Formal Verification of Device Discovery Mechanism using UPPAAL

Shivangi Arry
Punjabi University Regional Centre for I.T. and Mgmt., Mohali

Amardeep Kaur
Assistant Professor in (CS)
Punjabi University Regional Centre for I.T. and Mgmt., Mohali

ABSTRACT
Wireless communication among the mobile devices is the growing area of research. Ad hoc networks are created among them for exchanging information over the small distances. The distance among the devices uniquely determines the topology of these ad hoc networks. Bluetooth is a low-cost and low-power wireless technology used to form short-range wireless ad hoc networks. It is based on frequency hopping technique. In order to guarantee the accurate functioning and to verify the connectivity of the devices, there are numerous tools and techniques exits. Formal verification is one of the technique used to validate and verify the correctness of any protocol. Using the verification tool UPPAAL a Bluetooth ad hoc system is modeled to formally verify the device discovery mechanism. In the tool, an ad hoc network consisting of Bluetooth transmitter and receiver is designed using frequency hopping technique. The transmitter and receiver verified the properties in the tool. These properties are for checking the connectivity of transmitter and receiver and reception of data among them. The connectivity of transmitter and receiver is confirmed by verifying the receivers reply to the transmitter and data connectivity is verified by checking the energy level of receiver. From the simulation, it is found that the high energy level of receiver depicts the acceptance of data and the zero energy level depicts the non-acceptance of data. These results were verified using the verifier component of the tool and proved to be satisfied.

Keywords
Bluetooth Ad hoc Networks, Spread spectrum, Bluetooth Device Discovery, Formal verification, UPPAAL model checker.

1. INTRODUCTION
Wireless networks have changed the way various devices are interconnected. There are various features that provide ease to communicate in the wireless medium like, devices can move freely, communication between these does not require any central access point, reduction in complexity for establishing the connections and many more. Among most promising wireless technologies that can be effectively used by the wireless mobile devices, Bluetooth technology is one of them.

Bluetooth technology, that is widely used to create ad hoc connections, provides the short-range wireless communication. It is considered as a Wireless Personal Area Network (WPAN) system that is intended for cable replacement and short distance ad hoc connectivity. Its main goals include robustness, low power, low cost, and low complexity. Using Bluetooth technology devices like cell phones, computer, laptops, PDAs etc. can be easily connected for exchanging information. The Bluetooth device establishes communication with other devices after discovering each other in their neighboring environment. So, in order to communicate, they must first discover each other. The work under study covers the Bluetooth device discovery mechanism and uses the formal approach to verify it.

There are various methods for analyzing the systems, which include simulation, testing, and formal verification. Among all, simulation cannot cover the complicated situations in the real networks, and so some subtle bugs may not be found out. Moreover, the testing can be performed only after a protocol is implemented. However, the formal verification verifies some properties of the specifications and finds the latent bugs in the design of the protocols. Formal Verification Techniques are used to provide a systematic way to assess the correctness of protocols, process, and systems. Compared to the other methods, the main difference lies in the fact that instead of only examining a limited area of the operational space of the system, they are used to examine the complete state space of possible operations. It implies that all possible combinations of inputs and actions can be taken into account and, therefore, all possible outputs can be derived.

The paper has been organized as follows. Sections 2 presents the background and cover the topics: Bluetooth ad hoc network, formal verification, spread spectrum, frequency hopping. Section 3 talked about the model checker UPPAAL. Modeling specifications of the system and results of formal verification is described in the section 4. Section 5 consists of the conclusion and future scope of the work. Finally, the section 6 gives the acknowledgements.

2. BACKGROUND
An ad hoc network is a network with temporary plug-in connections, in which the network devices are part of the network only for the duration of a communications session. Ad hoc networks are infrastructure less networks that do not require any central information hub for making connectivity. Connections between the devices are set up based on wireless media like, two PCs equipped with wireless adapter cards can set up an independent network whenever they are within range of one another. There are various wireless technologies available to create ad hoc network like Induction Wireless, Infrared Wireless, Ultra Wideband, ZigBee, IEEE 802.11b and Bluetooth [1]. Among all technologies, Bluetooth is the most promising wireless technology. It is a Personal Area Network technology, which aimed in particular at low-power devices communicating over short distances. It replaces the cable connectivity with wireless communication and promotes ad hoc networking [12]. Bluetooth technology is increasing at a rapid rate now days, many devices such as mobile phones, PDA's and laptop computers have Bluetooth built into them.

