase. Both master and slave device have a time limit to consider (newconnectionTO). In the connection state, a sliding technique is used to ensure the data transfer. If

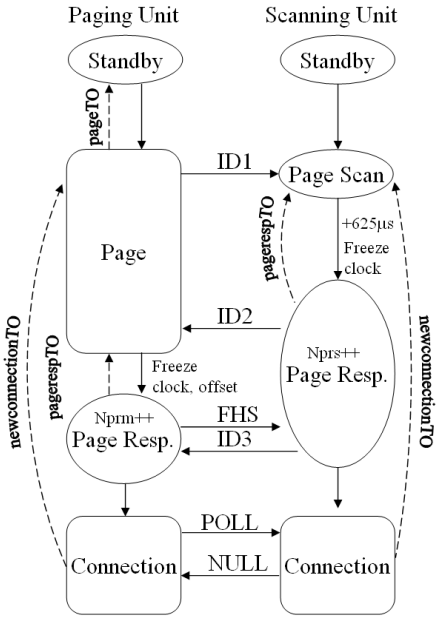


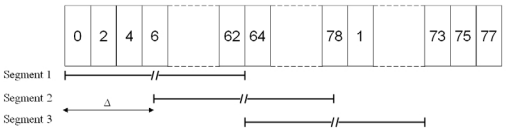
Figure 2: Page and Connection procedure. The left column addresses the Master device, the right column addresses the Slave device.

somebody tries to jam the communication on certain frequencies between the connected devices on several channels, the sliding technique assures the communication anyway: From Figure 3 column F, we see that the frequency changes every 0.04 s according to bit 7 in CLK_{27-7}.

Frequency Hopping

The Bluetooth device is using the 79-frequency system in most of Europe and the US. It only uses 32 hop frequencies for scanning and 32 hop frequencies for response, spanning 64 MHz, see Section 11.2 in [2] and Figure 3. This covers approximately 81 % of the frequencies, see Section 11.3 in [2].

Two states (Standby and Connection) and seven substates (inquiry, inquiry scan, inquiry response, page, page scan, page master response and page slave response) are depending on four variables to be able to calculate the next frequency. First there is a selection of the 23- or 79-frequency mode (23 or 79 number of frequencies used). The second is the address-variable BD_ADDR_{27-0}. The third is based on same clock CLKN, CLK or CLKE. The



	# of hops	Segment length	Δ	F
Europe/US	79	32	16	$16 \times \text{CLK}_{27\text{-}7} \bmod 79$
Japan/France/Spain	23	16	8	$8 \times \text{CLK}_{27\text{-}6} \bmod 23$

Figure 3: Top figure: Register bank, 79-frequency system. Bottom figure: Hop selection scheme for the connection state, 23 and 79-frequency system (23 or 79 frequencies used).

fourth is the number of times each so-called train of frequencies, A_{train} and B_{train}, should be used before alternating between them.

These trains are groups of frequencies to be visited in a predefined order. Equations 1 and 2 produces a sequence of numbers to form the contents of the A_{train} and B_{train} depending on k_{offset}. Phase 0 in the A_{train} includes of the frequencies $f(k-8)\ldots f(k)\ldots f(k+7)$ and the phase 0 in B_{train} includes the frequencies $f(k{+}8)\ldots f(k{+}15)$, $f(k-16)\ldots f(k-9)$, see Table 1. Corresponding entries for the B_{train} are obtained by adding 16 and taking the result modulo 32. For the A_{train}, phase 0 consists of the following sequence of numbers: 24, 25, 24, 25, 26, 27, 26, 27 $\cdots$ 6, 7, 6, 7. For the B_{train}, phase 0 consists of the following sequence of numbers: 8, 9, 8, 9, 10, 11, 10, 11 $\cdots$ 22, 23, 22, 23. Every train must be repeated $N_{\text{inquiry}} = 256$[1] times ($N_{\text{page}} = 128$) and takes 0.3125 ms/freq$\cdot$32 freq = 10 ms to be transmitted. In total, there is a period of $256 \cdot 10$ ms = 2.56 s for each type of train for the inquiry procedure and $128 \cdot 10$ ms = 1.28 s for each type of train for the page procedure. Consequently, only two whole phases or partly three phases are used before changing to another train depending on the starting point.

