Wireless Netw (2016) 22:1199–1220
DOI 10.1007/s11276-015-1033-x

A generic framework for energy evaluation on wireless sensor networks

Nanhao Zhu[1] · **Athanasios V. Vasilakos**[2]

Published online: 4 August 2015

Abstract Due to reliance on batteries, energy consumption has always been of significant concern for sensor node networks. This work presents the design and implementation of a house-build experimental platform, named Energy Management System for Wireless Sensor Networks (EMrise) for the energy management and exploration on wireless sensor networks. Consisting of three parts, the SystemC-based simulation environment of EMrise enables the HW/SW co-simulation for energy evaluation on heterogeneous sensor networks. The hardware platform of EMrise is further designed to facilitate the realistic energy consumption measurement and calibration as well as accurate energy exploration. In the meantime, a generic genetic algorithm based optimization framework of EMrise is also implemented to automatically, quickly and intelligently fine tune hundreds of possible solutions for the given task to find the best suitable energy-aware tradeoffs.

Keywords Wireless sensor networks · Energy consumption · Energy evaluation · Simulation · SystemC · Measurements · Optimization · Genetic algorithm

✉ Nanhao Zhu
nanhaozhu@foxmail.com

Athanasios V. Vasilakos
vasilako@ath.forthnet.gr

[1] CETC Group, GCI Science and Technology Co., Ltd, No. 381, Xingangzhong Road, Haizhu District, 510310 Canton, China

[2] Department of Computer Science, Electrical and Space Engineering, Lulea University of Technology, 97187 Lulea, Sweden

1 Introduction

With the great development in embedded systems and wireless communication technologies, wireless sensor networks (WSNs) have gained worldwide attention and have been developing rapidly in the past decade. Wireless sensor networks are large-scale networks with low-cost, small-size, low-power and limited processing sensor nodes deployed in various kinds of environments. As the basic part of WSNs, a typical sensor node comprises several components: microcontroller, transceiver, sensor and power supply (Fig. 1). By combining these different components into a miniaturized device, these sensor nodes become multi-functional and have been applied to a wide variety of application scenarios: medical and health care [1], environment and ecological monitoring [2], home and building automation [3], industrial monitoring [4] and battlefield [5]. Thus, such highly diverse scenarios impose application-specific requirements on WSN design and distinguish them from conventional networks.

Generally, WSNs are easy to achieve great scalability and a wide range of densities, due to the low cost and small dimensions of sensor nodes. In the meantime, specific sensor network protocols and algorithms with self-organizing capabilities are usually designed and applied to sensor nodes, so the whole sensor network is low-maintenance and tolerant against communication failure and topology changes which could be caused by node malfunction, node mobility or energy depletion of the nodes. Furthermore, sensor nodes can also be deployed in harsh environments with self-organized network to carry out given tasks in unmanned manners.

However, further and potential applications are limited due to inherent WSN disadvantages. For instance, the limited processing ability and low data rate of sensor nodes

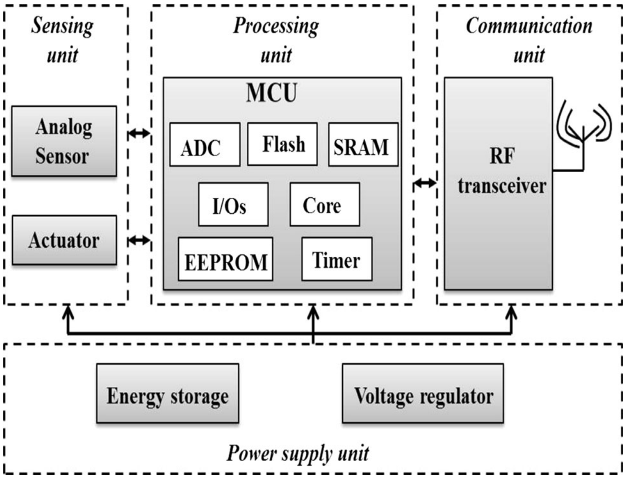


Fig. 1 Sensor node hardware architecture

cannot guarantee high performance in some scenarios especially for real-time applications. Short communication range can cause energy waste and network inefficiency, since multi-hop communications are always required for data transport between source node and sink node. The severe energy constraints also lead to a worse observation: the improvement of data processing ability by using powerful processors are not expected, since the energy will be depleted very quickly and renders the network useless on the given task. Limited energy also means it is not possible to also maintain the operation of multi-hop communications for a long time. Meanwhile, sensor nodes are expected to be used heavily in many remote area applications, where the frequent changing or recharging of batteries is inconvenient.

Hence, an energy-efficient sensor network is necessary to carry out the work in such conditions. Taking energy consumption as the major consideration, while not undermining other network performances, is therefore desired as the priority strategy in the design of current wireless sensor networks.

2 Related works

The emerging field of wireless sensor networks and their applications promises a higher quality of life in many aspects of our daily activities, but the strict energy constrains stated above significantly limit their functionality expansions and possible applications, which makes energy consumption one of the most critical concerns in WSNs. Therefore, energy consumption analysis and evaluation is a critical process in the design and implementation of the energy-efficient sensor network.

In recent years, great efforts which have been made on energy saving management for wireless sensor networks can be grouped into three categories: (1) simulation/emulation based approach for energy consumption calculation and analysis; (2) hardware based method for real-world energy measurement and calibration; (3) optimization based strategy for the exploration of energy-efficient behaviors and configurations.

First of all, as a useful tool for the energy analysis and management on wireless sensor networks, simulation/emulation based method is widely accepted and commonly used, it can provide a more realistic model for the evaluation of energy consumption rather than the oversimplified and idealized hypotheses that are assumed in the mathematics based theoretical models. Although relatively slowly, it offers the tradeoff between accuracy and efficiency. General simulation tools like NS-2 [6], OMNeT++ [7], Prowler [8], written respectively in C++, Java and MATLAB, provide satisfactory efficiency at the early design stages, since the models are usually at a high abstraction level to facilitate the testing and verification of algorithms, protocols and strategies. However, only a relative coarse energy consumption evaluation can thus be expected due to the lack of realistic low level models. Emulation tools like TOSSIN [9], ATEMU [10] and Avrora [11] can compensate this drawback on energy consumption evaluation but at the cost of efficiency, and they are basically limited to specific hardware platforms and operating systems (TOSSIN is used for sensor nodes with TinyOS [12] operating system, both ATEMU and Avrora are used for AVR microcontroller based node platforms).

From the aspect of hardware based method, an increasing number of research works have already focused on energy consumption in real-world sensor nodes. A detailed energy consumption analysis based on the MICAz mote [13] is presented in [14], several benchmarks are used for energy estimation, battery charge effect and battery lifetime are also considered. Authors in [15] present AEON (Accurate Prediction of Power Consumption) and evaluate detailed energy profiling on a MICA2 sensor node [16]. Elaborate energy consumption data on the TelosB mote [17] is provided based on the authors' measuring methodology and power consumption measuring system (PCMS) in [31]. Sensor Node Management Devices (SNMD) [19, 20] is presented with detailed performance parameter configuration for the energy data measurements.