2.1 Bluetooth

Bluetooth technology is designed for wireless communication between wide varieties of different Bluetooth enabled devices [30]. It reduces the need of cable connectivity. Bluetooth enabled devices transmit and receive in an unlicensed ISM (Industrial, Scientific, and Medical) frequency band of 2.4 GHz that is available globally [12]. The ISM band is free for use by other wireless devices and it creates a potential problem of interference. Thus, the interference immunity becomes a very important issue for Bluetooth. In order to avoid interference, Bluetooth uses a frequency-hopping spread-spectrum (FHSS) communication, which transmits data over different frequencies at different time intervals and devices alternate rapidly among 79 available frequencies in a pseudo-random fashion [15].

2.2 Spread spectrum technique

Spread Spectrum is a modulation method applied to digitally modulated signals that increases the transmit signal bandwidth to a value much larger than is needed to transmit the underlying information bits. Spread Spectrum, originally developed for the military guidance and intelligence requirements [5], "break a call into multiple parts, and uses constantly changing radio frequencies to improve data transfer reliability, reduce sensitivity to interference and jamming, and deny eavesdropping" [18].

It provide many benefits like, transmission security, resistance to interference from other radio sources, redundancy, resistance to multipath and fading effects, etc. As a result, Spread Spectrum systems can coexist with other radio systems, without being disturbed by their presence and without disturbing their activity. There are two forms of spread-spectrum techniques: frequency hopping spread spectrum (FH) and direct sequence spread spectrum (DS) [18].

2.2.1 Frequency Hopping Spread Spectrum (FHSS)

FHSS is a method of transmitting radio signals by rapidly switching a carrier among many frequency channels, using a pseudorandom sequence known to both transmitter and receiver [18].The pseudorandom sequence or the hopping sequence is also known as FH code. If a long enough pseudo-random FH-code is used then unintended users are not able to determine the hopping pattern and use that information to intercept the transmission. It supports the anti-jamming capability and protection against undesired interception. It is a very robust technology, with little influence from noises, reflections, other radio stations, or other environment factors [19]. This technique is more suited to relatively low-data rate, low-power systems [13].

2.2.2 Direct Sequence Spread Spectrum (DSSS)

DS involves transmitting at a much higher rate than the data-rate, and convoluting the data with pseudo random spreading (chip) sequence [21]. Both the transmitter and receiver use the same pseudo random sequence to spread and de-spread the information signal [34]. At the receiver, the signal can be decoded by using correlation techniques, comparing the incoming signal with the spreading codes. DSSS can be used to increase the security of the transmission and also overcome jamming, multi-path and signal fading problems.

DSSS has the advantage of providing higher capacities than FHSS, but it is a very sensitive technology, influenced by many environment factors mainly reflections [19]. DSSS is more complex to implement than FHSS and is not suitable to low-power systems due to the high data rates involved [13].

2.3 Device Discovery in Bluetooth

Devices in mobile ad hoc environments initially have no information about their surrounding environment. There is no centralized instance to query about the environment. Therefore, there must exist a protocol that provides means for detecting devices and enables devices to set up a connection, Bluetooth uses the Baseband protocol for this task. Two procedures are used in the device discovery procedure; inquiry and page [18]. For communicating, Bluetooth devices organize themselves into small networks called piconets, comprising one master and up to seven slave devices, in which the frequency hopping sequences are synchronized and controlled by the master. The master follows the two procedures to discover the device as follow [20, 15]:

- Inquiry: In the inquiry phase, the master invites other devices to join itself to form a piconet. Essentially, the master solicits responses from other devices. Devices willing to join a piconet respond to the master's solicitation. In the inquiry phase, the master solicits responses by transmitting a standard packet, called ID (identifier) packets, on different hop channels (i.e., on different frequencies), and listens for response packets, called FHS (frequency hopping sequence) packets, from potential slave devices. The sender of an FHS packet puts its own address and clock values in the packet. When the master receives an FHS packet, it knows what device is willing to join the master to form a piconet. However, mere response of a device with an FHS packet does not form a piconet. After receiving an FHS packet, the master executes the paging step.
- Page: After receiving the address and clock values of a (potential) slave device, the master communicate with the device via the paging process. During the paging process, the master establishes a connection with a slave [10]. The paging procedure takes a very short time (at most 20 ms) while the inquiry procedure might cause a significant delay (up to a few seconds on average) [14].

