SIMBA: A SIMULATION ENVIRONMENT FOR BLUETOOTH APPLICATIONS

Marco Gönne
University of Lübeck
Institute of Computer Engineering
Ratzeburger Allee 160
D-23538 Lübeck, Germany
E-mail: goenne@iti.uni-luebeck.de

Abstract—**Management of complex ad-hoc infrastructures with certain needs on connection quality is still known to be a difficult task. In general, simulation is a good first step before implementing scenarios using hardware based platforms. Resulting from simulation, experiences on vulnerability to network disturbances or other unforeseen side effects can help adopting application scenarios before complex implementations are made for the final platform.**

As a matter of fact, current simulation environments for Bluetooth ad-hoc networks mostly concentrate on very special questions or currently provide simulation models that are primarily designed to examine network structures in conjunction with synthetic usage profiles. Simulation of specific application behavior in ad-hoc networks can be treated as an unsolved problem. This work gives a concept for a modular simulation environment capable of handling complex application scenarios. Furthermore, it provides a brief overview on the work-in-progress of the corresponding implementation.

I. INTRODUCTION

Due to the strict specification of Bluetooth and the public availability of the Bluetooth specification [Blu03], the whole interaction flow between multiple Bluetooth devices underlies are very deterministic behavior that can be predicted easily by any application developer. In locations that allow perfect and error free communication it leads to constant data rates and latencies completely free of any jitter. Therefore Bluetooth seems to be an ideal communication standard for wireless applications with fixed need of bandwidth, low latency or even realtime awareness. But in reality, Bluetooth appears to be very sensitive to environmental influences what in some conditions leads to unsteady bandwidth and latency what may form a problem for applications relying on certain network capabilities.

Regarding the Bluetooth specification, data packets are transferred using a combination of optional as well as mandatory forward error correction schemes and additional checksum mechanisms to ensure the integrity of transmitted data. If integrity checks fail retransmissions are triggered autonomously and completely invisible for the user-space application. Retransmitted packets, however increase the transmission time for the affected payload and the amounts of packets sent in total and, thus, reduce the available bandwidth. On the other hand this retransmission scheme also behaves very deterministic allowing a systematical analysis of retransmission-related bandwidth and latency deviation in relation to specified cases of disturbance. For applications depending on certain throughput and/or latency properties such as medical surveillance systems the simulation results may be used to derive appropriate usage parameters to allow proper operation in an adequate amount of cases. Unfortunately, the Bluetooth API hides all the transfer characteristics from the application and only provides limited abilities to react on the communication behavior. So, safe operation parameters can only be estimated experimentally without detailed knowledge of the impact on the lower layers. To guarantee safe operational parameters for applications with certain demands on communication infrastructure more detailed knowledge on the communication behavior is a crucial problem.

Prediction on synthetic radio disturbance schemes with given distribution functions and synthetic network load models can be handled by statistical models like Markov chains very well [MZP04]. More complex scenarios using multiple kinds of packets with different failure sensitivity, partial disturbances/interferences on selected channels or temporary disturbances, interoperability with other network technologies as well as the analysis of non-constant traffic profiles are difficult to be modeled using mathematical approaches and can therefore be handled more detailed with simulation approaches.

In the next sections the most relevant details on Bluetooth communication technology will be given and the state of the art of current Bluetooth simulation environments will be discussed briefly. In section V an own simulation model will be introduced that closes the gap on tracing the transfer flow of data packets through the whole communication process covering detailed information on delays, error correction and retransmissions.

II. BLUETOOTH

Before giving details on Bluetooth simulation, the relevant details on Bluetooth [Blu03] are summarized in this section. Bluetooth personal area networks (PAN) are arranged in piconets that consist of a master device and up to seven slaves. Multiple piconets can form larger scatternets where the master of each piconet also has to be a slave in a neighboring piconet.

The Bluetooth radio model uses low power radio transceivers of range from 10m to 100m that operate at a base frequency of 2.45GHz. The available radio band of 83.5 MHz is divided into 79 channels of 1 MHz width each. Bluetooth only utilizes one channel per time. The channel is switched 1600 times per minute regarding to a channel hopping scheme derived from the piconet master device address and the master device clock. With the Gaussian Frequency Shift Keying (GFSK) modulation and a signal timing of 1µs,

Proceedings 20th European Conference on Modelling and Simulation
Wolfgang Borutzky, Alessandra Orsoni, Richard Zobel © ECMS, 2006
ISBN 0-9553018-0-7 / ISBN 0-9553018-1-5 (CD)

this results in a theoretical transfer limit of 1 MBit/s.