In the view of optimization based energy management strategy, corresponding research works have been done from both hardware and software: (1) From the hardware perspective, better energy efficiency can be achieved by optimizing the power consumption of related hardware components. By employing an ultra-low power based microcontroller (e.g., MSP430 [21]) and configuring it into its five different low power modes, a significant amount of energy can be saved. Recent years, transceivers with some emerging technologies such as Bluetooth low energy technology [22], Ultra-wideband (UWB) [23], and ANT [24] can also provide excellent performance in lifetime improvement. Besides, transceiver with a higher data rate described in [25], has proven to be a very energy-effective choice. From the viewpoint of energy supply, emerging energy harvesting technologies [26] are also highly useful methods for overall energy management and power optimization, since the dependence of battery-driven on sensor motes can be greatly reduced. (2) From the software perspective, the optimization method can be grouped into three categories: the development of new communication protocols (MAC and routing) for optimization, the adoption of energy-aware management methods for optimization, and the configuration/exploration of the optimal set of existing protocols for optimization. Firstly, such as S-MAC [27] and T-MAC [28] protocols are proposed based on the contention-based scheme for collision avoidance and radio-off during the sleep periods for energy saving. Secondly, the uses of strategies for energy management optimization include in-network processing, data aggregation, and cross-layer related optimization. For the final method, by exploring different parameter configurations of a beaconless IEEE 802.15.4 network [29] via simulation, the goal of achieving energy saving and better latency performance is realized in [25].

3 Motivation

Despite the tradeoff can be achieved between accuracy and efficiency with simulation/emulation based method for energy management and evaluation on wireless sensor networks as mentioned above, without hardware-based instruction level model for energy evaluation only unrealistic result can be acquired by simulation method. Meanwhile the inefficient and platform-specific based emulation method cannot be considered as a better solution. Since SystemC [30] is a system-level and hardware description language offering support for HW/SW co-simulation, concurrency, Modelling at different abstraction levels as well as other flexible and diverse Modelling advantages, it is therefore regarded as a good alternative to model and simulate WSNs for accurate, flexible and quick energy evaluation.

Although measurements from large-scale of sensor nodes are hard to acquire and sometimes these nodes are placed in harsh and inaccessible areas, the hardware based measurements from real world testbed nodes are still considered necessary and significant since they offer a more realistic environment to the real deployment and are able to calibrate and improve the accuracy of energy evaluation. Research works mentioned above can only provide simple energy measurements on specific node platforms and with costly measurement equipment (high-performance oscilloscope [14, 15], acquisition cards [31, 32]) on the sensor nodes to sample the varying low currents and voltages, which limit the scale of deployment for energy measuring beyond the lab environment. Therefore, the building of a cost-effective, multi-functional and measurement reliable hardware platform is currently required.

For the hardware based energy optimization methods mentioned above, such as the use of ultra-low power devices, emerging low power radio technologies, higher data rate as well as the energy harvesting method can be very useful energy-efficient solutions. However, the improvement on these hardware metrics always takes some time, such that it is difficult to satisfy rapidly evolving applications and requirements. Relatively speaking, efforts on software aspects can provide a quicker alternative for performance optimization on given scenarios in terms of energy and a variety of other metrics. In spite of the efficiency provided by software based simulation, the exhaustive and full simulation process on all the possible configurations of the protocol are not only time-consuming, but also unnecessary most of the time. Thus, an approach that can effectively and automatically select an appropriate protocol parameter setting out of hundreds or thousands of possible options is urgently needed. With the fast and intelligent search ability, Genetic Algorithms (GAs) [33] enables the fine tuning of parameter space of the protocol to find the most suitable configuration for energy saving and management.

4 The design and implementation of EMrise

Based on the limitations and suggestions of the three aspects mentioned above, our EMrise (**E**nergy **M**anagement System fo**r** **Wi**reless Sen**s**or N**e**tworks) is proposed in this chapter. Section 4.1 describes the design and implementation of EMrise_SS (EMrise_**S**imulation **S**ystem) which is a SystemC-based simulation environment at the system-level and transaction-level, and the energy performance of several different sensor motes under various case scenarios are also investigated by EMrise_SS in this section. Section 4.2 presents EMrise_MS (EMrise_**M**easurement **S**ystem), a cost-effective and flexible hardware based energy measurement platform for the calibration and management of realistic energy data. Section 4.3 focuses on the introduction and design of EMrise_OpS, a generic and genetic algorithm based optimization framework for the optimized energy management on WSNs. Finally, Sect. 5 concludes this book chapter and discusses the possible future works.

4.1 EMrise_SS (EMrise Simulation System)

EMrise_SS, also known as iWEEP_SW [36], is a system-level, transaction-level and energy-aware SystemC based generic sensor network simulation which implements several popular commercial off-the-shelf (COTS) low data rate based sensor node models including the Telos series (e.g., TelosB [17], Tmote Sky [34], Shimmer node [35]), the MICA series (e.g., MICA2 [16], MICAz [13]), and a house-built high data rate and ultra-low power testbed node iHop@Node [37] (PIC16F microcontroller with nRF24L01+ transceiver, maximum data rate up to 2 Mbps). Four MAC protocols are integrated and modeled in EMrise_SS which are unslotted and slotted CSMA/CA protocols in IEEE 802.15.4 [29], as well as Enhanced ShockBurst (ESB) and ShockBurst (SB) protocols [38] embedded in the high data rate iHop@Node.

4.1.1 The framework of EMrise_SS

EMrise_SS is composed of SystemC defined components, ports, channels, interfaces and connections shown in Fig. 2.

Components correspond to basic hardware modules such as microcontroller, transceiver, sensor and battery as well as various kinds of peripherals. Each component in EMrise_SS is modeled as an individual module of SystemC. In these hardware component models, the sensor is modeled to generate sensed data periodically from physical or environment conditions. The microcontroller (MCU) collects these analog data and converts them into digital signals by the built-in Analog to Digital Converter (ADC). The converted digital data are then sent to the transceiver according to different application scenarios and communication strategies. The transceiver (Radio) takes the responsibility of over-the-air packet transmission and reception, as well as clear channel assessment (CCA). The energy module functions as a power consumption monitor which can track and assess the energy consumption of other hardware components at different abstraction levels from bit-accurate/cycle-accurate to transaction-accurate based on designer's requirements. In the timer module, several sub-timer components can be defined for different purposes such as timer for sample interval, timer for each transmission interval, protocol related timer, etc. For some other peripherals such as the memory model and UART model inside the MCU, their Modelling levels are optional, which depend on the detailed design requirements. In addition, the network module is also modeled as a component since it is inherited from the TLM-based SystemC Network Simulation Library (SCNSL) [39], which is used to establish network topology, implement packet transmission and handle network collisions.

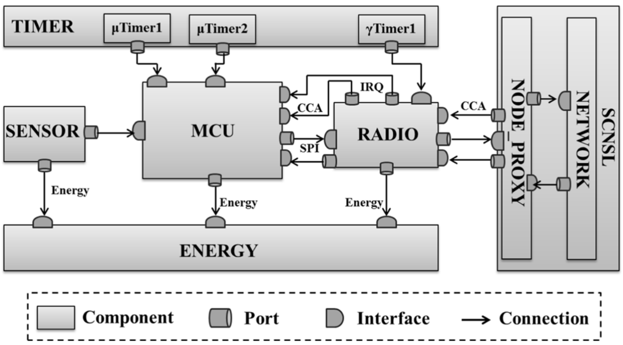


Fig. 2 EMrise_SS framework

Channels are responsible for the communications between different components, such as the internal bus of the microcontroller connecting CPU core and peripherals, the Serial Peripheral Interface (SPI) bus connecting microcontroller and transceiver. The channels' implementations are not mandatory and they are always abstracted to simplify the development process according to the designer's requirements. Each component and channel can have one or more interfaces to distinguish and specify a set of functions (or transactions), which are used to interact with the relevant components or channels. Like interfaces, multiple ports can be defined by components and channels, and they are utilized to specify the types of the interfaces, so each function-specific port can be connected to the related component or channel as long as the corresponding interface is implemented by that component or channel [40].