Equation 1 is valid for the *inquiry* (i) mode, while Equation 2 addresses the *page* (p) mode, see Figure 4. Equation 4 is valid both for *inquiry scan* and *inquiry response* (ir) modes. Equation 5 addresses the *page response master* (prm) mode. Equation 6 addresses the *page response slave* (prs) mode, see Figure 4. In some conditions the values have to be frozen (i.e. kept constant even thought the clock is ticking), which is indicated with an asterisk ($*$), see Equations 5 and 6. The counter N is increased by one each time a FHS packet is sent, see Equation 4 and each time CLKN_1 or CLKE_1 is set to zero, see Equations 5 and 6.

$$\text{Xi}^{(79)}_{4-0} = [\text{CLKN}_{16-12} + k_{\text{offset}} + (\text{CLKN}_{4-2,0} - \text{CLKN}_{16-12}) \bmod 16] \bmod 32 \quad (1)$$

$$\text{Xp}^{(79)}_{4-0} = [\text{CLKE}_{16-12} + k_{\text{offset}} + (\text{CLKE}_{4-2,0} - \text{CLKE}_{16-12}) \bmod 16] \bmod 32 \quad (2)$$

$$k_{\text{offset}} = \begin{cases} 24 & \text{for } \text{A}_{\text{train}} \\ 8 & \text{for } \text{B}_{\text{train}} \end{cases} \quad (3)$$

$$\text{Xir}^{(79)}_{4-0} = [\text{CLKN}_{16-12} + \text{N}] \bmod 32 \quad (4)$$

$$\text{Xprm}^{(79)}_{4-0} = [\text{CLKE}^{*}_{16-12} + k^{*}_{\text{offset}} + (\text{CLKE}^{*}_{4-2,0} - \text{CLKE}^{*}_{16-12}) \bmod 16 + \text{N}] \bmod 32 \quad (5)$$

$$\text{Xprs}^{(79)}_{4-0} = [\text{CLKN}^{*}_{16-12} + \text{N}] \bmod 32 \quad (6)$$

Inquiry, inquiry scan, and inquiry response substates, see Figure 1, use different algorithms to generate the frequency. In particular, for the inquiry scan and inquiry response algorithms, the input Y1 is toggled between 0x00000 for receiving mode (Inquiry scan substate) and 0x11111 for sending mode (Inquiry response substate). So, the sequence 24, 25, 24, 25, $\cdots$ becomes 48, 50, 09, 13 when using CLKE = 0x0000000 and ULAP = 0x00000000 (UAP_{3-0} and LAP_{23-0} put together).

Finally, after having performed several steps in the selection box, a subfrequency-to-frequency conversion is made. The proper way is to convert the derived frequency according to the algorithms in section 11.1 [2] and Figure 4, right-hand side. The subfrequency-to-frequency vector is created by first listing all even numbers starting with frequency 0 and ending with frequency 78. Finally,

[1]When validating the selection box against the specification, one should always use $N_{\text{inquiry}} = 128$.

Table 1: Sequence from Equation 1 and Equation 2 for the A_{train}

Phase	0	···	7	8	···	14	15
0	24	···	31	0	···	6	7
1	8						
⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋮
8	⋮	⋮	15	⋮	⋮	⋮	⋮
9	⋮	⋮	⋮	16	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮
15	⋮	⋮	⋮	⋮	⋮	22	⋮
16	⋮	⋮	⋮	⋮	⋮	⋮	23
17	24	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋮
24	⋮	⋮	31	⋮	⋮	⋮	⋮
25	⋮	⋮	⋮	0	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮
31	⋮	⋮	⋮	⋮	⋮	6	⋮

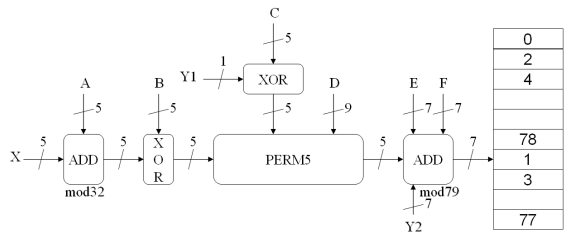


Figure 4: Selection box, 79-frequency system.

all odd numbers are added to the vector starting with frequency 1 and ending with frequency 77. All seven substates inquiry, inquiry scan, inquiry response, page, page scan, page response slave and page response master and connection state (five tables of data) have been validated and verified against the specification, see section 2 in the specification [2]. For the inquiry response substate, a table in the specification is missing. This should match the inquiry and inquiry response submode. One way to validate and verify the proper frequency and channel is to look at the inquiry table. The slot for receive is matched towards the one for inquiry response and the slot for transmit is matched towards the one for inquiry scan, respectively.