The device discovery is performed by each node randomly entering into an inquiry mode. Inquiry nodes select a repeated pattern of 32 frequencies and send signal on selected frequency in given spot. Inquiry scan nodes also select a frequency at random in each spot and listen to the transmission at the selected frequency. The discovery (and establishment of master-slave relationship) occurs when both sender and receiver nodes are at the same frequency.

2.4 Formal Verification

In order to increase the quality of the protocols, designers of the protocol must validate the design of the protocol. There are several methods for validation like using a software model of protocol and performing it on virtual devices in simulated environment or doing live test on real hardware. These tests can be used to identify bugs but cannot exclude protocols blueprint errors [3]. Another method used to confirm the routing protocols is the Formal verification.

Formal verification is a systematic process that uses mathematical reasoning for the specification, development, and verification of computer–based software and hardware systems [2][6]. It provides the systematic way to access the correctness of protocol, process, and system. It checks whether a design satisfies some requirements (properties). There are three types of Formal verification techniques [3]:

- **Model checking:** It is the technique in which a system verifies certain properties by means of an exhaustive search of all possible states that a system could enter during its execution. SPIN, SMV model checker, and UPPAAL are the model checking tools.
- **Theorem proving:** It is the technique in which a system attempts to produce a formal proof, given a description of the system, a set of logical axioms, and a set of inference rules. Some of theorem proving tools are Isabelle, HOLY, PVS, ACL2.
- **Equivalence checking:** It formally checks the equivalence of two models, which are at different abstraction level. CVE toolset, and Synopsys are equivalence checking tools.

There are various tools and techniques used for formal modeling and verification of the ad hoc network routing protocols. It ranges from Petri nets, SPIN Model Checker, PROMELA, AVISPA, HLPSL, UPPAAL Model Checker, SDL and BAN logic [3].

3. UPPAAL

UPPAAL is a formal verification tool for modeling, simulation, and verification of real time system. Real time systems like communication protocol, multimedia application, and real time controller. It has been jointly developed by Department of Computer science, Uppsala University in Sweden and BRICS, at Aalborg University in Denmark[g]. It is an established model checker for networks of timed automata, used in particular for protocol verification [4]. The software is split into a graphical user interface (GUI) implemented in JAVA and verification engine in C++ both available for most common platform [16]. For installing the tool, the system must be configured properly with Java version 6 (e.g. J2SE Java Runtime Environment) or newer. The software runs under the Windows, Linux, and Mac OS X [7]. UPPAAL is freely available for non-commercial applications in academic and for private persons at http://www.uppaal.com/. The usage of UPPAAL can be obtained from the tutorials [9].

UPPAAL is designed for the formal verification of system that can be represented as networks of timed automata extended with elementary and structured data types and channel synchronization. Time automata are finite state machine with clocks [16]. These clocks handle time in UPPAAL. Time will progress globally at the same pace for the whole system [17]. As clock variables are used to represent time and permit measuring of time progress. Thus, the behavior of timed automata can be restricted by clock constraints [16].

UPPAAL uses client-server architecture and split the tool into a graphical user interface and a model-checking engine. The user interface, or client, is implemented in Java and the engine, or server, is compiled for different platforms (Linux, Windows, Solaris) [9]. The GUI interface of UPPAAL has three main tool components: the system editor, the simulator, and the verifier. These tools are integrated on one common interface and work on same internal system model. This makes the exchange of information easier, e.g. loading diagnostic trace generated by the verifier into the simulator for further inspection [8].

4. MODELING SPECIFICATION AND VERIFICATION RESULTS

In this study, a network of timed automata has been created using UPAAL. For this, three templates has been modeled to represent sender, receiver and to synchronize the frequency of sender and receiver. The template modeled for sender is shown in Figure 1.