In addition, in EMrise_SS framework, the whole sensor network system maintains two data items for packet frames. One is a generic packet format for Enhanced ShockBurst (ESB)/ShockBurst (SB) protocol-based sensor node network and the other is for IEEE 802.15.4 based sensor network, which are presented in Figs. 3 and 4 respectively.

4.1.2 Microcontroller modelling

Detailed model of the microcontroller is presented in Fig. 5.

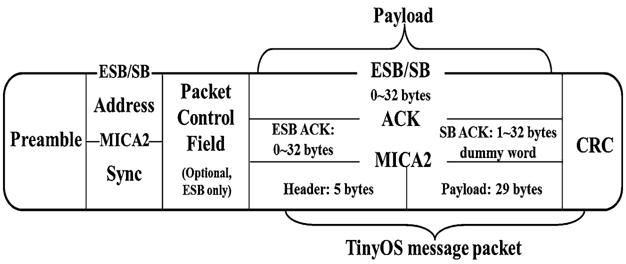


Fig. 3 Generic packet format

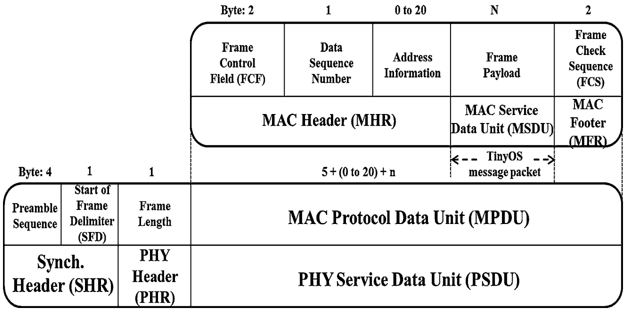
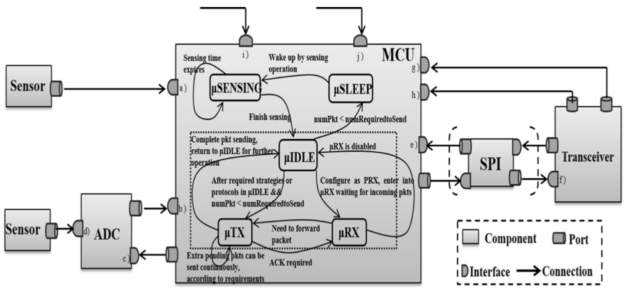


Fig. 4 IEEE 802.15.4 packet frame format

The microcontroller component defines the *sensordata_read_if* interface consisting of a single *Sensor-Data-Read* transaction (or function) which allows the microcontroller to first read data from the target sensor module and then convert these analog data into digital data by the built-in Analog to Digital Converter (ADC). In some application scenarios, if an external ADC is needed for a more accurate and quicker conversion, the microcontroller defines another *extADC_read_if* interface. This is maintained by an *extADC_read* transaction (function) that emulates the process whereby the microcontroller reads converted ADC values through its digital I/O port pins or through SPI pins from an external ADC chip. The microcontroller component also links ADC's *extADC_config_if* interface by the defined output port on the microcontroller side, which is composed of a single *extADC_config* transaction to receive configuration information from microcontroller. Moreover, the *extADC_readSensor_if* interface and *extADC_readSensor* transaction defined on the external ADC are used to acquire sampled sensor data. Considering that cycle-by-cycle based energy consumption on the sensor node would be evaluated in later work, the above interfaces and transactions can be easily extended to incorporate detailed communication mechanism and device driver code implementation. Another two transactions named *IRQdetect* and *CCAchecking*, which are defined by *IRQdetect_if* and *CCAchecking_if*, is responsible for interrupts detecting from transceiver and clear channel assessment processes respectively. Several timer transactions can be defined for varying functions such as the examples that are shown in Fig. 5. In addition, interactions between the microcontroller and transceiver are handled via SPI, considering that SPI communications between the microcontroller and transceiver are bidirectional. Hence, the SPI communication interface defines two transactions named *SPIdata_µrecv* and *SPIdata_µsend* on the microcontroller and transceiver components respectively. On the other hand, the operation of the microcontroller is performed with a finite state machine (FSM) in approximate-timed manner. A generic microcontroller model shown in Fig. 5 acts as the basic Modelling for other specific microcontrollers.


Sensor
Sensor
ADC
MCU
Sensing time expires
i)
Wake up by sensing operation
j)
µSENSING
µSLEEP
Finish sensing
numPkt < numRequiredtoSend
Complete pkt sending, return to µIDLE for further operation
µIDLE
µRX is disabled
After required strategies or protocols in µIDLE && numPkt < numRequiredtoSend
Configure as PRX, enter into µRX waiting for incoming pkts
µTX
Need to forward packet
µRX
Extra pending pkts can be sent continuously, according to requirements
ACK required
a)
b)
c)
d)
e)
f)
g)
h)
SPI
Transceiver
Component
Port
Interface
Connection

EMrise_SS MCU Interfaces and Transactions

a) ***Sensordata_read_if***
void MCU_xxx : : Sensordata_read (sensedPkt, size)

b) ***extADC_read_if***
void MCU_xxx : : extADC_read (ADCPkt, size)

c) ***extADC_config_if***
void ADC_xxx : : extADC_config (Commands, size)

d) ***extADC_readSensor_if***
void ADC_xxx : : extADC_readSensor (sensedPkt, size)

e) ***SPIdata_µrecv_if***
void MCU_xxx : : SPIdata_µrecv (Pkts, size)

f) ***SPIdata_µsend_if***
void Tran_xxx : : SPIdata_µsend (Comm+Pkts, size)

g) ***IRQdetect_if***
void MCU_xxx : : IRQdetect (type)

h) ***CCAchecking_if***
void MCU_xxx : : CCAchecking ()

i) ***SampleInterval_if***
void MCU_xxx : : SampleInterval (time)

j) ***Protocolrelated_if***
void MCU_xxx : : Protocolrelated (time)

Fig. 5 EMrise_SS MCU model

4.1.2.1 Modelling of PIC16 Developed by Microchip, MiWi [41] as a wireless networking protocol stack is especially designed for PIC microcontroller families from PIC16 to dsPIC33. Currently, only the non-beacon mode of IEEE 802.15.4 is implemented as MiWi's MAC layer, and our PIC16F microcontroller is modeled following MiWi's unslotted algorithm as a FSM in previously mentioned µIDLE, µTX and µRX states, and divides these three states into many sub-states, which is shown in Fig. 6.

