Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2010, Article ID 642531, 15 pages
doi:10.1155/2010/642531

Research Article

A High-Accuracy Nonintrusive Networking Testbed for Wireless Sensor Networks

Wei Huangfu,[1] Limin Sun,[1] and Jiangchuan Liu[2]

[1] *Institute of Software, Chinese Academy of Sciences, Beijing 100190, China*
[2] *School of Computing Science, Simon Fraser University, Burnaby (Metro-Vancouver), Canada BC V5A 1S6*

Correspondence should be addressed to Wei Huangfu, david.huangfu@gmail.com

Received 15 February 2010; Accepted 7 June 2010

Academic Editor: Dan Wang

It becomes increasingly important to obtain the accurate and spontaneous runtime network behavior for further studies on wireless sensor networks. However, the existing testbeds cannot appropriately match such requirements. A High-accuracy Nonintrusive Networking Testbed (HINT) is proposed. In HINT, the interconnected chip-level signals are passively captured with auxiliary test boards and the captured data are transferred in additional networks to test server. The test server of HINT collects all the test data and depicts the full network behavior. HINT supports networking test, protocol verification, performance evaluation and so on. The experiments show that HINT transparently gathers accurate runtime data and does not disturb the spontaneous behavior of sensor networks. HINT is also extendible to different hardware platforms of sensor nodes. Consequently, HINT is an upstanding testbed solution for the future fine-grained and experimental studies on the resource-constrained wireless sensor networks.

1. Introduction

A wireless sensor network (WSN) is usually composed of numerous micro low-power autonomous sensor nodes which collaboratively sense, process, and transmit specific interesting data in the monitoring area. In recent years, wireless sensor networks have attracted widespread attention in the international academic.

The network testing is important for the research and development of sensor networks. The foundation of the network testing is to perceive the network behavior. The runtime data which represent the network behavior are essential to verify network protocols, evaluate network performance, and so forth. The more accuracy and more spontaneous the runtime data are, more exactly and more deeply we understand wireless sensor networks. However, the network testing is challenge for wireless sensor networks. Wireless sensor networks are distributed resource-constrained embedded systems. It is difficult to generate and transfer runtime test data in such networks due to the constrained resource, the dynamics of wireless communications, the frequent failures of sensor nodes and the variety of the applications for wireless sensor networks.

Some testbeds or test platforms for wireless sensor networks are already present in the recent years. They are mainly divided into two categories according to the mechanisms of transfer the runtime test data. One is to transfer the test data over the wireless links of the wireless sensor network itself. The other is to transfer the test data over additional networks. For the former category, the bandwidth of the wireless sensor networks must be consumed for the purpose of collecting test data. Obviously, the test platforms of the latter category do not disturb the wireless communications of wireless sensor networks at all because the test data are not transferred over the wireless sensor network itself. However, for all of existing test platforms, the runtime test data are generated by the microcontrollers in the sensor nodes and the computing resource of the sensor nodes are consumed during the network testing. Therefore all these test platforms are still intrusive and they interfere with the spontaneous behavior of the wireless sensor networks. Moreover, the test data collected from the sensor nodes are not so precise

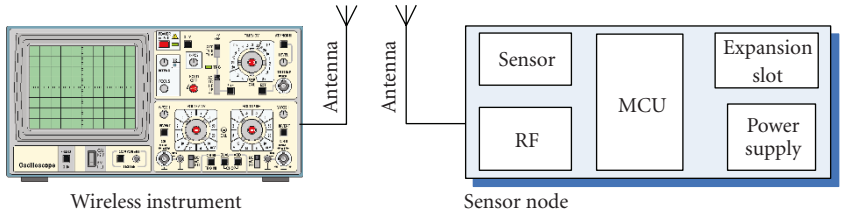


Figure 1: Gathering test data with instruments.

because of resource constraints of nodes themselves, such as computing capacity of the microcontrollers and the precision of the crystal oscillator of the sensor nodes. In summary, none of the existing test platforms provide users accurate information on the runtime data to represent the spontaneous network behavior of wireless sensor networks. The key issue here is how to observe the network status accurately in a nonintrusive manner, that is, with no side effects upon the network behavior itself.

A novel testbed HINT (High-Accuracy nonIntrusive Networking Testbed) is proposed. The test data are generated by auxiliary devices and transferred over additional networks in HINT. The major idea is to passively and accurately probe the internal chip-level signals inside sensor nodes in order to access the runtime network data in a nonintrusive and precise manner. The auxiliary devices run like many smart spies. They each probe the internal interconnected signals inside a sensor node with extra tributary wires, and then transfers the accuracy test data to the testbed server over additional networks. The testbed server of HINT collect, parse and understand all the test information, and then reconstruct the sensor network behavior.

In the rest of the paper, we first review relevant work on the existing test platforms and the architecture of typical sensor nodes. Next we present the major ideas and challenges of the design of HINT. In the next section we introduce the implementation of HINT aiming at the characteristics of typical hardware platforms for present sensor nodes. Then we show the experiment studies for HINT. Finally, we conclude with a discussion about HINT and with future work.

2. Related Works

Apparently, the awareness of the network behavior is important for the studies on wireless sensor networks. There are mainly two kinds of methods, that is, simulation and experiment, to gather the runtime data of wireless sensor networks.

NS2 [1], Glomosim [2], and OMNet++ [3] are general simulation tools, while TOSSIM [4], EMStar [5], ATEMU [6], SensorSim [7], SENS [8] are specific simulation softwares for wireless sensor networks. With simulation tools, researchers are able to gather the internal status of sensor nodes and access complete runtime data of the network. However, the simulation is usually based on simplified models under ideal assumptions, whereas the actual circumstance for wireless sensor networks is complex and the wireless communications are highly unpredictable dynamic. Therefore the simulation results are not credible to some extent.

Test instruments and test platforms are usually introduced to gather runtime data in the actual applications of wireless sensor networks, especially during the research and development stages.

The experimental test method to use additional test instruments (or sniffer nodes) is shown in Figure 1. The wireless instruments will probe the network packets on the air without any influence on the sensor network itself. However, the wireless instruments have none of knowledge about the internal status of autonomous sensor nodes. Users cannot judge whether the sensor node received the packets which is observed by the instruments and vice versa.

The experimental methods for the existing test platforms can be divided into two categories according to the mechanisms of gathering data.

The former category is to obtain test data by microcontroller of sensor nodes and transfer test data over the wireless links of the wireless sensor network itself (see Figure 2). The microcontroller unit (MCU) sends test data to radio frequency (RF) transceiver and afterwards these data are transferred in the wireless sensor network hop-by-hop towards the test server which gathers all test information to process further analysis. MoteWorks [9], designed by Crossbow Technology, belongs to this category. MoteWorks supports a series of sensor nodes made by Crossbow Technology, such as MICA and MICAZ. MoteWorks can monitor network topology, link quality, and residual energy. Without any auxiliary devices, MoteWorks is a low-cost testbed solution. However, the computing resource of the sensor node and the bandwidth of the wireless sensor network must be consumed for the purpose of collecting test data. In other words, it will interfere with spontaneous network behavior. Such interference will be quite serious because the resource of wireless sensor network is very limited. Also all test platforms of this category depend on the successful transmission for runtime data over the links of the wireless sensor network, and they do not apply to the low-layer communication debugging of wireless sensor networks, such as problems in the routing protocol.

The latter category is to obtain test data by microcontroller of sensor nodes but to transfer test data over additional networks, either wired or wireless (see Figure 3). The microcontroller sends test data to the expansion slot, which is linked to an auxiliary device. This auxiliary

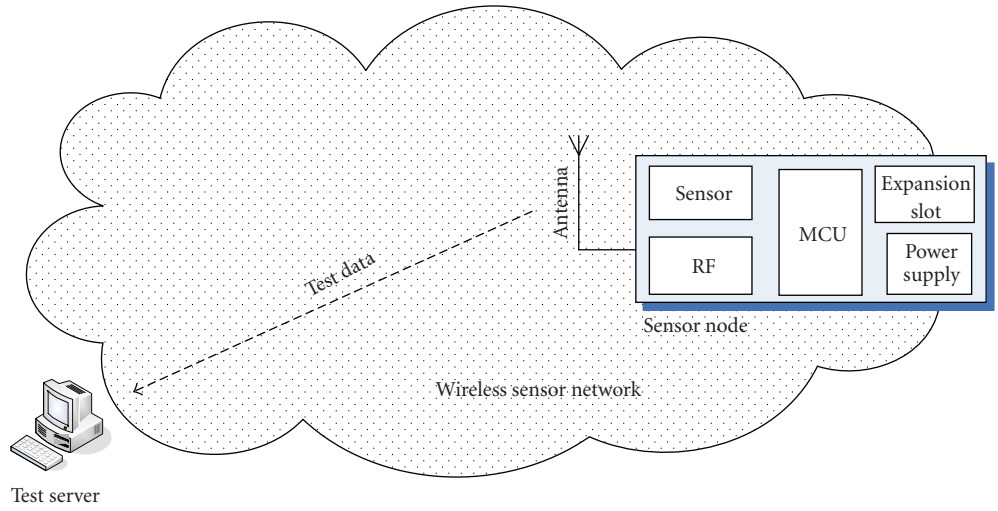


Figure 2: Gathering test data via wireless sensor network itself.