Bluetooth device activity can be summarized by four main operation modes:

- The Standby state depicts the power saving state that is used whenever the device is not involved in any piconet activity.
- The inquiry procedure is a very robust mechanism to scan for any Bluetooth devices without any knowledge about the remote devices. It returns a list of scanned devices containing their unique device address and the clock offset to the inquiring device clock.
- Paging is the procedure to establish a connection between two devices. The paging device sends a connection request to a specific remote device. If the requested remote device is in range and has page scanning mode enabled, the paged device has to answers this request to establish a connection. The speed and reliability of the paging procedure depends on the knowledge of the paged device clock. With exact knowledge of the remote clock the channel hopping sequence can be synchronized precisely and best results can be reached. The common way to estimate the clock offset between the paging device and the paged device is a preceding inquiry.
- The connection state is entered if the device becomes an active member in a piconet as master or slave device.

The Bluetooth architectural model is built up from different layers. The higher layers are typically implemented as a part of a software stack on the host device. The lower parts are generally implemented at the Bluetooth device core. The lower layers are accessed by the HCI layer which forms the lowest user accessible layer. It provides a standardized interface to all Bluetooth communication functionality. Additionally, it is used by Bluetooth software stacks to map the higher layer functionality to the lower layers. Thus, the layers from the physical radio layer to the HCI build the required core to simulate a fully functional Bluetooth environment. These layers are namely:

- PHY/radio (RF)
- Baseband
- Link controller (LC)
- Link manager (LM)
- Host controller interface (HCI)

Bluetooth timing is synchronized by the piconets master device clock. Slave clocks use an offset to estimate the master clock which is needed together with the master device address to calculate the piconets channel hopping sequence. Bluetooth radio access is exclusively granted to one device at a time using time slots of 625µs. The scheduling scheme is completely controlled by the piconet master. Bluetooth internal messages are limited to packets fitting into one time slot. User payload packets span over a single slot, three slots or five slots.

Data payload is transferred using either synchronous (SCO) or asynchronous links (ACL). Synchronous links provide a robust communication channel of 64 Kbit throughput that is mostly used for voice transmission. Asynchronous links provide data throughput of up to 433.9 kbit/s for symmetric transfer and 723.2 / 57.6 Kbit/s for asymmetric transfer links when using packets of maximum payload size (5 slots packet, no payload forward error correction). Therefore asymmetric links are the choice for most data link applications and will be the main focus here.

Fig. 1. ACL Packet Structure

Asynchronous links use a family of packet types that mostly differ in payload size and their use of optional forward error correction for payload data. The general format of an ACL packet is summarized in figure 1. All ACL-packets consist of a trailing access code block, a packet header and the data payload. The access code contains a synchronization pattern needed for radio timing synchronization and allows receiving devices to decide if the packet is addressed to its logical channel. The access code is not compared bitwise but with an auto-correlation mechanism. The configuration of the auto-correlator defines the robustness against erroneous bits in the access code or trailing/missing bits due to synchronization drifts. The packet header contains the active slave address, packet type, flow control, sequence numbering, acknowledgment information and an 8 bit header checksum (HEC) that ensures the correctness of the header information. The header is additionally protected by a forward error correction code that replicates the data 3 times. The payload field consists of a payload header, the payload data and a 16 bit CRC checksum. Payload header and payload data are treated as a single block on the link control layer. Different to high data rate packets (DH1, DH3, DH5) the data payload is optionally secured by a forward error correction scheme if medium data rate packets (DM1, DM3, DM5) are used. This forward error correction scheme groups the whole payload into blocks of 10 bit size and appends 5 bits of redundant error correction information to each block.

Failures on the packet header checksum or the payload checksum trigger a retransmission of the whole packet. These retransmissions occur autonomously on baseband/link-controller level and is completely out of the users control. Retransmitted packets of course increase the transmission time for the affected payload and the amounts of packets sent in total and thereby reduce the available bandwidth.

III. Existing Simulation Approaches

Bluetooth currently represents one of the mostly used ad-hoc communication platforms and has been targeted by different simulation approaches. All of them differ in topics of interests and depth of analysis. A brief selection of the most referred systems shall be given in this section.