In Fig. 6, the microcontroller first performs a random backoff duration and checks the channel status. If the channel is detected to be busy, and the number of CCA attempts is larger than the protocol parameter *macMaxCSMABackoffs*, then a Channel Access Failure (CAF) is reported. If on the other hand the number of CCA attempts is smaller than *macMaxCSMABackoffs*, the microcontroller will go back for a new round of random backoff process. When the channel is indicated as free by CCA, the microcontroller will trigger over the air transmission to send a packet to the transceiver. After the transmission of an ACK required data packet, the protocol will make the microcontroller wait within a fixed time period (0.864 ms or 54 symbols) for the ACK frame confirmation. If ACK is received in time, this transmission is regarded as successful. Otherwise, the packet retransmission process will start. A Collision Failure (CF) occurs only after failure to receive ACK frame *macMaxFrameRetries* times, which might be caused by collisions of data packets or collisions between data packets and the ACK frame. In addition, as long as there are pending data packets in the

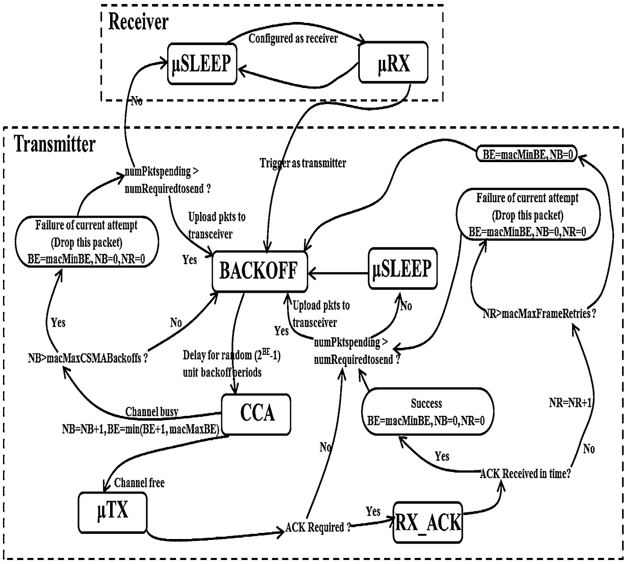


Fig. 6 Unslotted CSMA/CA algorithm model

MCU, they will be uploaded to the transceiver immediately, once the transmission process is over (no matter whether a success or a failure), which is to guarantee the timeliness and reliability of data transmission.

4.1.2.2 Modelling of MSP430 Compared with PIC16F, a powerful 16-bit MSP430 microcontroller is widely used in some current COTS (commercial-off-the-shelf) sensor motes such as the previous mentioned Telos, Tmote Sky and Shimmer nodes. A complete MAC protocol stack of IEEE 802.15.4 such as TKN15.4 [42] can be implemented in the TinyOS operating system on these popular sensor motes, which integrates both slotted and unslotted CSMA/CA algorithms (used in beacon-enable and non-beacon enable modes respectively). Hence, the MSP430 microcontroller component in EMrise_SS is modeled as a slotted and unslotted compatible FSM in the µIDLE, µTX and µRX states shown in Fig. 7.

Compared with the unslotted algorithm, the slotted CSMA/CA algorithm needs to be accurately timed, where a beacon packet broadcast by the coordinator at the beginning of the defined superframe [29] time period is used for synchronization between coordinator and sensor motes. After the reception of the beacon packet, sensor motes with enough packets will start a slotted-based CSMA/CA algorithm. This requires that the backoff period boundaries should be aligned with the superframe slot boundaries (simulation logs starting with '#####' in Fig. 22 show that the backoff period boundaries are accurately aligned with slot boundaries). Note that before each new backoff period there is a time evaluation process of checking whether remaining CSMA/CA can be undertaken before the end of the contention access period (CAP). If the number of backoff periods is less than the remaining number of backoff periods, the slotted algorithm will continue this backoff delay under the condition that two CCA analyses, the frame transmission and acknowledgement can be completed before the end of the CAP. Otherwise, it must wait until the start of the next superframe CAP. If the number of backoff periods is greater than the number of remaining backoff periods, the slotted algorithm shall first pause the backoff countdown at the end of CAP and then resume it at the beginning of the CAP of next superframe. Except for the time evaluation mechanism, another difference between the slotted and unslotted algorithms is that two times of idle channel checks before packet transmission are required for the slotted algorithm.

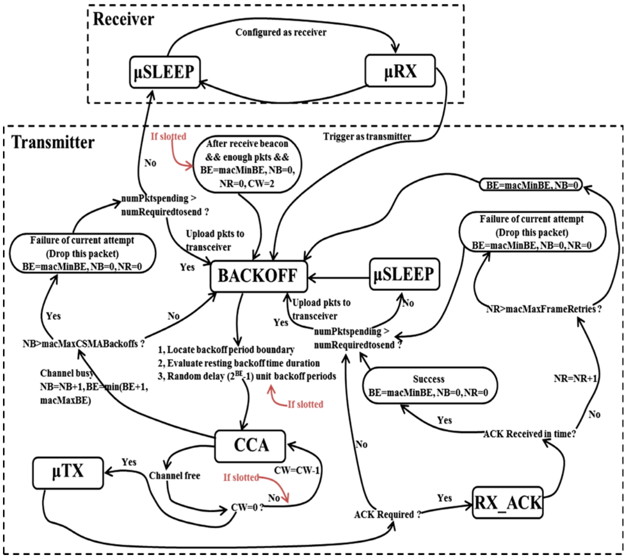


Fig. 7 Slotted and unslotted CSMA/CA model

4.1.3 Transceiver modelling

The behavior of the RF transceiver is handled via two interfaces. As mentioned previously, the *SPIdata_µsend_if* interface is defined on the transceiver component and consists of a single transaction *SPIdata_µsend*, which manages events from the microcontroller such as receiving configuration commands and payload data. The *Netdata_rInput_if* interface defines the transaction *Netdata_rInput* that allows the transceiver to detect network related events such as the validation of incoming packets from the RF channel and the transaction *Netdata_channelstatus* can perform CCA operation. If the communication protocol is embedded in hardware, then some protocol-related timer transactions are also required such as ACK waiting transaction and auto retransmission delay (ESB mode). A typical transceiver is modeled shown in Fig. 8.

A specific feature of transceiver model in EMrise_SS is that its configuration is based on register settings like real hardware, which means that related registers are defined in each specific transceiver model. With the register configuration mechanism, the simulation of the network becomes flexible because different nodes could keep their own settings during the whole simulation process shown in Fig. 9. The final results could be acquired from each node to evaluate the performance of such a heterogeneous configured network scenario.

4.1.3.1 Modelling of CC2420 The CC2420 is a 2.4 GHz IEEE 802.15.4 compliant RF transceiver chip with a maximum data rate of 250 Kbps. CC2420 is designed to support low power applications, it provides on-chip packet handling by means of automatically adding a preamble sequence, frame check sequence and frame delimiter to the data packet. It also provides data buffering, clear channel assessment, link quality indication and packet timing information, all of which reduce the load and power consumption on the host microcontroller. Since for CC2420 based sensor motes (e.g., Telos, Tmote Sky and Shimmer), MAC algorithms for channel access are programmed by software and embedded in the microcontroller, the CC2420 acts only as a TX/RX device for transmission and reception of the IEEE 804.15.4 format based data packet. Therefore, the model of CC2420 in EMrise_SS employs the generic model presented in Fig. 9.