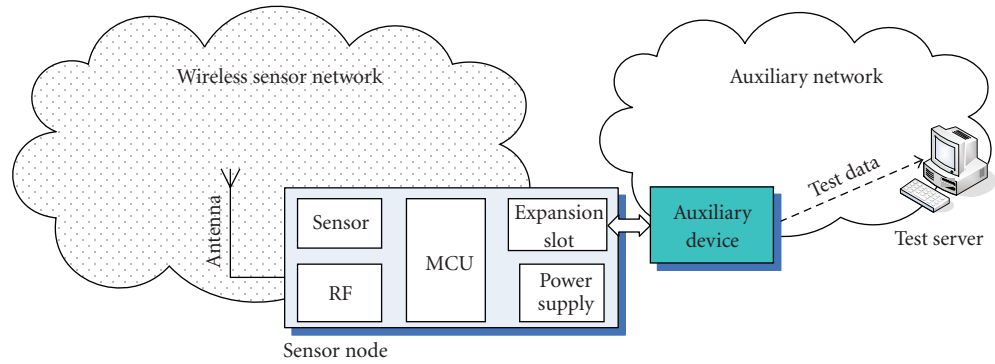


Figure 3: Gathering test data via extra network.

device forwards the data to the test server over additional networks. Kansei [10], MoteLab [11], Trio [12] and TWIST [13] all belong to this category. For example, Kansei takes XSM (with a 7.3 MHz 8 Bit CPU) as sensor nodes, StarGate (with 400 MHz PXA255 CPU) as auxiliary devices, and Ethernet and wireless local area network (WLAN) as the additional networks. The test platforms of this category do not disturb the wireless communications of the wireless sensor network at all because the test data are transferred over additional networks. However, the microcontrollers in the sensor nodes need to send test data out to the expansion slot. Hence they still interfere with the spontaneous behavior of wireless sensor network. Moreover, the test data collected from the sensor nodes are not so precise because of resource constraints of nodes themselves, such as computing capacity of the microcontroller and the precision of the crystal oscillator on the sensor nodes.

Sensor nodes are the basic cells of the wireless sensor networks. Therefore it is fundamental to learn the architecture of sensor nodes. At present there exist a large number of kinds of sensor nodes developed by universities, research institutes and companies. Some typical sensor nodes are listed in Table 1.

Table 1: List of typical sensor nodes.

Name	MCU	Transceiver
Btnode	Atmel ATmega 128L	TI CC1000
Eyes	TI MSP430F149	TR1001
EyesIFX v1-2	TI MSP430F149	TDA5250
IMote 2	Marvel PXA271	TI CC2420
Iris	Atmel ATmega 128L	Atmel AT86RF230
Mica	Atmel ATmega 103	RFM TR1000
Mica2	Atmel ATmega 128L	TI CC1000
MicaZ	Atmel ATmega 128L	TI CC2420
TelosB	TI MSP430	TI CC2420

The typical architecture of all the sensor nodes listed in Table 1 is shown in Figure 4. A typical sensor node mainly consists of MCU, RF transceiver and sensor. The components are connected with wires in PCB (Printed circuit board). In general, MCU and transceiver are discrete components in the present sensor nodes, although there are integrated chips which combine MCU and RF transceiver together, such as TI CC2430.

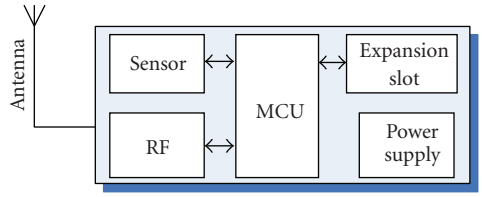


Figure 4: Typical architecture of sensor node.

TelosB is a kind of popular sensor nodes. It mainly consists of MSP430 as MCU, CC2420 as RF transceiver and sensors. There are 4 Serial Peripheral Interface (SPI) wires and 6 GPIO wires between MSP430 and CC2420. The SPI bus consists of Chip Select (CS), Serial Clock (SCK), Master In Slave Out (MISO) and Master Out Slave In (MOSI). To perform any packet sending or receiving operations, the microcontroller and RF transceiver will exchange command and data via the wires. The interactive protocols and timing requirements are present in CC2420 manual in detail. As a result, we should know full information on the packet sending or receiving if we capture and understand all signals between the microcontroller and the RF transceiver. Similarly, we should know full information on the sensor sampling if we capture and understand all signals between the microcontroller and the sensors.

In summary, the typical architecture of sensor nodes at present helps us to obtain the internal status of sensor nodes. We design and implement HINT just aiming at the characteristics of typical hardware platforms for present sensor nodes.

3. Design of Hint

We proposed a novel testbed HINT. The core idea of HINT consists of three part. First, auxiliary devices passively probe the internal chip-level signals to obtain the test data in a nonintrusive manner, which does not consume any computing and storage resource of the sensor node. Second, the test data, which represent the runtime behavior inside sensor nodes, are transferred over additional networks so that they do not consume any wireless bandwidth of the sensor networks. Finally, All the test data are collected by the testbed server and then the network behavior are reconstructed to execute a full network-scale testing. The mechanisms of HINT are shown in Figure 5.

The system architecture of HINT is shown in Figure 6. HINT consists of a testbed server and a number of test units. The testbed server and all test units are connected with additional network, usually with Ethernet.

Each test unit consists of a sensor node and a test board. All sensor nodes form a wireless sensor network while all test board form another wired or wireless network for the purpose of transferring test data. Each test board is linked to the corresponding sensor node in order to probe the internal chip-level signals inside the sensor node. The testbed server is used to collect test data from all test boards and perform future analysis.

The core idea of HINT is quite simple. However there are still a number of technical challenges in the design of HINT. First, Which signals should be captured to represent the node status by the test boards in HINT? Secondly, How should such data be collected and transferred to the test server? Thirdly, how does the test server parse the test data to reconstruct the network behavior? The solutions are introduced in what follows.

3.1. Capturing Chip-Level Signals. It is essential to decide which signals inside the sensor nodes should be captured.

The useful signals inside the sensor node are divided into 5 groups. The first group is the wires between the RF transceiver and the MCU inside sensor nodes, which provide information on the radio packet. The second group is those from the MCU to the sensor, which provide information on the sensing operation. The third group is the Joint Test Action Group (JTAG) pins of the MCU, which provide functions to reprogram and debug. The forth group is external communication pins of the MCU, such as RS-232, which provide information about external data sent by MCU. The last group is the power supply lines for the sensor node, which help us to turn on or off the sensor node and measure its power consumption. The test board is linked to the corresponding sensor node via one or more groups of wires carefully chosen. Among all the five groups, the first group, that is, the signal group between the RF transceiver and the MCU inside sensor nodes, is most important for the networking test.

The test unit is the basic cell in HINT for capturing test information. The diagram of the test unit is shown in Figure 7.

The Test board is composed of a signal acquisition and remote control (SARC) module, a CPU and an Ethernet controller. SARC is the core component of the test board and it connects all the wires from the sensor node. The main functions of SARC include signal acquisition, power supply and remote control for the sensor node. In the test board, the SARC connects to the CPU for test data exchange. The CPU can communicate with the testbed server via Ethernet. Hence all test data captured by the SARC can be transferred to the remote server.

SARC mainly passively probes these wires, which have no side effects on the spontaneous operation of the sensor nodes. In addition, SARC can also actively control some of the wires, such as the JTAG of the MCU, to provide features of remote control and debugging.

3.2. Transferring and Collecting Test Data. The captured data of raw signals are too huge to be transferred. In HINT, such raw captured data are encoded and compacted inside the SARC module of the test board. The compacted test data are then transferred to the testbed server.

The raw signals are mainly divided into two classes, that is, analog or digital. Different encoding methods are adopted for different signals.

For analog signals, the signals should be firstly converted to digital values by an Analog to Digital Converter (ADC).

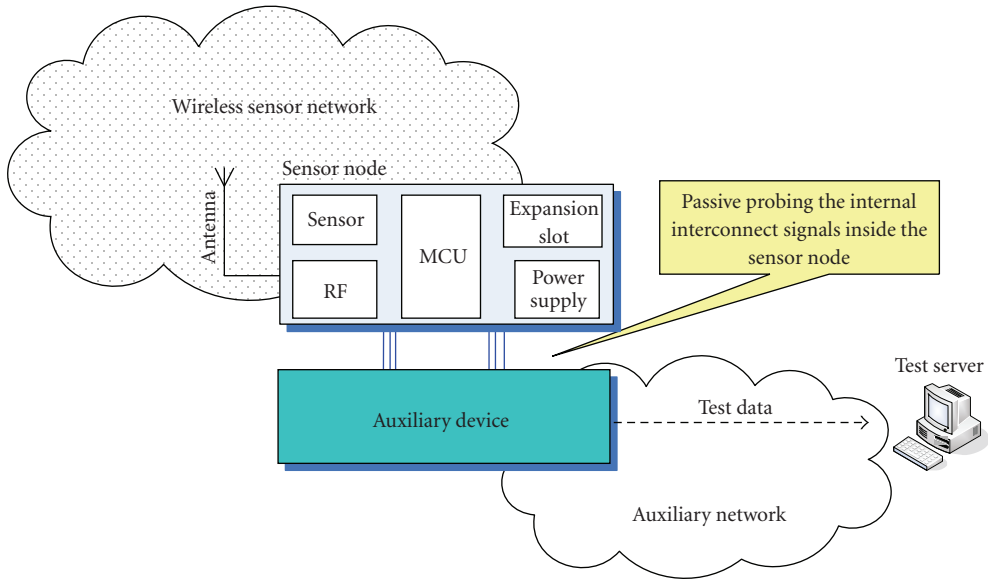


Figure 5: The mechanisms of HINT.