Due to the different fields of application that Bluetooth or other wireless communication systems can be used in, there are different approaches to behavioral simulation of wireless networks. With regard to their area of application they can be categorized in three classes of simulation environments:
1. Interface Simulation: Simulation of wireless communication by its external interface. This kind of simulation is mainly intended for development and testing of mobile ap-

plications. It does not necessarily cover simulation of the network architecture, the internal processing flow, the transport characteristics or security mechanisms besides of their external appearance to the application.
2. Infrastructure Simulation: Simulation of the network architecture and device interaction including organizational structures like piconets and scatternets. This class of simulation may include an abstraction of the radio communication as far as it is needed for the simulation of device interaction but does not necessarily cover disturbed radio or error correction mechanisms.
3. Transport Simulation: Representation of the transport flow down to the physical layer including radio interferences.

Pure interface simulation suites that are often hosted in application development suites for end-user application such as cell phone application are not taken into account here because they address a completely different application domain. The most important simulation environments will be discussed in the following:

- Bluehoc: Bluehoc developed of IBM India is one of the most cited simulation environments for Bluetooth. It is based on the NS-2 network simulator [BEF+00], [Fal00] and covers the physical layer, Baseband, Link Manager, and L2CAP protocols. Its main focus is set on infrastructure simulation. Physical constraints like mobility or explicit implementation of the radio layer have only been addressed at a high level of abstraction. Radio propagation uses the NS2 free space model representing the communication range of the transmitting device as a circle around it. Receivers within the circle receive all packets, scanning devices outside of the communication range receive nothing [Fal00]. Packet loss is modeled in a very simple table-based fashion. Applications can be simulated using predefined traffic patterns of NS-2 which covers TCP/UDP packets with source behavior models such as FTP, Telnet or HTTP [SI03]. The last releases of Bluehoc are dated in the year 2001 and though it misses the current extensions of the Bluetooth standard.
- UCBT: The UCBT [WA04] Bluetooth extension for NS-2 is also based on the NS-2 network simulator. It represents a more complete infrastructure simulation environment than Bluehoc that is partially ported to the current Bluetooth standard version 1.2. It provides an extended interference model based on interference queues as well as the original Bluehoc mechanism but physical channel / radio layer is not implemented either. Mobility is implemented using the Wireless LAN (WLAN) node model of NS-2. Application simulation also uses the NS-2 functionality as described for Bluehoc. UCBT is still under development and apart of its functional volume it is currently difficult to use due to the missing documentation.
- Suitetooth [Hig01] is the official Bluetooth implementation of the commercial OPNET network simulation framework. Suitetooth is referenced as an extraction of Bluetooth key features with a strong focus on transport and interference simulation including an explicit model of the physical radio layer. Suitetooth benefits from the network simulation functionality of OPNET giving a strong focus on analysis of TCP/IP transport over Bluetooth links. The link management protocol layer (LMP) is not implemented in Suitetooth but a user interface for own link manager implementations is provided instead. Therefore many characteristic Bluetooth functionalities depending on the link manager e.g. piconets are not supported without further development. Application simulation is implemented uses OPNET traffic source profiles similar to the NS-2 based systems.
- Conti and Moretti [CM05] give a new approach by referring to Bluetooth link manager and baseband simulation based on the SystemC hardware modeling platform. The projects addresses the power dissipation and performance analysis of Bluetooth in presence of noise with respect to the use of different packet types. The simulation environment provides a custom application interface for embedding SystemC based Bluetooth applications. The host controller interface and the physical layer are left out of scope in this approach.

Furthermore there are other simulation approaches addressing detailed aspects of Bluetooth e.g. Xiong and Pollard [XP05] do detailed analysis on Bluetooth transport reliability and provide a simulation model for the GFSK modulation of the Bluetooth radio layer.

Currently, there is no simulation environment covering all Bluetooth layers from PHY to HCI. Neither does any of the named systems provide an interface capable of applying it to custom applications. Typically the simulation environments provide application behavior interfaces controlled by scripting languages allowing basic creation and movement operations in conjunctions with synthetic traffic generators and loss models.

IV. Simulation Requirements

From the networking point of view information on connection robustness, effectively available bandwidth and latency characteristics are important measures that have to be taken into account for the network design.

As stated in section II Bluetooth packets have different safety features for different packet types and even for the different parts of a single packet. Therefore transmission errors have different impact on the communication depending on the position of their occurrence in the packet. Regarding this, it will not be sufficient to simulate a packet as a single event with static loss and interference probabilities to get a detailed view on communication performance and reliability.