4.1.3.2 Modelling of nRF24L transceiver series Designed by Nordic Semiconductor, the nRF24 transceiver series [43] represent ultra-low-power RF chips that provide available solutions for 2.4 GHz ISM band wireless applications. As a typical type in nRF24 family, nRF24L01+ supports Enhanced ShockBurst (ESB) with a maximum data rate of 2 Mbps and backwards compatible with ShockBurst (SB). Designed both by Nordic, ESB is a baseband MAC protocol engine embedded in transceiver chips with some basic channel access mechanisms, while SB is only a simple TX/RX protocol without any channel access algorithm. ESB and SB protocol has been modeled in EMrise_SS via a finite state machine shown in Fig. 10.

4.1.4 Network modelling

The network model is derived and modified from the SystemC Network Simulation Library (SCNSL). In network Modelling, SystemC provides primitives such as concurrency mechanism as well as events that are used to simulate network transmission and reception behaviors. Before the start of simulation (in the elaboration process), *nodeProxy* class objects are instantiated with the same number of node instances in the SystemC entrance function *sc_main()*. The use of *nodeProxy* is to decouple each node implementation of the network simulation, and each *nodeProxy* instance uploads some significant information to the network object, including node identity, two-dimensional position information, TX output power and

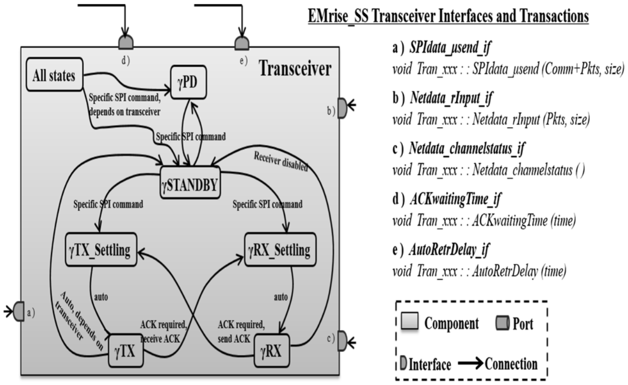

EMrise_SS Transceiver Interfaces and Transactions
Transceiver
All states
γPD
Specific SPI command, depends on transceiver
Specific SPI command
Receiver disabled
γSTANDBY
Specific SPI command
Specific SPI command
γTX_Settling
γRX_Settling
auto
auto
Auto, depends on transceiver
ACK required, receive ACK
ACK required, send ACK
γTX
γRX
a)
b)
c)
d)
e)
a) SPIdata_µsend_if
void Tran_xxx : : SPIdata_µsend (Comm+Pkts, size)
b) Netdata_rInput_if
void Tran_xxx : : Netdata_rInput (Pkts, size)
c) Netdata_channelstatus_if
void Tran_xxx : : Netdata_channelstatus ( )
d) ACKwaitingTime_if
void Tran_xxx : : ACKwaitingTime (time)
e) AutoRetrDelay_if
void Tran_xxx : : AutoRetrDelay (time)
Component
Port
Interface
Connection

Fig. 8 Typical transceiver model

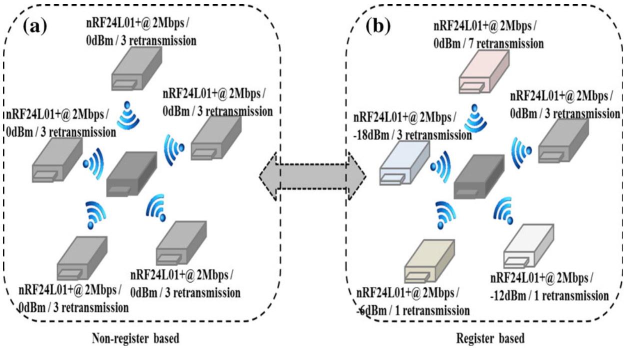

(a)
nRF24L01+@2Mbps/ 0dBm/3 retransmission
nRF24L01+@2Mbps/ 0dBm/3 retransmission
nRF24L01+@2Mbps/ 0dBm/3 retransmission
nRF24L01+@2Mbps/ 0dBm/3 retransmission
nRF24L01+@2Mbps/ 0dBm/3 retransmission
Non-register based
(b)
nRF24L01+@2Mbps/ 0dBm/7 retransmission
nRF24L01+@2Mbps/ 0dBm/3 retransmission
nRF24L01+@2Mbps/ -18dBm/3 retransmission
nRF24L01+@2Mbps/ -6dBm/1 retransmission
nRF24L01+@2Mbps/ -12dBm/1 retransmission
Register based

Fig. 9 Advantage of register configuration mechanism

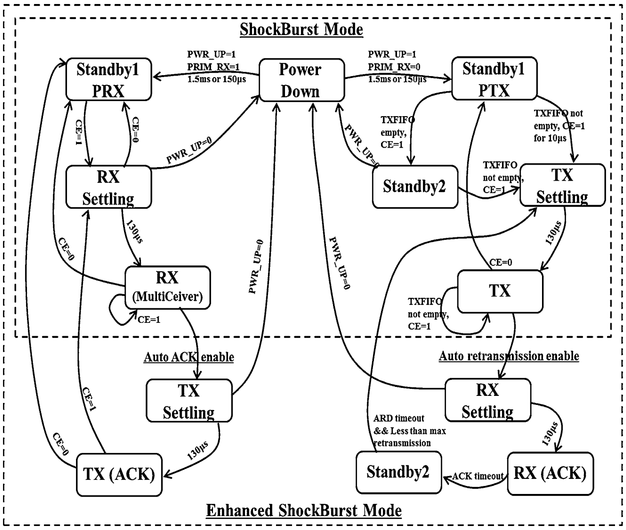


Fig. 10 Enhanced ShockBurst and ShockBurst model

receiver sensitivity. Combined with the channel model, these parameters are used to reproduce the network scenario and perform network connection assessment. At present, two types of channel model are supported in EMrise_SS, which are free space propagation model [44] and human's on-body Line Of Sight channel model. The most fundamental free space model in EMrise_SS is presented as follows.

$$P_r = \frac{P_t \cdot G_t \cdot G_r \cdot \lambda^2}{(4\pi D)^2} \tag{1}$$

where P_r is the receiver power sensitivity measured in dBm, P_t represents TX output power at transmitter, D is the distance (m) between the transmitter and receiver in two dimensional space, G_r and G_t are antenna gains of the receiver and transmitter respectively (it is common to select $G_r = G_t = 1$), and λ denotes the wavelength, calculated as

$$\lambda = \frac{c}{f_c} = \frac{3 * 10^8 (\mathrm{m/s})}{2.4 * 10^9 (\mathrm{Hz})} = 0.125(\mathrm{m}) \tag{2}$$

When simulation is initiated by *sc_start()* function, the over-the-air transmission time of the data packet and ACK frame will be calculated in the network model, and a waiting period is thus required for each transmission. After this time, the network model will distribute the packets to the receiver nodes within the radio transmission range. However, if collisions occur, no packets will be received. The collision scenario is simulated by checking the numbers of active transmitters that are interfering at a given receiver. If the number is greater than one, then packet transmission is considered as a failure because of the collision.