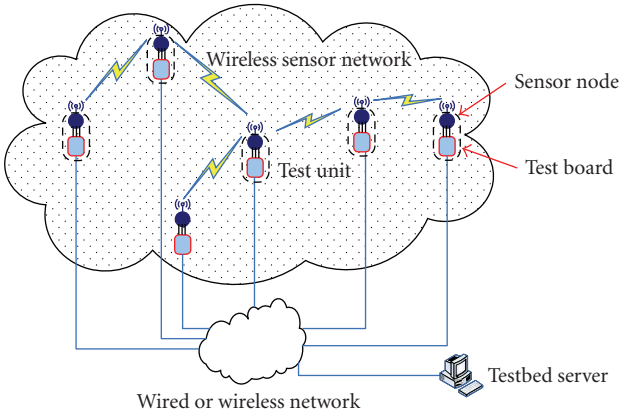


Figure 6: System architecture of HINT.

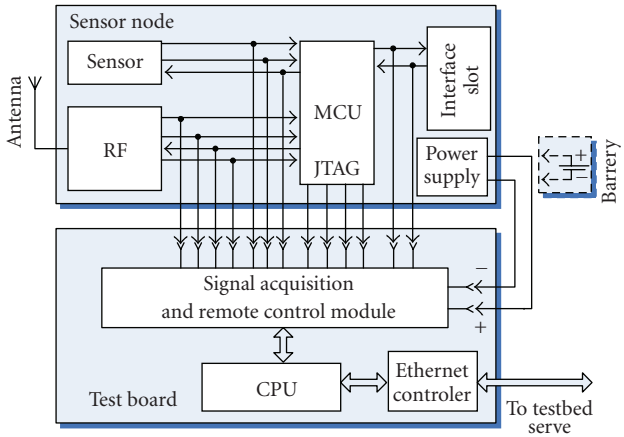


Figure 7: Block diagram of HINT test unit.

In order to decrease the amount of data, not all the sampled values will be transferred. Only the value which is beyond the threshold of last transferred value will be transferred with the corresponding timestamp (as shown in Figure 8), which will effectively decrease the amount of test data to be transferred.

The digital signals are also divided into two categories, with or without corresponding clock. For the digital signal with corresponding clock, it will be captured at the edge of the clock. For the digital signal without corresponding clock, it will be captured at the edge of internal clock driven by the SARC. Only changed values and their timestamp will be transferred for both categories.

The SARC also understands some digital communication protocols, such as SPI and CAN. Such communication data will be transferred instead raw digital signals.

With all these methods of signal acquisition, the data amount encoded from raw signals should be greatly reduced.

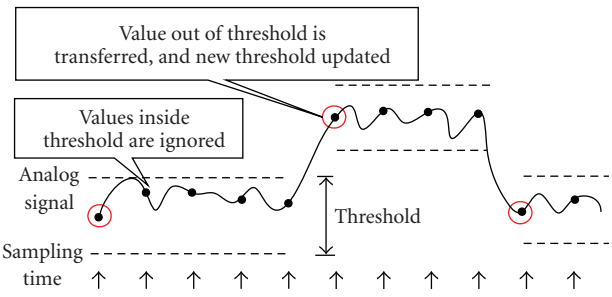


Figure 8: Analog signals acquisition.

Such compacted data are transferred by the an independent networks to the test server. The test server collect all the test data from test boards to perform the network-scale testing.

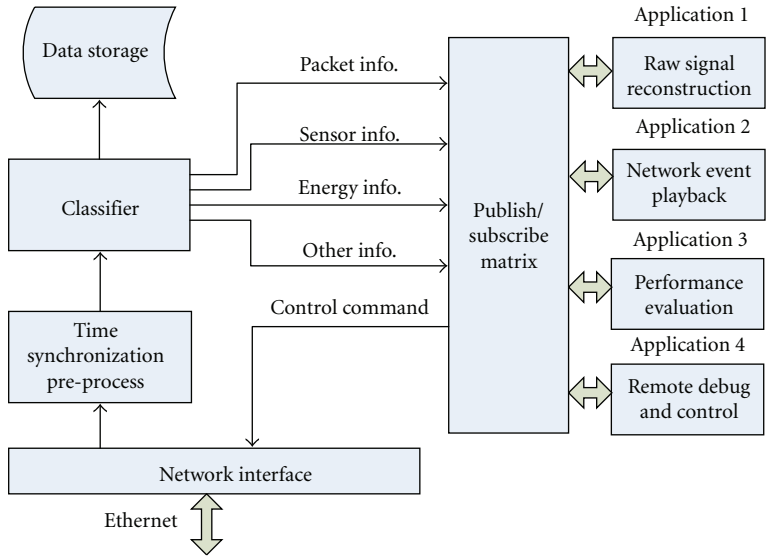


Figure 9: Block diagram of HINT test server.

3.3. Reconstruction and Analysis of Network Behavior. In HINT, all test data gathered by the test board are transferred to the test server. Different chips usually provide different electrical operation manners. Therefore the Finite State Machine (FSM) algorithm is used to parse the raw test data according to the chip-level electrical interface.

The clock synchronization among test units is the foundation of the protocol analyzing and delay measurements. In the test server, all the data received from network are firstly processed to adjust the timestamp of different test units to a unique standard timestamp corresponding to the internal time of test server. In order to obtain high accuracy of timestamp on the captured signals, three related methods are adopted. Firstly, there is a high-stability temperature-compensated crystal oscillator on each test board to provide the high-accurate clock. Secondly, each test board exchanges time synchronization messages with the test server over the additional network in order to prevent the system from synchronization drift. The time synchronization algorithms for wired network are quite mature. However, the round trip delays for the typical TCP/IP wired network are about tens of milliseconds. It is not easy to archive accurate clock synchronization to match the requirement of timing analysis in the sensor networks only with the clock synchronization between the test server and test units. If a sensor node sends a radio message in the wireless sensor network, the RF modules in all the neighbor sensor nodes will receive and decode the radio frame simultaneously. Since the HINT platform captures all these radio frame sending and receiving operations at every test unit, such accurate timing relations are helpful for the clock synchronization in the HINT platform. Therefore finally the test server collect all the sending and receiving operations among sensor nodes and adjust the clock synchronization to provide the high accuracy of the timestamps.

The test server is usually a desktop computer server or a laptop. The software diagram of test server is shown in Figure 9. All synchronized data are classified to separated categories, and saved to trace files at the same time.

There are a number of applications to analyze the network behavior for different testing purposes, such as raw signal reconstruction and visualization, protocol verification, network performance evaluation and remote debugging. These applications access test data via a Publish-Subscribe Matrix to decouple related software components.

4. Implementation

HINT is quite a complex system with mixed software and hardware. The detailed implementation is introduced as follows.

4.1. Sensor Node. The chip-level signals to be captured depend on the chips inside the sensor nodes. For instance, inside the sensor nodes like TelosB, IMote-2 and MicaZ, the RF transceiver chip CC2420 is used and hence the signals to be captured between the RF transceiver and the MCU for sensor nodes are CS, SCK, MOSI, MISO, SFD, INT, FIFOP and CCA. But inside the sensor nodes like Mica2 and Btnode, the RF transceiver chip CC1000 is used and hence the signals to be captured are DIO, DCLK, PCLK, PDATA and PALE. All these signals are explained in the CC2420 and CC1100 datasheets in detail.

Theoretically, the testbed HINT support all these various kinds of popular sensor nodes. However, these sensor nodes are not directly supported because there are no suitable expansion slots on them for the purpose of network testing. Users must connect the corresponding wires between the nodes and the test boards with clamps or by soldering, which are neither convenient nor flexible. If these internal chip-level

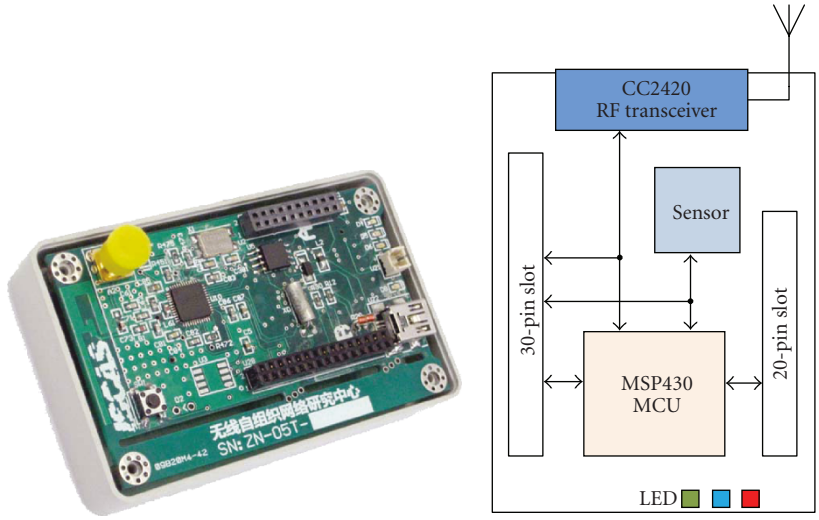


Figure 10: ZiNT sensor node.