EMPIRICAL ANALYSIS OF THE INQUIRY PROCEDURE

In this section, a presentation of the measurement setup is given. The empirical results are based on real-time measurement of inquiry and connection activity for device A. The Bluetooth devices were located closely together (0.1 m) during all measurements.

Measurement Setup

One program was implemented to handle two types of modes, master and slave. Each command and event packet from the Bluetooth device was logged with a time stamp resolution of 1 ms, see the example in Table 2. Two Bluetooth devices of type Ericsson Bluetooth Starter Kit (LPY 111 243) [10] were connected to a computer.

In master mode, the program starts with a uniformly distributed random time between 0 and 10.24 s to minimize correlation between subsequent samples. First, the master device was initialized with the *Host Controller Interface* (HCI) command *Reset* before making any attempts to find any Bluetooth devices. Second, the HCI command *Inquiry* was used.

In slave mode, the program starts with a HCI command *Reset*, *Set Event Filter* with *Inquiry Scan enabled*, and stays in this mode during the measurements. Second, the master device waits for an *Inquiry Result* event. Third, the HCI command *Create_Connection* is issued. Fourth, the master device waits for a *Connection Complete* event. Fifth, the HCI command *Disconnect* is performed. Finally, the master device waits for a *Disconnection Complete* event. This sequence of commands and events is conducted over a period of 500 samples each time.

Every time stamp between the *Inquiry* command and the successive commands and events are saved. Looking at the example in Table 2, the command for inquiry at time 12:40:46.037 and the *Inquiry Result* event at time 12:40:50.333 have a time difference of 4.296 s.

A possible pitfall is related to the fact that looking for the same device without any time delay between two samples could endanger the measurements, because both can still know each other

Table 2: Command packet sent and Event packet received from Master device

Time (hh:mm:ss.ms)	Command / Event
12:40:42.902	3092-Delay
12:40:42.922	01030C00-Reset
12:40:46.037	01010405338B9E3000-Inquiry
12:40:50.333	04020F011F8C1637800001000000000003351
12:40:50.353	0105040D1F8C163780000800000033D100-Con
12:40:54.328	04030B0001001F8C163780000100
12:40:54.579	01060403010013-Disconnect
12:40:54.789	04050400010016

which might imply a short inquiry time. Another potential problem is that other Bluetooth devices may happen to be located close to the measurement equipment. The program is also taking care of this problem. In this situation the "alien" Bluetooth address is displayed in an alert window.

Bluetooth Commands

Several Bluetooth commands were used during the measurements.

- *Inquiry* (Section 4.5.1 [2])
  The HCI command *Inquiry* has been initialized with the following parameters, *GIAC* (0x9E8B33), inquiry_length (N = 30) and *Num_Responses* (unlimited number of responses).

- *Create_Connection* (Section 4.5.5 [2])
  The HCI command *Create_Connection* has been initialized with the following parameters: BD_ADDR, *Packet_Type* (0x0008), *Page_ Scan_Repetition_Mode* (0x00), *Page_Scan_Mode* (0x00), *Clock_Offset* (extracted from the *Inquiry Result* event with bit 15 set to 1, valid clock offset) and *Allow_Role_Switch* (0x00).

- *Disconnect* (Section 4.5.6 [2])
  The HCI command *Disconnect* has been initialized with the following parameters, *Connection_Handle* (0x0001) and *Reason, other end terminated connection: User ended connection* (0x13).

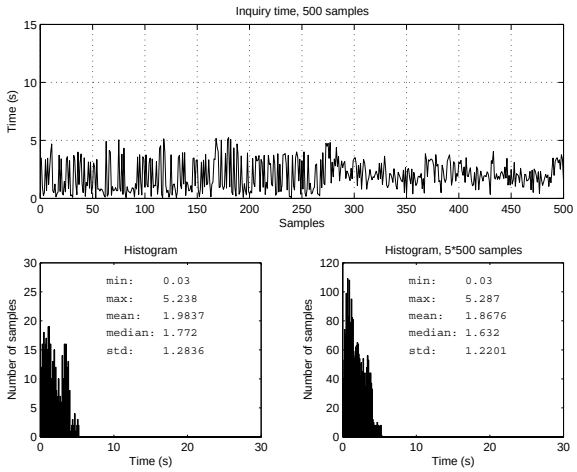


Figure 5: Empirical data from the device under test with default settings. Top and left: Plot and histogram of 500 samples. Right: Histogram of 2500 different samples.