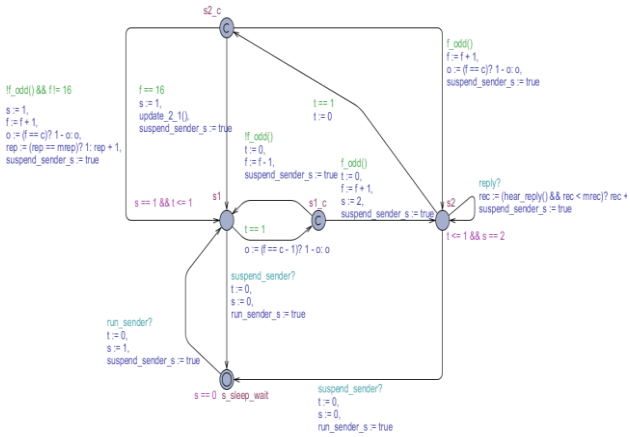

Fig. 1: Sender Template in Editor

There are three states defined for sender: sleep, sending and listening. The sender synchronizes with the receiver over the channels: run_sender, suspend_sender. Sender defines its local clock with variable ‘clock t’. There are four anonymous state defined to show the sender changing states. Sender initially starts with sleep mode and after synchronization with the receiver, it shifts to the sending mode. Sender checks its status of current frequency with the defined frequency module each time before transmitting a signal and changes its state.

The template for the receiver is shown in Figure 2. The receiver is defined with the states: sleep, scan, respond, and reply. Each state is given with time slots and one time slot is equal to 0.3125ms, specified in Table 1.

Table 1: Showing time slots with respect to different modes

Modes	Time slots
Sleep	2012 time slots (628.75 ms)
Scan	36 time slots (11.25 ms)
Respond	2 time slots (0.625 ms)

Receiver will first synchronize with the frequency hopping over the channel run_frequency and suspend_frequency and then synchronize with the sender over the channels: run_sender, suspend_sender, in order to create links.

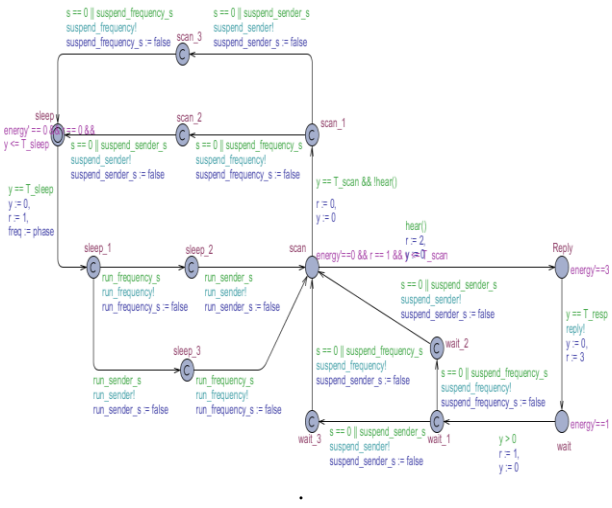

Fig. 2: Receiver Template in Editor

In the scanning state, the receiver synchronizes with the frequency with respect to phase. After sender and receiver synchronize over the channels, receiver will scan the incoming signal. While receiver is scanning the signal, it will receive the data depending upon its energy level.

Finally, a template showing the receivers frequency is created to synchronize the sender and receiver as depicted in Figure 3.

It is the receiver's frequency, which keep track of the frequency that the receiver will use next time to start a scan. Since the transmitter uses frequency-hopping spread-spectrum, the receiver device will synchronize its hops along with the sender's hop.

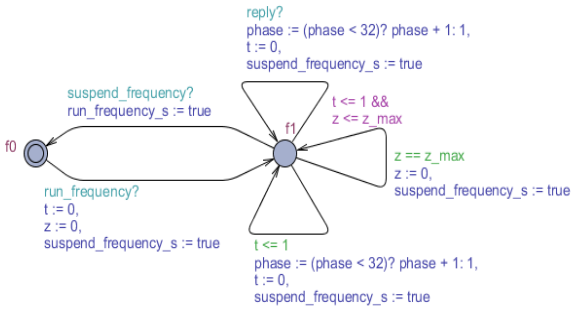

Fig. 3: Receiver-Frequency Template in Editor.

This will allow the receiver to communicate with the sender at the same frequency hopping at which the synchronization between them takes place. Sender and receiver will automatically form a connection when they come within the range of one another. When the receiver is in scanning state, it continuously listens to the current frequency. The frequency of the receiver changes with respect to its phase. Receiver, cycles through the 32 frequencies and changes frequently. Once the links are established between the sender and receiver, they start transmitting the messages.