Table 2: List of Signals to capture in ZiNT.

Group	Signals
RF transceiver	CS, SCK, MOSI, MISO, SFD, INT, FIFOP, CCA
Sensors	ADC0-ADC7
External communication	UART1TX, UART1RX
JTAG	RESET, TDI, TDO, TMS, TCK
Power supply	VCC

signals are connected to the test boards, no matter by clamps, soldering or expansion slot. the testbed HINT could obtain information enough to analysis the status of the sensor nodes and to perform the network testing.

In order to efficiently deploy the HINT testbed, we also exploited a kind of sensor node named as ZigBee Node with Testing expansion slot (ZiNT). ZiNT node is almost identical to TelosB produced by Crossbow Technology. The microcontroller and the RF transceiver of ZiNT are also TI MSP430 and CC2420, respectively. The main difference between ZiNT and TelosB is that there are more pins in the expansion slot on ZiNT for monitoring the suitable electrical signals inside the sensor nodes to support the HINT testbed directly. The actual picture and block diagram of ZiNT are shown in Figure 10.

All the signals which should be captured are connected to the expansion slot via tributary links. These signals are listed Table 2.

4.2. Test Board. The test board is the kernel component of HINT. The test board mainly consists of two parts. One part is for data processing and transferring including CPU and Ethernet controller, whereas the other for data acquisition and remote control, that is, SARC module.

We choose Atmel AT91SAM7X256 as CPU, which is a 55 MHz high-performance processor with 32-bit RISC architecture, 256 k Flash, 64 K SRAM, 10/100 M integrated Ethernet MAC controller. An Ethernet PHY transceiver DM9161A is relevantly introduced to implement network communications. Ethernet also offers power supply to test board by means of POE. Furthermore for the purpose of flexibility, the test board supports 3 kinds of power supply in fact, that is, 5V DC, USB and Power over Ethernet (POE) [14].

We adopted an embedded OS called FreeRTOS running on the Atmel AT91SAM7X CPU. FreeRTOS is an open-source real-time operating system. A light weight TCP/IP protocol stack library named as LwIP [15] has already been ported to FreeRTOS. Thus we developed some embedded software to transfer data with TCP/IP on the basis of FreeRTOS and LwIP.

We adopted a high-performance Field-programmable gate array (FPGA) Altera Cyclone II EP2C8 [16] in order to capture signals effectively. FPGA contains a number of programmable logic cells which can be configured by the customer to implement any logical function. Altera EP2C8 contains 8256 logic cells and 182 IO pins. The logical function is written in the hardware description language (Verilog).

Some auxiliary components are linked to FPGA. A 25 MHz crystal oscillator provides timing and thus the precision of timestamp to capture signals is only 40 ns. A 64 MB SRAM is used as data buffer when there occurs temporary network failure. A high-speed 8-bit analog-digital convert TLV5510 is used to acquire analog signals. A current-sense amplifier MAX4173 is used to measure the electric current to the sensor node. Four seven-segment Light-Emitting Diode (LEDs) are used to display working parameters of FPGA. A buzzer is for alarm on the test failure.

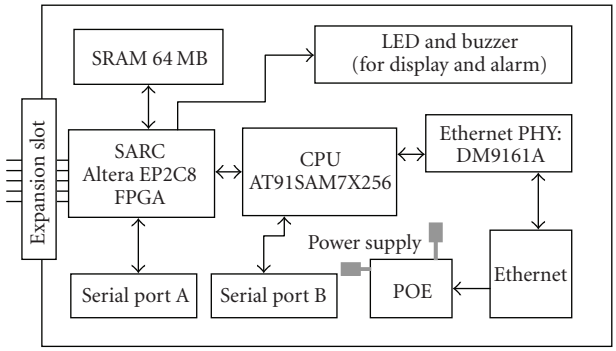


Figure 11: Block diagram of test board.

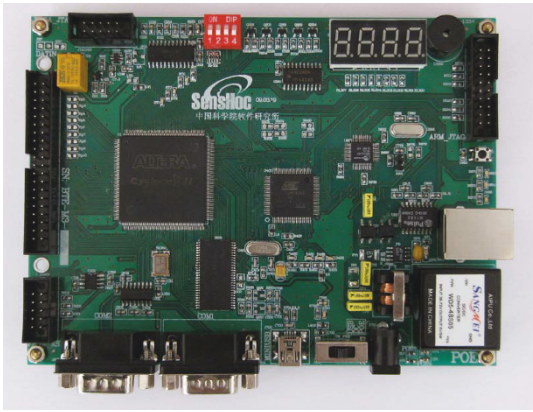


Figure 12: Actual picture of test board.

FPGA and CPU are connected via SPI interface. CPU is the master device to poll data from FPGA. FPGA will caches data in its internal storage or external SRAM while CPU is busy. It will greatly reduce the requirement for CPU response time. With the combination of FPGA and CPU, the test board offers us flexibility and extendibility both in hardware and software.

The block diagram and actual picture and of the test board are shown in Figures 11 and 12, respectively.

4.3. Adapter Board and Test Unit. One of the purposes of HINT is to support as many kinds of sensor nodes as possible. FPGA in the test board offers us flexibility for electric circuits. The adapter board will offer us flexibility for mechanical junction.

The adapter board is aimed at the expansion slot of the specific sensor node and provides two slots, of which one slot is modified to fit the corresponding sensor node and the other is fixed for the test board. With the adapter board, we cannot only easily join the sensor node and test board together to form a test unit, (see Figure 13) but also support a variety of sensor nodes via its corresponding adapter with just one kind of standard test board.

4.4. Test Server. Test server is usually a computer to run test applications in order to collect and process data received

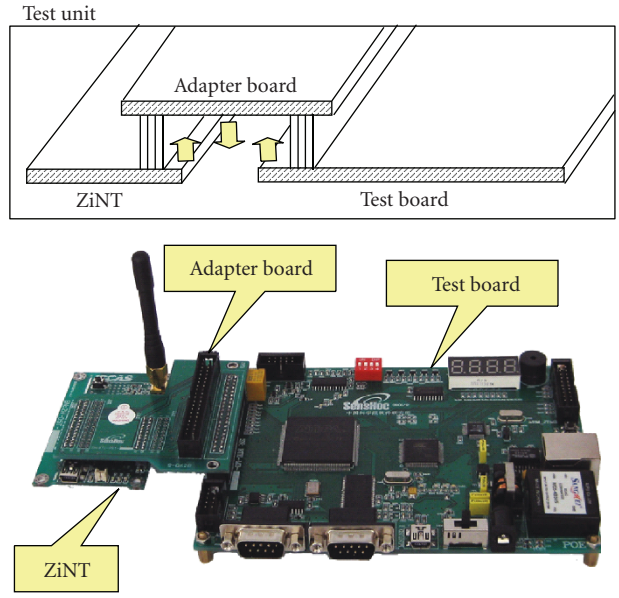


Figure 13: Mechanical structure of test unit.

from sensor nodes. All the applications are written in C++ and Python languages. Python is a flexible script language whereas C++ is an effective compiler language. The combination of Python and C++ are not only adaptable to various test requirements but also offer powerful data processing efficiency.

To comprehend the operations, that is, the radio packet sending or receiving, inside the sensor nodes, the test server unpack the test data to the raw signals and then parse the raw signals with the finite state machine algorithm. For instance, the algorithm corresponding to the CC2420 electrical interface is used for parsing the raw signals from the sensor nodes such as ZiNT, telosB and micaZ.

Once the operations in each sensor node are comprehended, the full network-scale behavior of the sensor network can be reconstructed. We have already developed a series of applications for some typical test requirements. The core application is for network data gathering and information classification. Other applications are listed as following. "Interconnect Signal Analyzer" is to reconstruct raw internal signals and visualize them. "Network History Player" is to replay the network behavior according to the stored data. "Network Performance Measurer" is to evaluate network performance such as traffic, delay and packet loss rate. Some screenshots of these applications are shown in Figure 14. WxWidgets is chosen for the Graphics User Interface (GUI) library, because it is a cross-platform library to support both MS Windows and Linux operating system.