When certain information on the network layer characteristics are needed it becomes mandatory to simulate the communication down to the resolution of single network token leading to a time-slice based simulational model.

V. Simulation Concept

The given concept basically uses strict encapsulation of the different entities acting together in the process of wireless communication. All entities interact using defined interfaces that try to represent the abilities of the 'real life scenario' as close as possible. Figure 2 gives an overview of the involved entities. Their function is explained in the following sections.

The base of the simulation is built up from a simulated world called BTWorld that interacts with multiple radio en-

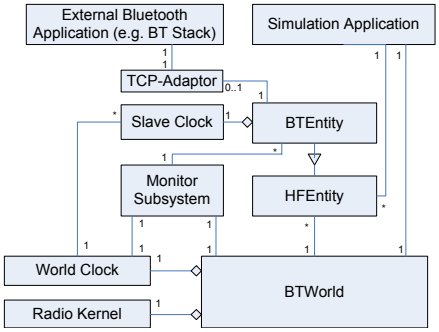

Fig. 2. Entity-Relationship Model of the Simulation Environment

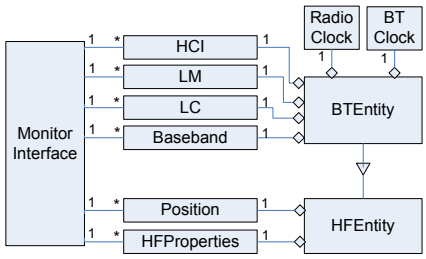

Fig. 3. Structure of Radio Enabled Entities

abled mobile entities named HFEntities. All HFEntities are related to a corresponding location record that represents its geographical position in the simulated world and to a HFProperties record that represents its current radio setup (e.g. the frequency the device is tuned to).

A. Timing

Timing and synchronization are critical issues on Bluetooth communication. To gain precise information on device interaction preceding the synchronization process as in connection establishment, inquiry or piconet changes on scatternets, the timing model should orient as much on real behavior as possible. This implies the use of time-slice based simulation instead of an event driven architecture or statistical models.

The BTWorld entity is related to a single master clock that represents the global simulation clock and is the base for all other clocks instantiated in the simulation environment. Slave clocks are instantiated by mobile devices. Slave clocks may directly forward the timing of the master clock to simulate a perfect clock or they may simulate clock skews by dropping clock ticks of the master clock. Simulation of a single slave clock running faster than the others has to be simulated originating from the master clock by dropping clock cycle of the slower slaves because no slave should be clocked faster than the Radio Kernel to prevent false detections of erroneous radio access.

Timing requirements can directly be derived from the Bluetooth standard. Bluetooth symbol transmission time of 1µs directly implies a minimum simulation timing resolution of 1µs. For the maximum useful resolution it can be regarded that Bluetooth radio timing is resynchronized with the synchronization word at the beginning of each packet. With a maximum packet length of 2871 bits in DH5-Packets, the specified minimum radio timing accuracy of +/- 20 ppm leads to a maximum error of $2 * 20/1000000 * 2871\mu s = 0.12\mu s$ [BS02], [Blu03]. This is less than one eighth of the symbol duration giving the receiving radio module enough time to accurately decode all symbols until the next resynchronization takes place. Therefore a correctly synchronized Bluetooth piconet will never suffer that much clock skew so that a simulation resolution of more then 1µs would give an increase on simulation quality.

B. Radio Enabled Entities

All radio enabled entities which are namely Bluetooth devices (BTEntities) or sources of local radio disturbance have to extend the abstract HFEntity model. HFEntities mainly provide a position record and an HFProperties record. The position record keeps the local positioning information in the simulated world. See figure 3 for the structure of HFEntity models. The HFProperties record represents the current radio configuration as the transmission power or the frequency the simulated antenna is currently tuned to. BTEntity represent the Bluetooth enabled extension of the HFEntities. BTEntities consists of a multi-layer model correlating to the layers of Bluetooth data processing. Additionally the BTEntity contains a simulated radio clock derived from the world clock and a Bluetooth clock derived from its own radio clock. The radio clock provides the timing needed for the symbol duration at the medium where the Bluetooth clock gives the half slot timing (312.5µs) needed for almost all timing constraints of baseband and link controller layer. The radio clock corresponds to the Bluetooth clock as defined in the Bluetooth specification [Blu03] and the BTClock corresponds to the simulated oscillator crystal timing.