4.1 Simulation

In the simulation, all-possible states of the system are implemented. Receiver synchronizes using the frequency hopping to set its hopping environment. Then it set up its path with sender by synchronizing over the channels. After synchronizing, sender shifts from sleep state to sending state and starts connection with receiver over the same frequency hopping. Now when receiver and sender connect, they will transmit signal and data over the channels. Each time receiver checks for the frequency to synchronize with the sender, establishes connection with the sender. During scanning, the energy level of receiver is checked. If energy level of receiver is 'zero', it shows that the data is not received over the connected link and if energy level is more than zero, it shows that the receiver is accepting the data over the connected links. Receiver will keep on scanning whether it is receiving data or not, and will changes its frequency each time in order to coordinate with respect to the phase. Simulation will keep on performing different transitions between the sender and receiver.

4.2 Results

This section presents the verification of the simulation results using the verifier component of UPPAAL. Verifier checks the properties to confirm the reply by receiver in establishing the connection. It verifies the reply of the receiver between different time intervals. Verifier evaluated the properties on bounded runs by time and checked the property as follows:

Pr [time<=30000] (<> receiver1.Reply): It checks the probability of replying within 30000 time units and the result is between [0.594986, 0.694986].

Similarly, probability to reply within 5000-70000 time units are checked and proved to be satisfied.

To check the receival of data between the sender and receiver, energy level of receiver is checked. Property in the verifier checks whether receiver is responding for the energy level within specified interval. Following shows the checked property:

Pr [energy<=1000] (<> time>=10000): It checks the probability of letting time pass 10000 time units with a limited energy budget and the result is between [0.95, 1].

Verification checks the probability of satisfying the property within the intervals.

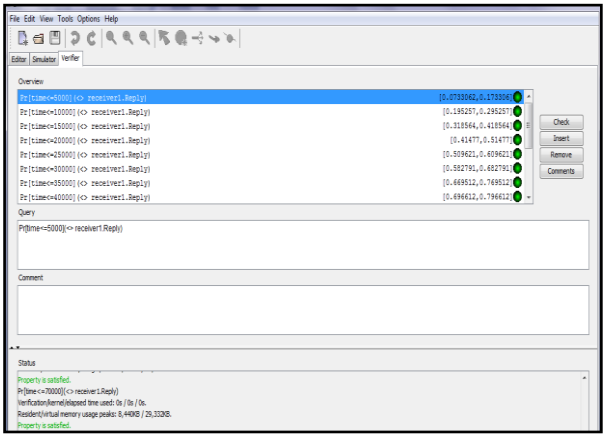

Fig. 4: Verifier

As shown in the Figure 5.5, all the properties are satisfied as represented by green dots. In the results described above, the connectivity of the receiver and sender in verified. It shows the time consumption of receiver to reply the sender. It checks whether the replies are received in the prescribed bounded

time or not. It also detects the probability of receivers energy. Simulation shows that 'zero' energy level of receiver depicts that receiver is not accepting the data or there is a loss of data and increase in the energy shows the receiver is accepting the data. The verifier of the tool gives the probability of energy detection and satisfies the property showing that the sender and receiver are connected over the network link.

5. CONCLUSIONS AND FUTURE WORK

Bluetooth is a widely used ad hoc technology for the communication over short distances. To verify the device discovery mechanism of Bluetooth ad hoc network, a Bluetooth ad hoc system consisting of a sender and a receiver is designed in UPPAAL. In the model designed, sender and receiver establish their connections using frequency hopping techniques. Two properties are checked to confirm the connectivity of the network, one confirming the connectivity of sender and receiver and another confirming the acceptance of data. The results are verified in the verifier component of the tool.

The sender and receiver connectivity is confirmed by verifying the receivers reply to the sender. The verifier component verifies the connectivity by checking the replies of receiver in different time intervals. The properties are proved to be satisfied in the verifier.

In order to confirm the data reception between sender and receiver, the energy level of the receiver is checked. From the simulation, it is found that the high energy level of receiver depicts the acceptance of data and the zero energy level depicts the non-acceptance of data. These results were verified using the verifier component of the tool and proved to be satisfied.

In future, this work can be extended by increasing the number of receivers and tracking the replies of different receivers in the tool. The sender and receiver can change their role by providing both side data communication. Verification of the tool can be checked for the properties such as trust between the different nodes.

6. ACKNOWLEDGMENTS

Genuinely thanks to my supervisor and my respectable teachers who one or in another way contributed and extended their valuable assistance in the preparation and completion of this work.

7. REFERENCES

[1] A. Dursch, D. C. Yen, D. Her Shih, "Bluetooth technology: an exploratory study of the analysis and implementation frameworks", Journal. Computer Standards & Interfaces, vol. 26, no. 4, pp. 263-277, August 2004.