5. Experiments

A series of experiments are conducted in order to fully study the features of HINT testbed. The experiment environment includes a center server (IBM Notebook T43), 20 test units (ZiNT nodes and their corresponding test boards)

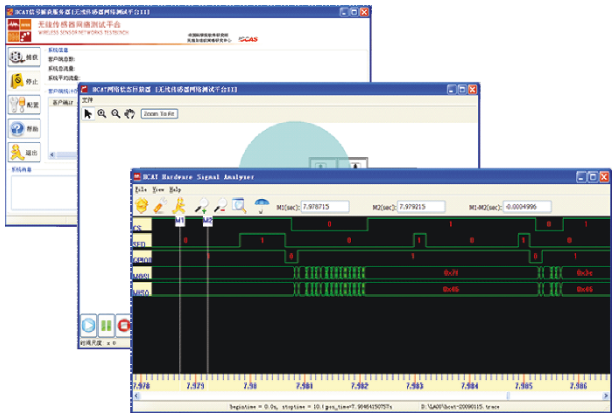


Figure 14: Screenshots of test server applications.

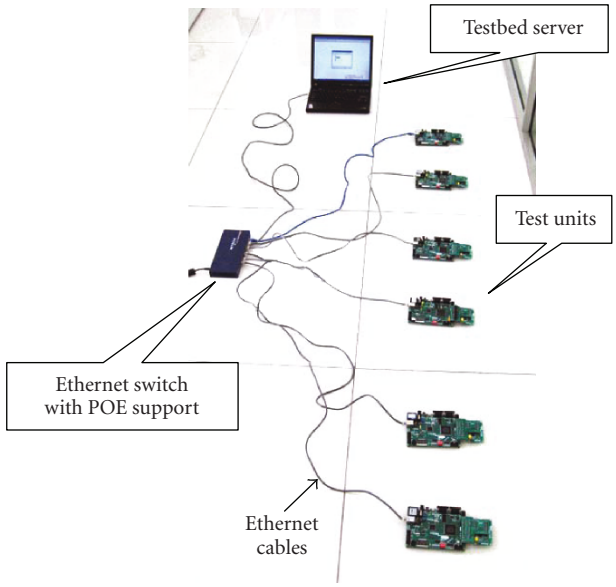


Figure 15: An experiment scenario of HINT.

and several fast Ethernet switches with POE (NETGEAR FS108P). Moreover, an oscilloscope (RIGOL DS5022ME) and a logic analyzer (LA1016) are adopted for the purpose of comparison. A simplified scenario of the experiments is shown in Figure 15. These experiments are introduced as below in detail.

The experiments will show that, with the help of HINT, users can gather information of remote sensor nodes, obtain network packets with precise timestamps, evaluate network performance parameters and so on.

There are three levels for HINT to observe sensor networks (see Figure 16). The fundamental level is to acquire the raw internal signals inside sensor nodes. The digital signals between the microcontroller and the RF transceiver inside sensor nodes are critical to obtain the radio packet information. The packet level can be achieved by parsing the raw signals. Furthermore, the network behavior could be reconstructed with all the packet information and then the network performance parameters are evaluated, for example, packet delay and loss ratio.

5.1. Raw Signal Level. It is fundamental for HINT to acquire the internal signals accurately inside a sensor node. Hence two experiments are carried out to verify the analog and digital signals acquisition by comparing with the outputs of oscilloscope and logic analyzer.

5.1.1. Digital Signals. The digital signals between the microcontroller and the RF transceiver consist of Start of Frame Delimiter (SFD), interrupt (INT), SPI bus and so on, which are critical for HINT to obtain the network packet information. Thus the correct acquisition of these signals will be proved in this experiment.

All the sensor nodes send packets periodically. We use HINT to acquire these internal digital signals of sensor nodes and demonstrate the result in the graphics user interface. At the meantime, these signals are observed by a logic analyzer LA1016 (see Figure 17). Both HINT and the logic analyzer support SPI protocol parsing. Comparing the output of HINT with those of the logic analyzer, we conclude that HINT can correctly gather the internal digital logic signals (including SPI communication protocol) inside sensor nodes.

5.1.2. Analog Signals. The energy consumption is a key parameter for wireless sensor networks. In HINT, the energy consumption parameter is deduced from the measurement of the electric current at power supply wire of the sensor node. The electric current is a typical analog signal. The correct acquisition of this signal will be proved in this experiment.

All the sensor nodes change their status periodically. The sensor nodes firstly stay at idle state with radio transceiver and all LEDs turned off and therefore the power consumption is very low. After 100 ms, the sensor nodes turn on radio transceiver and the power consumption increases. Then after another 100 ms, the sensor nodes turn on all LEDs and the power consumption increases more. Finally the sensor nodes enter the first idle state after 100 ms. Therefore the supply current will also change periodically.

We use HINT to acquire power supply analog signal of a sensor node and demonstrate them in the graphics user interface. At the meantime, the signal is observed by an oscilloscope, of which the probe connects to output of the current-sense amplifier MAX4173. The output waveforms are both shown in Figure 18.

Comparing the output of HINT with those of the oscilloscope, we conclude that HINT can correctly capture the current consumption of the node. Such conclusion also applies to other analog signals on the sensor nodes.

The energy consumption is a key parameter for wireless sensor networks. If the power supply voltage is a known constant V (3 V for TelosB/ZiNT nodes), the energy consumption E can be obtained by accumulating the product of the voltage, the elapsed time and the sampled value of power

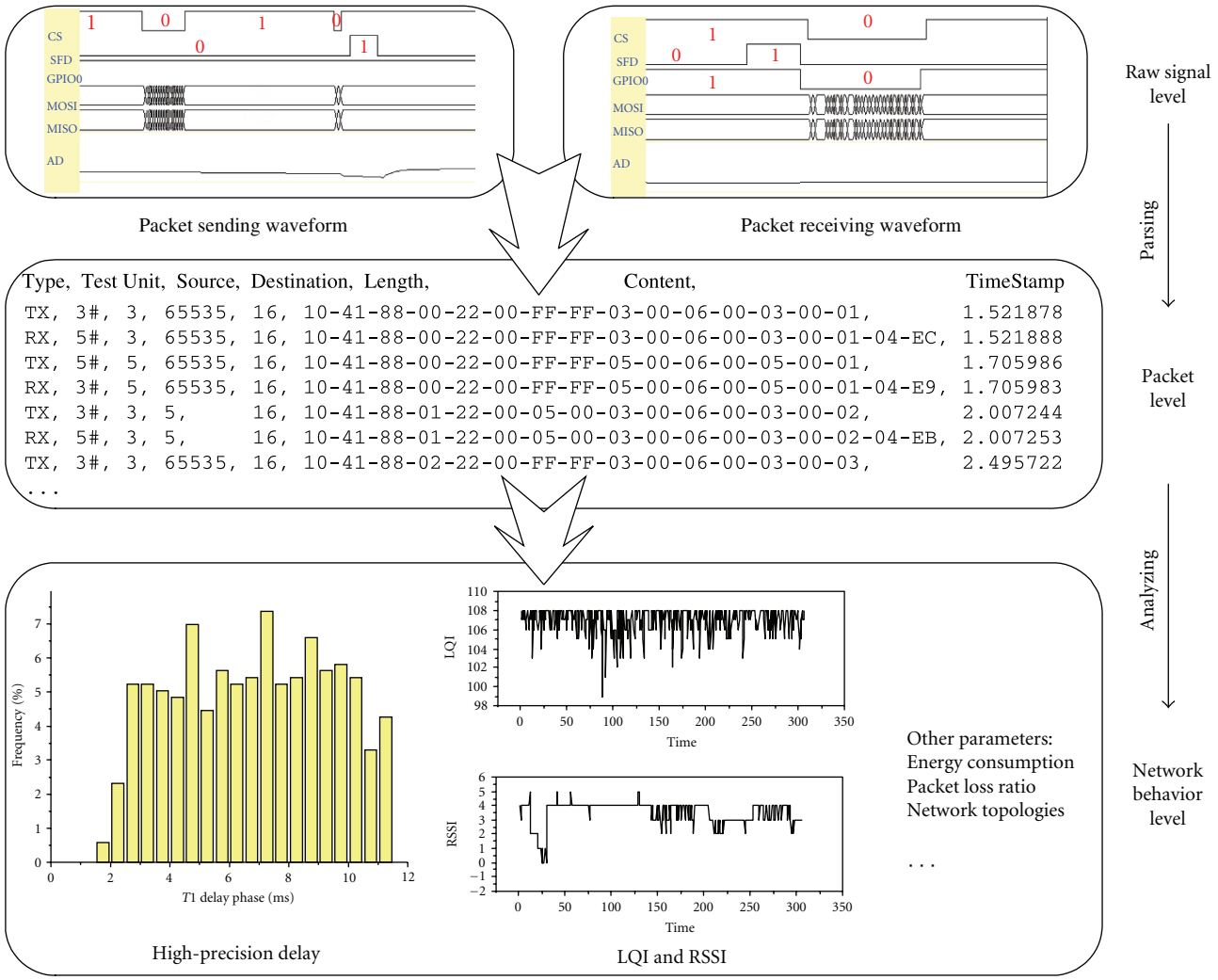


Figure 16: Feature levels of HINT.

supply current I. Here P denotes the power consumption of the sensor node.

$$E = \int P(t)dt = \int V \cdot I(t)dt = V \int I(t)dt \approx V \sum I_i \Delta t_i. \quad (1)$$

5.2. Packet Level

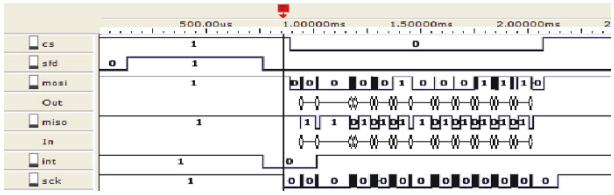
5.2.1. RF Chip Configuration Parsing. The RF transceiver chip of ZiNT node is CC2420 manufactured by Texas Instruments (TI). When the sensor node is turned on, the microcontroller will initialize the RF transceiver via signals including VREG_EN and SPI. CC2420 includes a low drop-out voltage regulator. The voltage regulator is enabled using the active high-voltage regulator enable pin VREG_EN. Next microcontroller will initialize the registers of CC2420 via SPI wires in order to set frequency, Output Power Amplifier (PA) level, frame types, MAC address, work modes, and so forth.