C. Monitoring Interface

All layers of Bluetooth packet processing are connected to a monitoring interface. Each data or command packet triggers a monitoring event for each processing layer it passes. Together with the events information on the device state, device clock as well as layer specific actions such as encryption, checksum calculation, error correction and retransmit filtering are provided to the monitor instance. Monitor instances can be user specific acquisition modules, graphical interfaces or the system internal file logger. By masking out selected layers of the Bluetooth protocol stack, the monitoring subsystem gives the application developer all the needed information to trace back the influence of the chosen setup down to all layers of Bluetooth packet handling.

D. Simulation Control

The simulation flow can be controlled by a dedicated simulation application that is able to instantiate HFEntities in the BTWorld and to access the HCI. Alternatively, it can associate the HCI to a system socket, enabling the simulation to connect to an external application. In either case the simulation application is responsible to spawn in HFEntity into

the BTWorld environment and manage the positioning and the movements of the mobile entity.

E. Radio Model

Though it is the most critical part to assure a correct simulation behavior, the simulation of the radio medium is encapsulated in a separate module that is connected to the BTWorld entity. The interfacing between BTWorld and the radio kernel is done by an interface providing a transmit operation that take the current payload symbol, the senders location record and the HFProperties as arguments. The second operation provided by the radio kernel is a receive method that takes the receivers location record and HFProperties to synthesize the resulting payload symbol from the radio state. The radio kernel takes the responsibility to verify all radio activity for validity.

VI. Radio Kernel

With the fact that Bluetooth devices can only possess one antenna per device that can either be in transmit or in receive mode for the duration of one time-slot the radio entities can be modeled by three states:

- no radio access
- transmit
- receive

To derive result for the receive operation from the media state, all transmit operations have to be finished before the receive operation can take place. Therefore the time-slice is divided into two stages: the TX-PHASE for transmissions and the RX-PHASE for receive operations. These stages are delimited by synchronization barriers.

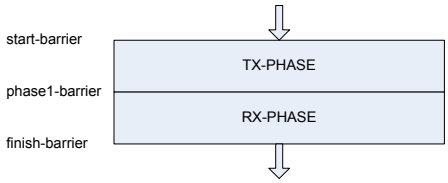

Fig. 4. Radio control flow

Each simulation round begins with a start barrier that all participating devices have to pass for the simulation round to start. The stage1- and the finish-barrier are initialized with the amount of devices that passed the start-barrier. Each barrier can only be passed if it has been reached by all devices. If a devices triggers a transmit operation its transmission payload symbol is stored together with the state of its radio parameters and its position coordinates. A transmit operation automatically calls the stage1-barrier which causes the device to be blocked until all remaining devices have called the stage1-barrier as well. Thereby it is guaranteed that all devices are only able to transmit once per round. If a device executes a receive operation the stage1-barrier is reached and the device is blocked until the stage1-barrier is released. After all devices have reached the phase1-barrier, it is unlocked and the RX-PHASE is triggered. For each device that has been blocked in a receive operation its result is calculated by a comparison of the receivers radio parameters and location coordinates compared to the transmissions stored in the TX-PHASE. If any device tries to call a receive or transmit operation after the stage1-barrier has been passed this indicates an error in the simulation implementation. If all devices finish their simulation round they reach their finish routine that triggers the finish barrier. If a device does not trigger any receive or transmit action in the current round, it directly reaches its finish routine without being blocked by any barriers. In this case the finish routine also counts down the stage1-barrier not to lock the other devices in the RX-PHASE. After all devices have reached the finish barrier the beginning of the next simulation round is triggered for the simulated devices and the radio kernel.

The level of detail regarding the radio modulation and the simulation aspects are likely to be kept customizable. This allows to implement time-efficient models similar to the 'free space model' of Bluehoc (see section III) or more complex simulation models e.g. given in [XP05] or [SHS02].

VII. Implementation

The concept has been realized as a Java framework currently referenced as 'SimBA'. The actual state of implementation covers most common functionality of the HCI that is needed for inquiry of devices, connection establishment and AIL data transfer.

Header error checksums, payload CRC checksums as well as forward error correction have been fully implemented regarding the Bluetooth Specification [Blu03]. Implementation of synchronous connections (SCO) and encryption functionality have been postponed. Frequency hopping is still limited to the non-adaptive scheme from Bluetooth version 1.1, but the adaptive scheme of Bluetooth 1.2 is scheduled for implementation.