- *Read_Inquiry_Scan_Activity* (Sec. 4.7.21 [2])
  The empirical tests were partly based on the two default values from the command *Read_Inquiry_Scan_Activity* with the subcommand *Inquiry_Scan_Interval* (Default: N = 0x0800, i.e. a time of 1.28 s) and *Inquiry_Scan_Window* (Default: N = 0x0012, i.e. a time of 11.25 ms).

Limitations

The use of 500 batches was a compromise due to limitations imposed by the hardware: After about 1,000 samples (i.e. 4 hours of operations), the device ceased to work. For sake of better comparability, the mean inquiry times of 2,500 measurements from 5 independent batches are given. Due to these limitations, no confidence intervals could be established for the measurements.

SIMULATIONS OF INQUIRY

In order to get an idea of the statistical quality of the mean inquiry times X_i obtains from 250 measurements each, confidence intervals are constructed. As the lag-1 correlation estimation of X_i is slightly negative (correlation coefficient of -0.0056), we use standard confidence analysis. The

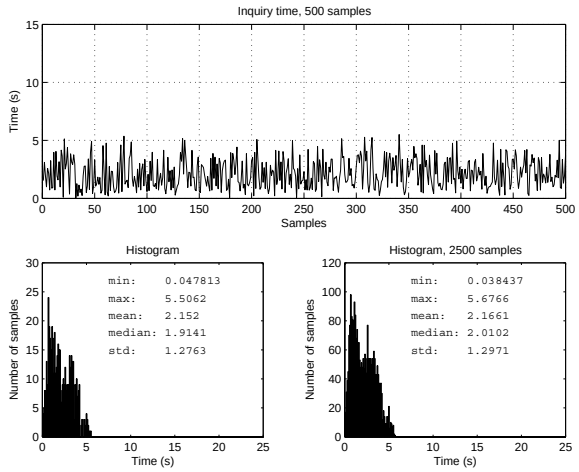


Figure 6: The statistical data from our own simulator model is only based on the inquiry time. Top and left: Plot and histogram of 500 samples. Right: Histogram of 2500 different samples.

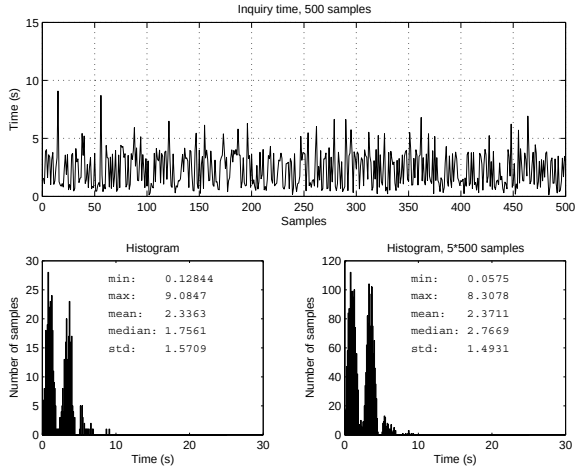


Figure 7: Statistical data from simulator RIBBIT version 1.0 [4], with default settings (based on Bluehoc). Top and left: Plot and histogram of 500 samples. Right: Histogram of 2500 different samples.

confidence interval is given by

$$\bar{X} \pm t_\alpha(n-1)\frac{s}{\sqrt{n}} \tag{7}$$

where $\bar{X} = 2.1579$ is the average of $n = 60$ mean inquiry times X_i, s $= 0.0810$ is the standard deviation and $t_\alpha(n-1) \simeq 2.00$ for a confidence level of $1-\alpha = 95$ %. This results in a half size of the confidence interval of 0.0209, which is less than 1 % of the mean.

Is there a way to check if the system behaves, of inquiry times, as it should? The answer is yes. The shape of the histogram is like a fingerprint. It can show if the specification is implemented correctly or not. Composing Figures 5 and 6, we observe similar shapes of the histograms based on measurements and our own, validated simulator. The simulation software RIBBIT [8], however, produced different shapes, see Figure 7. Additionally, there is a small deviation of the mean value between hardware device and the simulation reference, see Table 4.