By passively monitoring these signals, HINT is able to hear and understand what the microcontroller talks to the RF transceiver CC2420. Technically, this progress is known as data parsing.

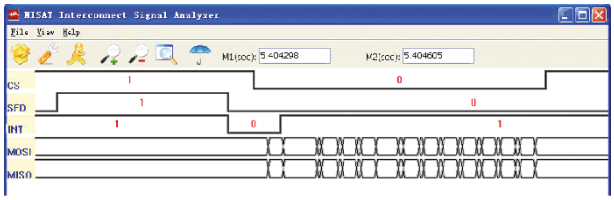
An initialization byte sequence send by the microcontroller via SPI bus captured by HINT is something like "1D-00-18-01-1C-02-7F-18-41-AB-11-2A-E2-17-2A-56-1D-00-00-E8-80-22-00-05-00" in hexadecimal format. For example, the byte slice "18-41-AB" will store value 0x41AB to register 0x18 (Frequency Synthesizer Control and Status Register), which means to set the radio frequency to 2048 + 0x01AB = 2475 MHz. Again, the subsequence "E8-80-22-00-05-00" sets PAN ID to 0x0022 and node address to 0x0005. These analytical results totally correspond the setting in the TinyOS modules running on sensor nodes.

5.2.2. Packet Parsing. The datasheet of CC2420 describes the technological processes to send and receive wireless packets in detail. The core technology is the finite state machine (FSM) in CC2420.

The following steps are necessary to send a packet. First, the microcontroller enables CS, writes the packet to

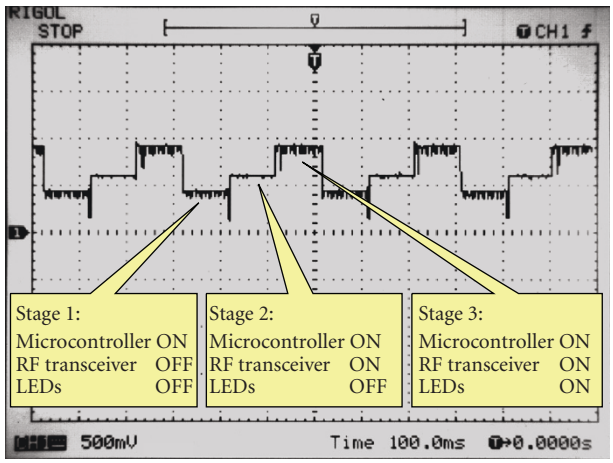


(a) Signals capturde by Logic Analyzer LA1016

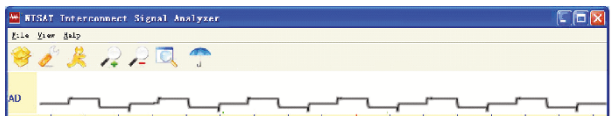


(b) Signals captured by HINT

Figure 17: Comparison of logic signal acquisition.



(a) Signal captured by an oscilloscope



(b) Signal captured by HINT

Figure 18: Comparison of analog signal acquisition.

TXFIFO (128 bytes transmit FIFO) via SPI, and disables CS. Then the microcontroller enables CS again, sends STXON or STXONCCA command via SPI, and disables CS. Finally, once the packet is sent on air, the SFD pin goes active when the start of frame delimiter (SFD) field has been transmitted. The sending progress is demonstrated with a partial screenshot of HINT software in Figure 19.

Accordingly, the following steps are necessary to receive a packet. First, CC2420 receives a packet on air and stores it in RXFIFO (128 bytes receive FIFO). The SFD pin goes active after the frame delimiter has been completely received. Then CC2420 informs the microcontroller by an interrupt signal on the INT (GPIO0) pin. Finally, the microcontroller enables

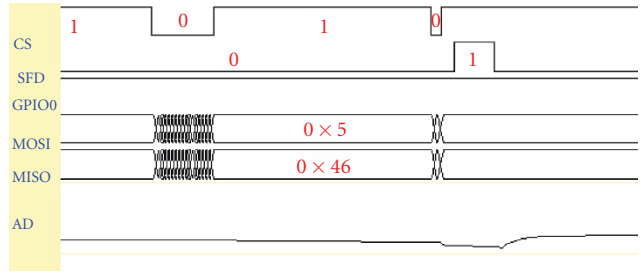


Figure 19: Process of packet sending.

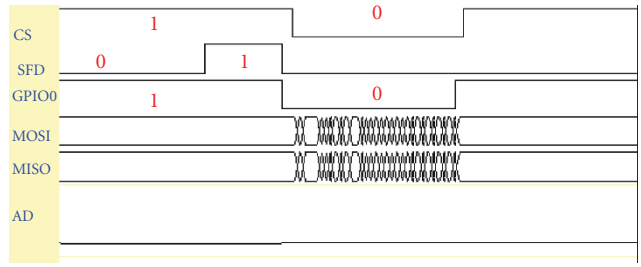


Figure 20: Process of packet receiving.

CS, reads the packet from RXFIFO via SPI, and disables CS.(see Figure 20)

HINT is able to parse what the microcontroller talks with the RF transceiver CC2420. Thus for all packets either sent or received, HINT can obtain the corresponding complete information including packet contents and precise timestamps. We carried out a simple packet send-receive experiment on the test environment and use HINT to capture the packets. A typical byte sequence which MCU write to TXFIFO to send packet captured by HINT is something like "10-41-88-01-22-00-05-00-03-00-06-00-03-00-02" in hexadecimal format, which means 16 (0x10, the 1st byte) bytes in total length, source address 5 (0x0005, the 7-8th bytes), destination address 5 (0x0003, the 9-10th bytes), 6 (0x06, the 11th byte) bytes for payload (4 bytes data "00-03-00-02" and 2 extra CRC bytes).

The typical byte sequences which MCU read from RXFIFO to receive packet captured by HINT are similar to those of TXFIFO, except that there are 2 extra bytes at the end of each received packets, that is, Link Quality Indication (LQI) and Receive Signal Strength Indicator (RSSI).

Comparing the output of HINT with the application predesigned running on the sensor nodes, we conclude that HINT can correctly parse for network packets. Similarly, the conclusion above also applies to other sensor node status capture by monitoring inter-connect signals among the microcontroller and sensors, serial port, and expansion slot.

Furthermore, the precision of timestamps in HINT depends on the logic function of FPGA and the frequency of crystal oscillator. In our implementation, the precision of timestamp is about tens of nanoseconds, which is far superior to the existing testbed.

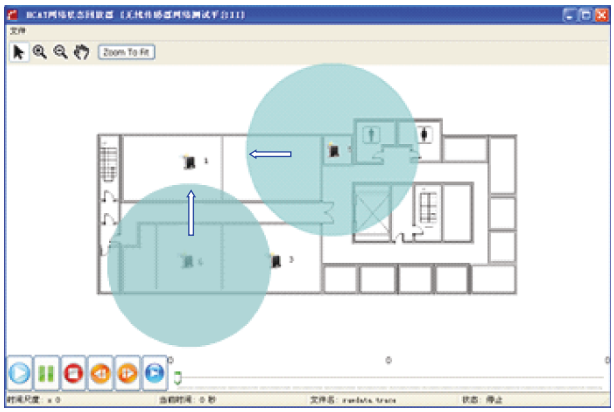


Figure 21: Animation of the network behavior history.

5.3. Network Behavior Level. Since all test data are gathered in the testbed server, HINT can effectively reconstruct the network behavior and evaluate performance parameters of wireless sensor networks with centralized analysis.

All test data from different test units are firstly synchronized with the time synchronization algorithm. Then the packet information for different nodes are corresponding to the same time basis. The information of any nodes send or receive any packets at anytime are easily deduced. HINT can then depict the full spatiotemporal network actions in an animation-like manner, as shown in Figure 21.

The precision of the time synchronization in HINT is less than 10 us. The interval of a packet on the air for wireless sensor networks is about hundreds of microseconds. For example, a 20-byte packet consumes 640 us on the 250 kbps ZigBee radio channel. Therefore HINT knows whether two or more packets collides on the air. In Figure 21 it is shown that two nodes are sending packets to the same destination almost simultaneously. Such feature is very helpful for the research on the MAC protocols of wireless sensor networks.