The system comes with two different radio kernel implementations. The first one uses a simple radio propagation model similar to BlueHoc (see section III).

The other radio kernel is based on the path-loss and interference model given in [Mor02]. This simulation model allows to give a path-loss parameter n as a measure for the environmental communication characteristics. The basic principle for path-loss PL calculation is the equation

$$PL = 20log\left(\frac{4\Pi}{\lambda}\right) + 10n * log(d)$$

with distance d between the sending and receiving device, wavelength λ and *path-loss exponent* n. The wavelength λ for a frequency of 2.4 GHz is fixed at $\frac{299{,}792{,}458 m/s}{2.4*10^9/s} \approx 0.125m$ [Mor02]. This substitution gives the final basic path-loss formula

$$PL = 40 + 10n * log(d)$$

that can be used to calculate the path-loss for given d and n. With the path-loss given in dBm and the knowledge of the senders output power it can be estimated if the receiver is able to receive the signal properly. It can also be solved to d as the maximum range for given path-loss PL and path-loss exponent n in the form of $d = 10^{\frac{PL-40}{10n}}$ where PL is the difference between the transmitter output power in dBm and the receiver sensitivity, giving the maximum transmission range for the devices configuration.

This model will allow a more realistic simulation of communication characteristics by a moderate reduction of simulation performance. Assuming the path-loss exponent n is fixed for one simulation run, this model causes an overhead of one logarithm-operation and one multiplication per transferred token multiplied by the number of possible receivers.

VIII. Conclusion

In this paper some of the most common simulation environments have been analyzed regarding the completeness of the simulated Bluetooth implementation and their usability with custom ad-hoc network applications. The research came to conclusion that none of the chosen environments provide sufficient support for simulation based analysis of user defined ad-hoc networking applications.

Based on this conclusion, requirements for a simulation environment capable of simulating custom Bluetooth application have been cumulated and a custom simulation concept has been developed and partially implemented.

The simulation framework allows implementation of custom Bluetooth applications and detailed analysis of its communication characteristics down to logical transport or even the physical transport layer. So, resource critical applications can be optimized for proper operation by analysis of the logical and physical transport characteristics which are completely hidden by the Bluetooth API in hardware implementations. The impact of environmental radio characteristics can be analyzed by adjusting the path-loss parameters to get detailed information on the application behavior in locations with different radio characteristics. By the results of the simulation, the target application can be optimized for bandwidth or latency by tracing back the impact on different packet sizes and error correction schemes.

References

[BEF+00] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, 2000.

[Blu03] Bluetooth SIG. *Specification of the Bluetooth System. v1.2*. www.bluetooth.org, 2003.

[BS02] Jennifer Bray and Charles F. Sturman. *Bluetooth 1.1: Connect without Cables*. P T R Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 2002.

[CM05] M. Conti and D. Moretti. System level analysis of the bluetooth standard. *Proceedings of the conference on Design, Automation and Test in Europe*, 3:118–123, 2005.

[Fal00] K. Fall. *ns Notes and Documentation*. www.isi.edu/nsnam/ns/ns-documentation.html, 2000.

[Hig01] Highland Systems, Inc. *Bluetooth Simulation Model Suite for OPNET*. http://www.highsys.com/products/Suitetooth.pdf, 2001.

[Mor02] R. Morrow. *Bluetooth: Operation and Use*. McGraw-Hill Professional, New York, USA, 2002.

[MZP04] D. Miorandi, A. Zanella, and G. Pierobon. Performance evaluation of bluetooth polling schemes: an analytical approach. *Mobile Networks and Applications*, 9:63–72, February 2004.

[SHS02] R. Schiphorst, F.W. Hoeksema, and C.H. Slump. Bluetooth demodulation algorithms and their performance. *Proceedings of the workshop on Software Radios*, pages 20–21, 2002.

[SI03] M. Subramani and M. Ilyas. Simulation Based Analysis of Bluetooth Networks. *2003 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pages 688–694, 2003.

[WA04] Q. Wang and D. Agrawal. *UCBT – Bluetooth extension for NS2*. www.ececs.uc.edu/ cdmc/ucbt, 2004.

[XP05] X. Xiong and J. Pollard. Modelling for bluetooth pan reliability. *Proceedings of 19th European Conference on Modelling and Simulation*, pages 580–584, 2005.