Table 3: Parameters used in our simulation model

	Parameter	Value
Inquiry	inquiryTO	20 s
	inqrespTO	2.56 s
	T inquiry scan	1.28 s
	Tw inquiry scan	11.25 ms
	Random backoff	0 – 640 ms
	N inquiry	256
Page	pageTO	2.56 s
	pagerespTO	5 ms
	T page scan	2.56 s
	Tw page scan	11.25 ms
	N page	128
Connection	newconnectionTO	20 ms

The hardware meets the expectations regarding the specifications. Some available simulation programs like BlueHoc [6] and RIBBIT [8] using them in an unmodified way tend to be misleading, see Figure 7 and Table 4. This could result in doubtful inquiry, page and connection times.

Table 4: Mean inquiry time

Type	Mean	# samples	Figure
Simulation	2.1579 $\pm$0.0209	$60 \cdot 250$	(ref.)
Hardware dev.	1.8676	2,500	5
RIBBIT	2.3711	2,500	7
Simulation	2.1661	2,500	6

The mean inquiry time for the hardware is to be found close, but not within the confidence interval of the simulations.

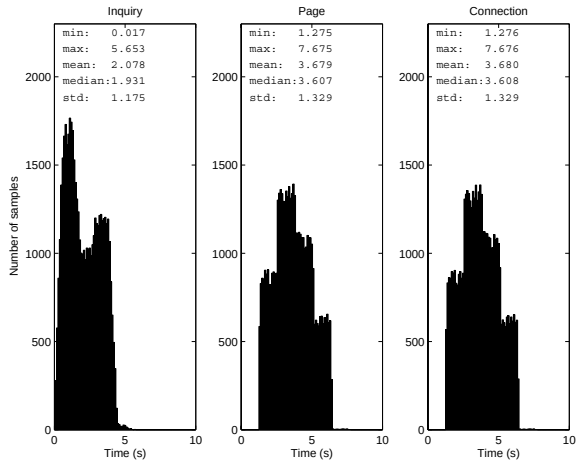


Figure 8: The statistical data from our own simulator model is based on the inquiry, page and connection time. Left figure: Inquiry time. Middle figure: Total time up to Page completed. Right figure: Total time up to Connection completed. Histograms of 50,000 samples are shown in all three figures.

TIME SIMULATIONS

In this section we present a complete measurement of the inquiry, page and the connection time. The parameters used for the simulations are shown in Table 3.

Table 5: Mean value of time for Inquiry, Page and Connection using 50,000 samples

Standby-Inquiry	Inquiry-Page	Page-Connection
2.0779 s	1.6012 s	0.0012 s
56%	43%	1%

Finally, a larger number of sample data from our simulation model has been extracted to show all three steps inquiry, page and connection towards a complete connection of a Bluetooth device with one master and one slave, see Figure 8 and Table 5. Our simulation model is capable of handling collision detection. This is important because is has an impact on the simulation results if more than one master and one slave are used. This is not implemented in the Bluehoc package. Our simulation model is also capable of handle the 23-frequency system.

SOFTWARE ANALYSIS

In this section a number of bugs in popular simulation tools are shown together with proposed workarounds. Two programs were considered during the analysis, RIBBIT version 1.0 [4] based on Bluehoc version 2.0 [6], and our own simulation program.

The parameters Xi and Xp

At first glance, this implementation seems to be executed correctly. This implementation is used both in Bluehoc version 2.0 [6] and RIBBIT version 1.0 [4]. However the result of the modulus operator is not equal to the remainder operator in case of negative x value; see Equation 8.

$$x \bmod y \left\{ \begin{array}{c} = \\ \neq \end{array} \right\} x \,\%\, y,\; x \left\{ \begin{array}{c} \geq \\ < \end{array} \right\} 0 \qquad (8)$$

In the specification, the modulus operator should be implemented, but the remainder operator is used instead. One way of taking care of this problem is using a filter in front of remainder operator to eliminate the sign.