[2] A. Sanghavi, "What is formal verification?" Available at: http://ww.eetasia.com/.../EEOL_2010MAY21_EDA_TA_01, May 21, 2010.

[3] A. Verma, "Formal Verification of Ad hoc Networks Routing Protocols", International Journal of Advanced Research in Computer Science, vol. 2, no.4, July-August 2011.

[4] A. Fehnker, R.V.Glabbeek, P.Hofner, A.McIver, M.Portmann, and W.Lum Tan "Automated Analysis of AODV using UPPAAL", 18th International Conference on Tools and Algorithm for the Construction and Analysis of System, TACAS 2012, Tallinn, Estonia, ISSN. 0302 - 9743, pp. 173 - 187, 2012.

[5] C. Bisdikian, "An overview of the Bluetooth wireless technology" IEEE Communications, Magazine, vol. 39, pp. 86-94, 2001.

[6] D.Camara, A.A.F.Loureiro, F.Filali, "Methodology for Formal Verification of Routing Protocols for Ad Hoc Wireless Networks" Global Telecommunication Conference, GLOBECOM' 07, pp. 705 - 709, November 26-30, 2007.

[7] F.W.Vaandrager, "A First Introduction to UPPAAL", International Journal Tretmans, editor. Quasimodo Handbook, Submitted, 2012.

[8] G. Behrmann, K. G. Larsen, O. Moller, A. David, P. Pettersone, and W. Yi, "UPPAAL – Present and Future", Proceedings of 40th IEEE Conference on Decision and Control, Orlando, Florida USA, December 2001.

[9] G. Behrmann, A. David, K. G. Larsen, "A Tutorial on UPPAAL", Available at: http://doc.utwente.nl/51010/1/Tutorial-UPPAAL-Behrmann.pdf, November 17, 2004.

[10] G. Chakraborty, K. Naik, D. Chakraborty, N. Shiratori, and D. Wei, "Analysis of the Bluetooth device discovery protocol", Journal of Wireless Networks, vol. 16, issue. 2, pp. 421-436, February 2010.

[11] J. Karlsson and A. Persson, "Device and Service Discovery in Bluetooth Networks", M.S. thesis, Department of Telecommunications and Signal Processing, Blekinge institute of Technology, Sweden, June 4, 2002.

[12] J. Valimaki, "Bluetooth and Ad hoc Networking", Available at: http://www.netlab.tkk.fi/opetus/s38030/k02/Papers/16-Jari.pdf.

[13] K. H. Torvmark, "Frequency Hopping Systems",

Available at:

http://www.ti.com/lit/an/swra077/swra077.pdf.

[14] M. Asa, and G. Hildesheim, "An efficient algorithm for inquiry process in Bluetooth", Available at: http://www.seniku.com/pdf_cn_spring_03.html, August 25, 2012.

[15] M. Duflot, M. Kwiatkowska, G. Norman and D. Parker, "A Formal Analysis of Bluetooth Device Discovery", International Journal on Software Tools for Technology Transfer, STTT, vol. 8, issue. 6, pp. 621-632, Springer-Verlag, November 2006.

[16] M. Instenberg, A. Schneider, S.schnetter, U. Heinkel, K. G. Larsen, and G. Behrmann, "Formal Methods for Abstract Specifications- A Comparison of Concepts", Department of Computer Science CISS- Center for Embedded Software Systems Distributed Systems and Semantics, Technical Report, IEEE press, pages- 6, 2006.

[17] M. Stigge, "UPPAAL 4.0: Small Tutorial", Available at: http://www.it.uu.se/research/group/darts/uppaal/small_tutorial.pdf, Nov 16, 2009.

[18] R. Brown, "Frequency Hopping Over Wireless Network ..Why Not?", Available at:

http://www.cistp.gatech.edu/.../files/CALLAHAN_AWARD_Brown.pdf, March 26, 2002.

[19] S. M. Schwartz, "Frequency Hopping Spread Spectrum (FHSS) vs. Direct Sequence Spread Spectrum (DSSS) in Broadband Wireless Access (BWA) and Wireless LAN (WLAN)", Available at: http://sorin-schwartz.com/white_papers/fhvsds.pdf.

[20] "Spread Spectrum transmission", Available at: http://www.audiotelsupport.com/site/appnotes/Spread%20Spectrum%20transmissions.pdf.