4.1.5 Sensor modelling

A generic sensor model can be designed in 3 ways. The first is periodic data generation, which samples the signals of the physical environment by an application defined time interval (sampling period). The second is to read data from an input file where this file can be a designer-specific data input file at the early design stage, if the actual sensor has not yet been chosen. This input file can also be the actual measurements from experiments. The third way is to model the sensor in a specific function.

In this work, sensor component modelling is mainly focused on the generic model development for periodic data sensing which consistent with the high-level design philosophy of EMrise_SS.

4.1.6 Energy modelling

As a significant part of EMrise_SS, the energy model plays an important role in energy consumption calculation and therefore it must be well designed and optimized to adjust accurate, realistic and flexible energy consumption evaluation. The characteristics of design method of the energy model is illustrated as follows.

Elaborate lib and register based An energy model is developed to incorporate an elaborate library with the current consumption of different operation modes of each hardware component. The working mechanism of this energy model is register-based. In the EMrise_SS energy model, the hardware registers of each component are mapped into the energy model as the mirrors, and the updates of these values are synchronous with the corresponding register prototypes in related hardware components, as shown Fig. 11. Developers and researchers can benefit greatly from this design method. Since the tracing of the correct energy value is always consistent with the change of component configuration during the whole simulation process, the energy model is therefore flexible enough to adjust scenarios where the hardware configurations need to be changed during the simulation. In that sense, the energy model is able to proceed without any interruptions to the whole simulation process, and no recompilations are required for the new scenarios after configuration changes.

Energy evaluation for multi-abstraction levels The energy model in EMrise_SS is able to trace the energy consumption at different abstraction levels. If the energy evaluation is focused on high-level, the change in operation state of each hardware component is linked to the energy model, and then the duration and current consumption of each state can be identified for the energy consumption calculation by applying the following basic formula.

$$E = P * t = (V * I) * t \tag{3}$$

If the energy evaluation is at low-level, the energy model can track each executed instruction in different operation states, record their execution time and associate them with

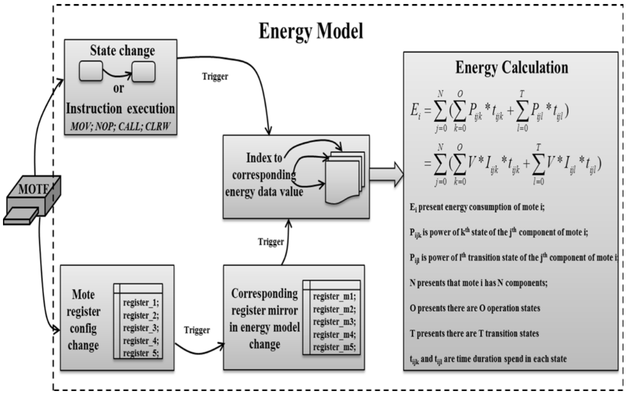

Energy Model
State change
or
Instruction execution
MOV; NOP; CALL; CLRW
Trigger
Index to corresponding energy data value
MOTE
Trigger
Mote register config change
register_1;
register_2;
register_3;
register_4;
register_5;
Trigger
Corresponding register mirror in energy model change
register_m1;
register_m2;
register_m3;
register_m4;
register_m5;
Energy Calculation
$E_i = \sum_{j=0}^{N}(\sum_{k=0}^{O} P_{ijk} * t_{ijk} + \sum_{l=0}^{T} P_{ijl} * t_{ijl})$
$= \sum_{j=0}^{N}(\sum_{k=0}^{O} V * I_{ijk} * t_{ijk} + \sum_{l=0}^{T} V * I_{ijl} * t_{ijl})$
E_i present energy consumption of mote i;
P_{ijk} is power of k^{th} state of the j^{th} component of mote i;
P_{ijl} is power of l^{th} transition state of the j^{th} component of mote i
N presents that mote i has N components;
O presents there are O operation states
T presents there are T transition states
t_{ijk} and t_{ijl} are time duration spend in each state

Fig. 11 Workflow of energy model

corresponding current loads in the energy library for the accurate energy estimation.

Calibrated energy support The energy model incorporates calibrated energy values from real-world measurements for realistic and accurate energy evaluation. These measurements can be acquired from any COTS sensor motes and in-house testbed motes. However, accurate power consumption data are not easy to extract from product specifications, since a sensor mote is composed of various chips and some chips need additional peripherals to guarantee running operation. These peripherals on chip modules and sensor mote boards can cause extra energy consumption (nRF24L01+ module in Sect. 4.2.2). Thus, the calibrated energy model in EMrise_SS becomes necessary.

Transition state energy consumption The energy model considers state transitions in each hardware component, because each transition introduces extra cost to the overall energy consumption that cannot be ignored. So the total energy consumption is calculated by adding the individual energy consumption of each operation state of every hardware component as well as the energy consumption of each transition state of every hardware component.

Multi performance metrics The energy model supports various performance metrics and can be calculated in *mW, mJ, mAh*. On the other hand, if using a specific battery module measured in capacity, then the lifetime performance can be estimated in minutes, hours, days, months or years.

4.1.7 *Case studies for energy evaluation and management*

In this section, EMrise_SS is used for the energy evaluation and management on four mote type based networks (Telos, MICAz, MICA2 and iHop@Node) under different case scenarios. Detailed energy information and comparison data are presented and analyzed for the corresponding mote platform in each case scenario. The impact of sensing start time and payload collecting strategy on the energy performance is studied, while the energy consumption of ESB and SB can be refer to [36].

4.1.7.1 Sensing start time Due to the spatially distributed characteristic of sensor node networks and cheap manufacturing of hardware components, it is difficult to guarantee the same start up time or clock period for each sensor node. Therefore, energy consumption with simultaneous sensing start time and random sensing start time (each node starts sensing first data at a random instant between the beginning of the simulation experiment and the first sample interval) are analyzed via a medical application scenario.

A typical health monitoring based network for human body area [1, 45] (Fig. 12) is studied to explore the impact of sensing start time on the energy consumption of sensor network. Consisting of 3–10 nodes, the network is attached to the human body for different physiological signal monitoring purpose (10 Hz for temperature sensor, 50 Hz for glucose sensor, 100 Hz for ECG-electrocardiogram). Three types of sensor mote platforms applied in the simulation experiments are iHop250 K (iHop@Node with 250 Kbps), Telos (250 Kbps) and MICAz (250 Kbps). For other parameter settings, a 2 byte payload is used to store 12 bits of sensed data. The default 0 dBm output power is selected on the transceiver for over-the-air packet transmission. The packets format of Telos and MICAz are shown in Fig. 4. The packet format of ShockBurst (SB) shown in Fig. 3 is applied by iHop250 K (1 byte preamble, 3 byte address as network id, 1 byte CRC, 4 bytes payload field including 2 bytes source node id and 2 bytes real payload data). Since only the unslotted CSMA/CA based MiWi communication protocol is supported by iHop@Node, both Telos and MICAz motes also need to apply the same unslotted CSMA/CA for the comparison. Four parameters of the unslotted algorithm are set as default values, which are 3, 5, 4 and 3 for *macMinBE*, *macMaxBE*, *macMaxCSMABackoffs* and *macMaxFrameRetries* respectively. In addition, each case runs for the same simulation time (4 s in this work) with different seeds to generate random backoff slot numbers, and the average value of 50 times of runs are employed.