Bluehoc and RIBBIT

The original code is as follows:

```
Xp_or_i = (CLK_12to16 + k_off +
((CLK_0_2to4 - CLK_12to16) % 16)) % 32
```

As long as the inner part of the equation is positive and small (with no sign), there is no problem. The problem emerges when the inner part becomes negative. Which is due to the definition of the variable as (int). The variable should be set to an (unsigned int), or the masking technique described above should be used to eliminate the sign. The correct implementation using the masking technique should be as follows:

```
Xp_or_i = ((CLK_12to16 + k_off +
((CLK_0_2to4 - CLK_12to16) & 0x1f % 16 )
% 32) & 0x1f
```

Bluehoc

Some effort has been made in this program, to compensate the problem (1) with the following formula:

```
Xp_or_i = ((CLK_0_2to4 < CLK_12to16 ?
((Xp_or_i + 16) % 32 : Xp_or_i
```

This algorithm should not be included in the

model. It's not a part of the specification.

Impact of the bit Y1

The reason for using Y1 is to toggle between receiving and sending mode. The specification [2] states in section 11.2.3: $C_i \oplus Y1$ for $i = 0 \ldots 4$, which means that each bit of C should be compared to Y1 that is actually only one bit. Instead the variable C is used together with Y1, which leads to a wrong result. Both programs are using a direct operator xor on each of the variables. The following expression is used in Bluehoc [6]:

```
ip1_EXOR2 = CLK_1
```

The following expression is used in RIBBIT version 1.0 [4]:

```
ctrlY1 = clk1
```

The correct implementation should be for Bluehoc:

```
ip1_EXOR2 = 31 * CLK_1
```

and for RIBBIT

```
ctrlY1 = 31 * clk1
```

Inquiry Scan / Inquiry Response

One option during the inquiry process is to count the number of FHS packets received from the slave. In this case, the master choose to wait for more than one FHS packets. This value is missing in BluHoc and RIBBIT. Both programs BlueHoc and RIBBIT are using the following implementations:

```
CLKN_16-12
```

The proper implementation should be, according to the specification [2]:

```
[CLKN_16-12 + N] mod 32
```

Frequency Conversion

Finally, after having conducted several steps in the selection box, both programs Bluehoc and RIBBIT forget to do the frequency conversion see right-hand side of Figure 4. The proper way is to convert the derived subfrequency to the proper frequency according to the algorithms in Section 11.1 of [2].

CONCLUSIONS AND OUTLOOK

This paper present a validation of simulation models for the inquiry, page and connection procedure of Bluetooth. For that purpose, the specification was investigated in detail and compiled into a validated simulation program used for comparison. Results from this program were compared to our measurements. Mean and histogram of the inquiry time matched well between the simulation program and the hardware, other popular simulation tools (BlueHoc and RIBBIT) displayed different values. Potential sources of problem we identified and workarounds were described. This investigation once again shows the importance of verification of program code and validation of simulation models before treating their output as credible and using it for future research.

REFERENCES

[1] A.M. Law and W.D. Kelton. *Simulation Modeling & Analysis, Second Edition*. McGraw-Hill, Inc., 1991.

[2] Specification of the Bluetooth System, Core, Version 1.1. http://www.bluetooth.com.

[3] Haartsen J.C. Bluetooth-ad-hoc networking in an uncoordinated environment. In *Proceedings of the 2001 Acoustics, Speech, and Signal Processing*, pages 2029–2032, 2001.

[4] Welsh E., Murphy P., and Frantz J.P. Improving connection time for bluetooth devices in mobile environments. In *Proceedings of the International Conference on Fundamentals of Electronics, Communications and Computer Sciences*, 2002.

[5] The Network Simulator, ns-2. http://www.isi .edu/nsnam/ns.

[6] BlueHoc, Open source Bluetooth Simulator, Bluehoc 2.0. http://www-124.ibm.com/bluehoc.

[7] Murphy P., Welsh E., and Frantz J.P. Using bluetooth for short-term ad hoc connections between moving vehicles: A feasibility study. In *Proceedings of the 55th IEEE Vehicular Technology Conference*, pages 414–418, 2002.

[8] RIBBIT, 1.0. http://koala.ece.rice.edu/bluetooth /ribbit.

[9] Tae-Jin Lee, Kyunghun Jang, Hyunsook Kang, and Jonghun Park. Model and performance evaluation of a piconet for point-to-multipoint communications in bluetooth. In *Proceedings of the 53th IEEE Vehicular Technology Conference*, pages 1144–1148, 2001.

[10] Ericsson Microelectronics. http://www.ericsson .com/microe.