With the reconstructed network behavior information, the network performance parameters also can be deduced. Several typical parameters, link quality, the single hop delay and throughput, the network topology and the route delay will be introduced as examples.

5.3.1. Link Qualities. With HINT, we can also analyze the link quality between two sensor nodes on the basis of LQI and RSSI field at the end of CC2420 received packets. In this experiment, the typical fluctuation of link quality within 300 seconds is shown in Figure 22.

5.3.2. Single Hop Delay and Throughput. Delay is an important family of network performance parameters. In this experiment, only single-hop packet delays in the sending and or receiving cycles are studied, but the measurement mechanism is extendible.

Single-hop packet delay is defined as the interval between the start of packet sending on the sender and the end of packet receiving on the receiver. HINT can calculate the delays with the timestamp gathered from both sender

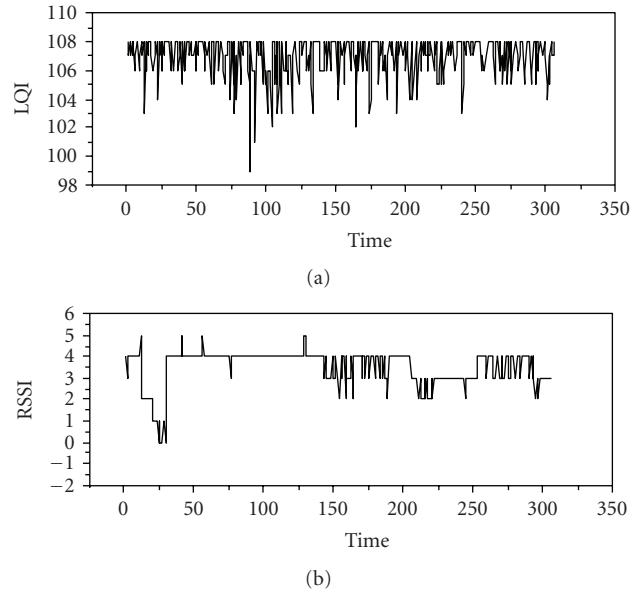


Figure 22: Link quality versus time.

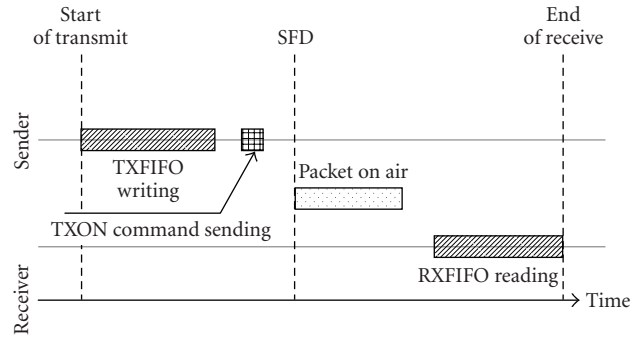


Figure 23: Single-hop delay model for sensor nodes.

Table 3: Single-hop delays for sensor nodes.

Phase	Mean (ms)	Std (ms)
T1: from start of transmit to SFD	7.0	2.9
T2: from SFD to end of receive	6.5	2.6
Total single-hop delays	13.5	5.5

and receiver sensor nodes, on condition that all test units in HINT are synchronized. Furthermore, HINT also can achieve delays for fine-grained phases, such as the elapsed time for TXFIFO writing (see Figure 23).

The default MAC protocol in TinyOS 2.0 is adopted in the experiment. The delays obtained by this experiment are listed in Table 3, here the total packet length is 16 bytes. The radio channel capacity for ZigBee is 250 kbps. However, the sender only send one packet in average 7.0 ms with the default MAC protocol. The actual throughput is about 18.3 kps, which is only 1/13 of the channel capacity. Moreover the sender should send one packet about every 13.5 ms if it wait the receiver finishing the receiving process and the throughput is 9.5 kbps under such condition.

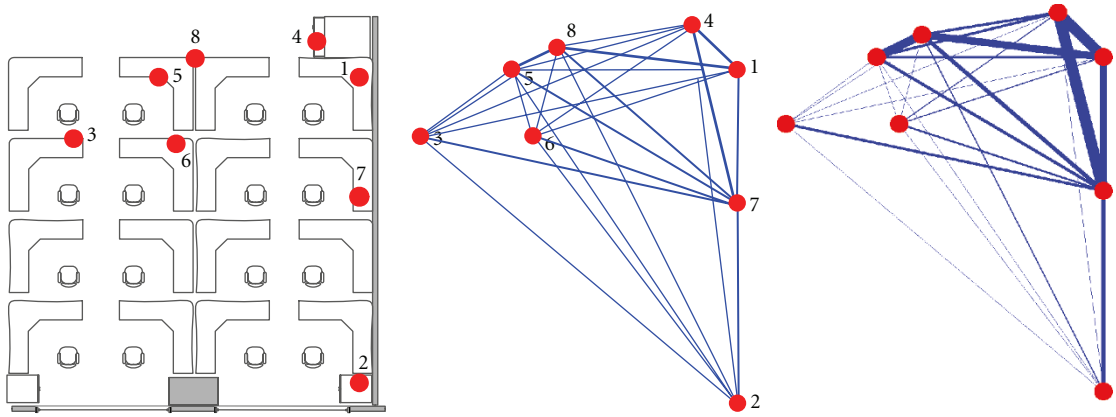


Figure 24: The experiment scenario and link qualities of CTP.

The cycles for FIFO reading and writing and the interval of MAC backoff make the actual throughput far less than the channel capacity. It accords with the existing experiences. However, HINT provides more accurate data about each delay stages and it is necessary for the future studies.

5.3.3. Network Topology and Route Delay. The Collection Tree Protocol (CTP) is a typical route protocol for wireless sensor networks. In this experiment, the CTP route protocol is deployed in a number of sensor nodes to form a multihop adhoc sensor network. Eight nodes are placed in the laboratory and the first node (with node identifier 1) is the sink of the network. The experiment scenario is shown in the left part of Figure 24.

The network performance parameters can be deduced from the test data. As an example, the qualities for all wireless links are illustrated in the right part. The thickness of the lines represent the link quality. Wider lines mean better links whereas thinner lines mean worse links.

The route path and route delay can be obtained since all packet information are collected in the HINT. The fourth node is near the sink. The route path from the fourth node to the sink is almost 1 hop. However the second node is far from the sink. The route path from the second node to the sink changes dynamically among 2 to 3 hops. The Figure 25 shows the route delay from the source nodes to the sink. The route delay for the fourth node is about 10 ms. The route delay for the second node changes from 20 ms to 1 second, which depends on the multihop queue delay and multiple retransmission.

5.4. Other Features. HINT also provides a series of interesting and valuable features in addition to those mentioned above.

5.4.1. Offline Analysis. In HINT, the center test server is able to store data gathered from all nodes to detailed trace files, which are similar to those generated by simulation tools. The trace files consist of full information of network events,

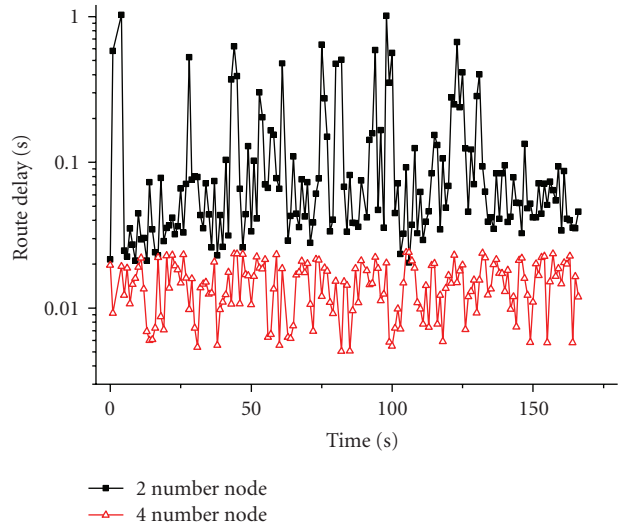


Figure 25: The route delay of CTP.

in which the source address, destination address, length, content and timestamps of related phases are present for each packet (see Figure 26).

With these trace files, users are able to replay the network events, validate network protocols and analyze the stored data in the future. Thus HINT help users to study wireless sensor network experimentally but in an easily and flexible manner, just like those in network simulations.

5.4.2. Remote Control. In the test board there is a controllable switch for the power supply to the node, which is used to turn on or off the electric power of the node. We can control whether the node is powered either on or off from HINT center server remotely.