Taking the results of Telos mote for analysis, Fig. 13 reveals that the case with random start time is more energy-efficient. Compared with the random start sensing case, simultaneous sensing consumes more energy due to protocol overhead. This is because all sensor nodes start data sensing at the same time under the simultaneous case, and before over-the air-transmission 2^{BE}-1 (BE initialize as *macMinBE*) unit backoff periods are required. However, when *macMinBE* is initialized to a smaller value (e.g., default value 3), it is easy for different sensor nodes to get the same number of backoff slots (0–7). Thus, with the same sensing start point and the same/similar backoff period, the transmission time of many packets can overlap with each other, which causes serious packet collisions during a short period of time and therefore extra energy is

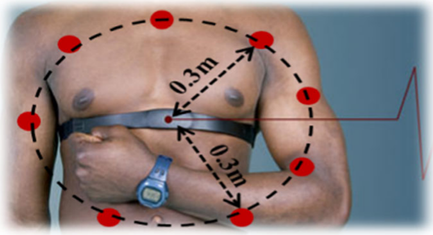


Fig. 12 Sensor node deployment on human body

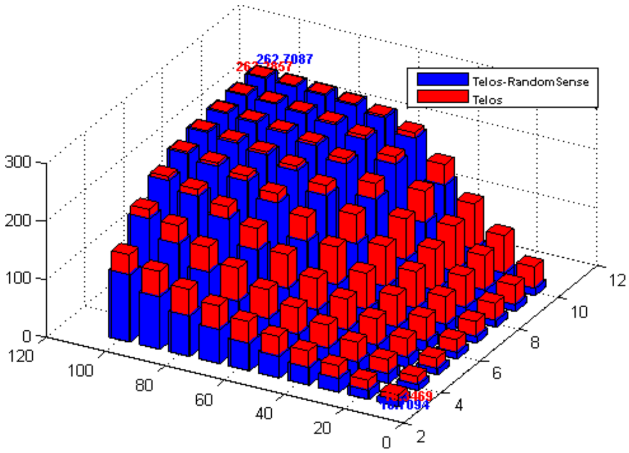


Fig. 13 Impact of random sensing start time on energy comparison

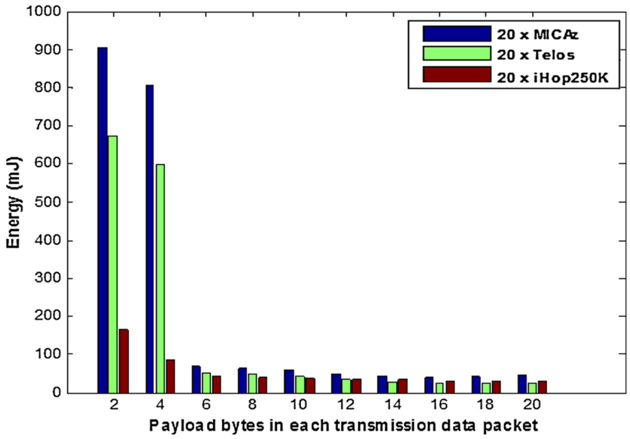


Fig. 14 Energy consumption under payload collecting strategy

consumed. On the contrary, the transmissions of the packets are spread out and channel competition is light with random sensing start time. Much energy is saved since it is more likely that packets can be successfully delivered with no collisions and therefore taking low protocol overhead.

4.1.7.2 Packet collecting strategy A typical lab-sized network (with a size of about 20 sensor nodes [46]) is under investigation with payload collecting strategy for energy evaluation. The comparison experiments are made on iHop250 K, MICAz and Telos-based networks. In the first experiment, a packet with a 2 byte payload is transmitted every 10 ms, then a 4 byte payload packet every 20 ms, a 6 byte payload packet every 30 ms until the transmission of a 20 byte payload packet every 100 ms. Each experiment runs for the same time (10 s) and the average value of 50 runs is adopted. The results are presented in Fig. 14.

Since the increasing payload data can be attached to the same size of packet overhead and when this data collecting strategy is used, fewer data packets are transmitted when the same amount of payload data is required. Therefore, the energy wasted in packet overhead under frequent transmission conditions can be greatly saved, while energy cost is also reduced with fewer ACK frames. Fewer transmission also saves the energy wasted in active mode due to possible packet overflow preventing the sensor node entering into sleep mode. Besides, the experimental results show the packet loss probability drastically reduces from 96.3 % in the 2-byte, 90.1 % in 4-byte cases to 3.98 % in the 20-byte case for both MICAz and Telos (MICAz and Telos share the same protocol model). For iHop250 K, this rate is from 93.9 % in the 2-byte case, 36.0 % in the 4-byte case, and is reduced to 1.97 % in the 20-byte case.

Although the payload collecting method can conserve energy, it has some drawbacks. When the channel is of low quality with a high bit error rate (BER)/packet error rate (PER), longer data packets are more likely to subsume errors and necessitate retransmissions. This will definitely cause extra energy consumption and longer packet latency.

4.2 EMrise_MS (EMrise Measurement System)

EMrise_MS, known as iWEEP_HW [36, 37], is the hardware based energy measurement and management platform that is made up of a multi-channel energy measurement device (MEMD) and energy data management software platform (EDMSP) to be able to measure the energy related data of iHop@Node, MICA mote, Telos mote and other similar sensor motes. For MEMD, it can provide simultaneous measurements on different components of a sensor node or simultaneous measurements on several different nodes by using its multiple channel advantage. For EDMSP, it offers an easy to use GUI to help users manage and save energy data from the sensor node.

4.2.1 iHop@Node testbed node

In order to explore energy performance of future sensor network which has high data rate and ultra-low power features, iHop@Node [36, 37] is designed and built by Microchip PIC16F88 [47] microcontroller with 0.93–1.30 mA current load in active mode and 0.3–0.5 µA in sleep mode, as well as high data rate (2 Mbps/1 Mbps) ultra low power transceiver Nordic nRF24L01+.

iHop@Node has a small-dimension of 4.1″ * 2.4″ and it also has multi-layer architecture design, which means that the microcontroller layer, transceiver layer, sensor layer and other components layer (e.g., LEDs, external memory) are all separate and can be fed either by the same power supply or by different supplies. When different power supplies are adopted, each hardware component is considered as an independent load under test and detailed energy information can be measured respectively and synchronously by using MEMD. Compared with the software based functional components decomposition approach [31], this means is at hardware level and provide much more reliable results.

4.2.2 MEMD

Without introducing side-effects to the sensor node hardware or software for energy measurements, and saving cost from using expensive measurement equipment such as oscilloscope and acquisition card [31]. A dedicated MEMD (**M**ultichannel **E**nergy **M**easurement **D**evice) is therefore designed and implemented.

Figure 15 is a simple architecture of one channel of MEMD. A shunt resistor (Rsense) of the known value is used between power supply and sensor node for energy consumption measurements, the voltage drop (Vsense) across the resistor will be amplified as the final measured energy usage data (Vout). Then, a 10-bit ADC chip [18] is employed to sample this Vout, and the sampled values can further be stored in the buffer of PIC18 [48]. The sampled data can be saved continuously in the whole buffer and then sent out in a buffered sampling mode. Alternatively, the data can also be sent out immediately after each sampling, in an unbuffered sampling mode. To simplify the development, we use commercial virtual COM port module UR232R [49] for the communication between PIC18 and data terminal. Serial UART signals generated by the MCU is able to be sent over UB232R to data terminal for later evaluation and analysis. However, a large amount of raw measurements are likely to cause memory problem if the experiments run for a long time.