5.4.3. Remote Test and Monitor. The HINT test board affords peripheral sniffing to almost all of the interconnect signals in the nodes, including GPIO pins of the microcontroller linked

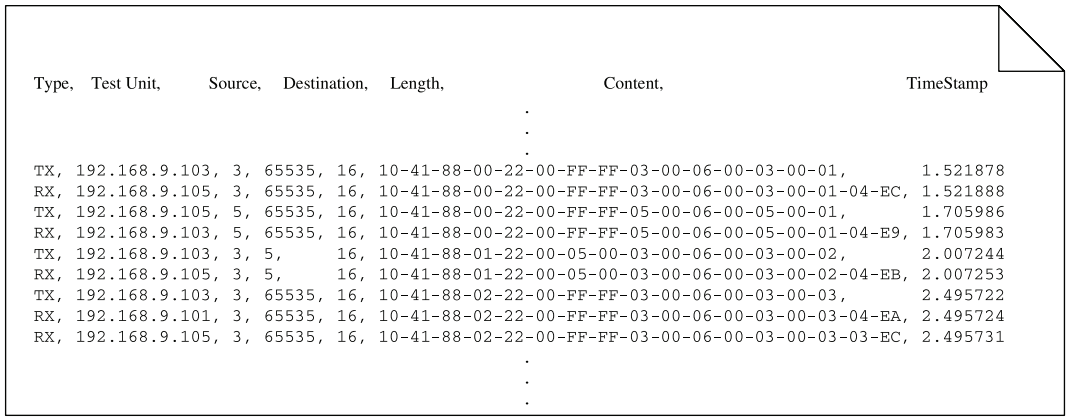


Figure 26: Trace file format of HINT.

to the LED diodes and the serial port. Thus we are able to know the LED status and data transferred via the serial port of sensor nodes at the remote center server. Therefore HINT provides features of the existing testbeds. Users can append their debugging code into the software running on sensor nodes and execute testing remotely.

5.4.4. Remote Programming and Debugging. The JTAG interface of the microcontroller in the node is also connected to the HINT test board. Thus theoretically we can download binary code to any sensor node, run the code with predefined breakpoints or debug step by step.

6. Discussion

Two advanced characteristics should be emphasized for HINT. The first characteristic is that HINT is a nonintrusive testbed, that is, disturb-free on the runtime behavior of wireless sensor networks. The second is that HINT offers us a mechanism to precisely observe internal status of sensor nodes.

6.1. NonIntrusive and Transparency. In the test platforms such as Kansei or MoteLab, users must add software modules into the microcontrollers of the sensor nodes in order to send out test data via extra wired or wireless networks. The computing resource, the storage resource and the node status are inevitably affected for the sensor networks under testing. In the test platforms like MoteWorks, the test data are even transferred over the links of wireless sensor network itself. The wireless communications and the network traffics are awfully disturbed. Therefore the existing test platforms have more or less side effects on the runtime behavior of wireless sensor network, and all of them are not transparent to the applications. For the resource-constrained wireless sensor networks, any intrusive testing action should be avoid to obtain the real spontaneous behavior.

The test mechanism of HINT is to passively probe the chip-level interconnected signals inside the sensor node. The chips, including the microcontroller and the RF transceiver, are never aware of the existence of the test board. HINT does not disturb the spontaneous operations of sensor nodes. Moreover, the test data are transferred over additional networks and consequently HINT does not disturb the wireless communications of the wireless sensor network at all. Therefore HINT has no size effects on the spontaneous network behavior and it is a nonintrusive testbed for wireless sensor networks.

Furthermore, the sensor nodes run in a transparent way and are never aware of the existence of HINT. Users do not need to append or modify a bit of the binary codes for the testing purpose. Hence HINT is also a transparent testbed. Consequently, HINT is especially suitable for black box testing or product testing, where the source codes of sensor nodes are not available for the reasons such as copy rights or security.

6.2. High Accuracy. Although the additional network instruments help users to capture and analyze packets on the air precisely without any side effects on the wireless sensor network, they are not aware of what happen inside sensor nodes.

For Kansei, MoteLab and MoteWorks platform, the internal status of nodes could be observed if the microcontroller sends the status data out. However, these platforms cannot obtain precise and complete information about node status limited by some factors including the computing and storage capacity of the microcontroller, the precision of the crystal oscillator on the nodes, unreliable network links, and the performance of time synchronization protocol of the wireless sensor network itself. For instance, in the Kansei testbed the timestamp accuracy are affected by the performance of the XSM nodes, the interrupt response delay on the StarGate nodes which embedded Linux operating system without hardware real-time, and the time synchronization between the StarGate and the center server over Ethernet and WiFi links. The timestamp accuracy can hardly reach nanosecond level. Moreover, these testbeds lack in measurement the energy consumption of sensor nodes.

Owing to the auxiliary test boards with high-performance CPU and FPGA, HINT can obtain precise information of the internal status of nodes. For example, the

time precision of HINT can reach about tens of nanoseconds with high-frequency and high-precision crystal oscillator. Again, FPGA can accumulate energy consumption by sampling power supply current at 10M SPS (samples per second). The hardware-level real-time capturing and sampling are guaranteed by the FPGA.

7. Conclusion and Future Work

It becomes increasingly important to obtain the accurate and spontaneous runtime data which represent the network behavior for further studies on the wireless sensor network. However the test mechanisms nowadays cannot appropriately match such requirements.

We proposed a novel test mechanism that the internal chip-level signals inside sensor nodes are passively probed in order to access the runtime network data in a nonintrusive manner. Subsequently we designed and implemented the testbed HINT. In detail we introduced the design, implementation and the experiment studies of HINT.

We showed that, with the help of HINT, users can gather information of internal signals on the remote sensor nodes, measure the energy consumption of sensor nodes, parse the interconnect signal for the radio packets, obtain precise timestamps of events for fine-grained phases, evaluate the network performance parameters, program and debug the sensor nodes remotely, store the runtime data to trace files, and so forth. Most of these features are disturb-free and transparent to the applications. HINT provides users real and accurate information on the runtime data to represent the spontaneous network behavior of wireless sensor networks.

HINT is a nonintrusive testbed. HINT provides users high-accuracy information on the network behavior. HINT also supports the test mechanisms of the existing testbeds. Thus HINT is an upstanding testbed solution for the future fine-grained and experimental studies on the resource-constrained wireless sensor networks.

HINT supports any software platforms and various hardware platforms. In fact HINT cares neither the operating system and software modules running on the sensor nodes, nor the integrated chips on the nodes. HINT only depends on the chip-level interconnected protocols of the chips inside sensor nodes. However the flexibility of FPGA and adaptor still lead to a feasible solution. A series of nodes are already tested in HINT, such as TelosB, Mica2, MicaZ and ZiNT.

Up to now HINT remains a small-scale testbed. But we have already used it in some related research projects. Some interesting phenomenons are observed for future studies. We will keep improving the features and expanding the appliance fields of HINT. A remote web-based service to run the applications and download trace data for any interested users will be also provided.

Acknowledgments

This paper was supported in part by the National Major Project of Fundamental Research of China under Grant no. 2006CB303007, and the National Nature Science Foundation of China under Grants no. 60903211 and 60933011.

References

[1] "NS2: the Network Simulator," http://www.isi.edu/nsnam/ns.

[2] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," in *Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS '98)*, pp. 154–161, May 1998.

[3] A. Varga, "The OMNet++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference (ESM '01)*, Prague, Czech Republic, June 2001.

[4] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 126–137, November 2003.

[5] L. Girod, N. Ramanathan, J. Elson, T. Stathopoulos, M. Lukac, and D. Estrin, "Emstar: a software environment for developing and deploying heterogeneous sensor-actuator networks," *ACM Transactions on Sensor Networks*, vol. 3, no. 3, Article ID 1267061, 13 pages, 2007.

[6] J. Polley, D. Blazakis, J. McGee, D. Rusk, J. S. Baras, and M. Karir, "ATEMU: a fine-grained sensor network simulator," in *Proceedings of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON '04)*, Santa Clara, Calif, USA, October 2004.

[7] S. Park, A. Savvides, and M. B. Srivastava, "SensorSim: a simulation framework for sensor networks," in *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '00)*, pp. 104–111, August 2000.

[8] S. Sundresh, W. Kim, and G. Agha, "SENS: a sensor, environment and network simulator," in *Proceedings of the 37th Annual Simulation Symposium (ANSS '04)*, pp. 221–228, April 2004.

[9] Crossbow, "Moteworks wireless sensor network platform," http://www.xbow.com/.

[10] E. Ertin, A. Arora, R. Ramnath et al., "Kansei: a testbed for sensing at scale," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, pp. 399–406, April 2006.

[11] G. Werner-Allen, P. Swieskowski, and M. Welsh, "MoteLab: a wireless sensor network testbed," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, pp. 483–488, April 2005.

[12] P. Dutta, J. Hui, J. Jeong et al., "Trio: enabling sustainable and scalable outdoor wireless sensor network deployments," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, pp. 407–415, April 2006.

[13] V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "TWIST: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks," in *Proceedings of 2nd International Workshop on Multi-Hop Ad Hoc Networks: From Theory to Reality (REALMAN '06)*, vol. 2006, pp. 63–70, Florence, Italy, May 2006.

[14] IEEE 802. 3af, "Data Terminal Equipment (DTE) Power via Media Dependant Interface (MDI)," IEEE Computer Society, June 2003.

[15] A. Dunkels, "Design and implementation of the LwIP TCP/IP Stack," Tech. Rep., Swedish Institute of Computer Science, February 2001.

[16] Altera Corporation, "Cyclone II Data Sheet," http://www.altera.com/.