With the multi-layer architecture design of iHop@Node, MEMD can measure energy consumption on each hardware component separately and synchronously as shown in case 1 of Fig. 16a). While for case 2, instead of separate hardware components, MEMD is used to measure the energy consumption of four independent sensor motes. In Fig. 16b), the MEMD prototype is presented. Channel 1 of MEMD is responsible for the energy measurement of nRF24L01+ transceiver of iHop@Node, channel 2 is for PIC16 microcontroller energy measurements, channel 3 can be attached to the sensor chip for energy measurement, and channel 4 is used for other hardware components such as LEDs or external memory. All the channels run samplings simultaneously, and measurements are stored in respective microcontrollers via buffered sampling mode for high speed. These data will be sent and saved separately in data terminal according to the channel number for later evaluation. In addition, the cost of this MEMD is much lower than an oscilloscope, an acquisition card and recently proposed Sensor Node Management Device (SNMD) built by KIT [50]. Table 1 lists the detailed measurement capabilities of MEMD.

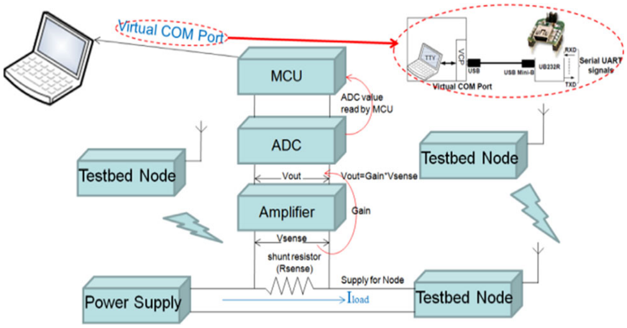


Fig. 15 Architecture of MEMD (one channel)

4.2.3 EDMSP

Compared with [51, 52], which provide SD or MMC card memory interfaces for energy measurement storage, EMrise_MS provides an alternative means for energy data saving and management.

Since huge storage space is needed for the collection of raw measurements, the on-board memory card based method cannot even support the experiment running for several minutes or hours. Therefore, saving measurements onto hard disk with TB capacity level would be a better choice. Thus, EDMSP is built to facilitate the raw measurement sampling process as well as the data sending to PC process for storage and future evaluation. A GUI-based Energy Data Management Software Platform (EDMSP) is designed and presented in Fig. 17.

The implementation of EDMSP is based on the well-known CSerialPort class [53] developed by MFC and

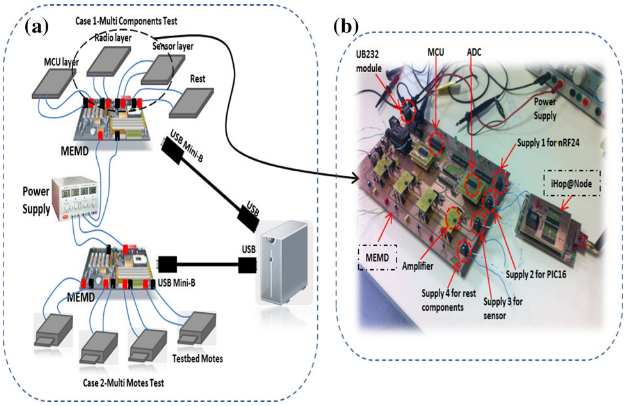
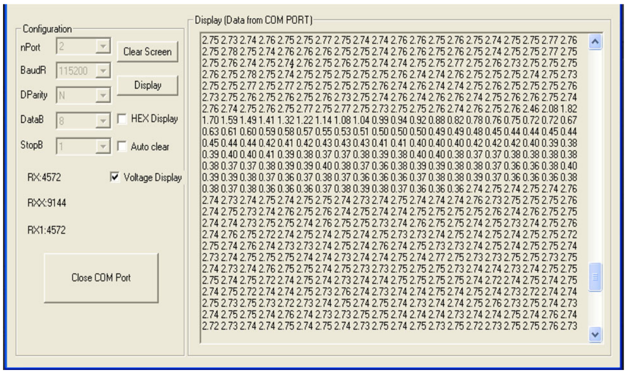


Fig. 16 MEMD prototype

Table 1 MEMD capabilities

MEMD capabilities
Four channels for sampling energy usage data
Synchronous energy usage data sampling
10-bit resolution for ADC value
1900 samples buffers (each channel)
Configurable gain (by tuning R1 and R2 in schematics)
150 kHz buffered sampling rate
5.5 kHz unbuffered sampling rate
Measurement range 0–40 mA
Configurable data rate via virtual COM port interface (UB232R module)



nPort 2
BaudR 115200
DParity N
DataB 8
StopB 1
□ HEX Display
□ Auto clear
☑ Voltage Display
RX:4572
RXX:9144
RX1:4572

```
2.75 2.73 2.74 2.76 2.75 2.75 2.77 2.75 2.74 2.74 2.76 2.76 2.75 2.76 2.75 2.74 2.75 2.75 2.77 2.76
2.75 2.78 2.75 2.74 2.76 2.76 2.76 2.75 2.75 2.74 2.76 2.76 2.75 2.76 2.75 2.74 2.75 2.75 2.77 2.75
2.75 2.76 2.74 2.75 2.74 2.76 2.75 2.76 2.75 2.74 2.74 2.75 2.75 2.77 2.75 2.76 2.73 2.75 2.75 2.75
2.76 2.75 2.78 2.75 2.74 2.75 2.75 2.75 2.75 2.75 2.76 2.74 2.74 2.76 2.75 2.75 2.75 2.74 2.75 2.73
2.75 2.75 2.77 2.75 2.77 2.75 2.75 2.75 2.75 2.76 2.74 2.76 2.74 2.74 2.75 2.76 2.75 2.75 2.75 2.76
2.73 2.75 2.76 2.75 2.76 2.75 2.76 2.73 2.75 2.74 2.76 2.74 2.76 2.76 2.74 2.75 2.76 2.76 2.75 2.74
2.76 2.74 2.75 2.76 2.75 2.77 2.75 2.77 2.75 2.73 2.75 2.75 2.76 2.74 2.76 2.75 2.76 2.46 2.08 1.82
1.70 1.59 1.49 1.41 1.32 1.22 1.14 1.08 1.04 0.99 0.94 0.92 0.88 0.82 0.78 0.76 0.75 0.72 0.72 0.67
0.63 0.61 0.60 0.59 0.58 0.57 0.55 0.53 0.51 0.50 0.50 0.50 0.49 0.49 0.48 0.45 0.44 0.44 0.45 0.44
0.45 0.44 0.44 0.42 0.41 0.42 0.43 0.43 0.43 0.41 0.41 0.40 0.40 0.40 0.42 0.42 0.42 0.40 0.39 0.38
0.39 0.40 0.40 0.41 0.39 0.38 0.37 0.37 0.38 0.39 0.38 0.40 0.40 0.38 0.37 0.37 0.38 0.38 0.38 0.38
0.38 0.37 0.37 0.38 0.39 0.39 0.40 0.38 0.37 0.36 0.38 0.39 0.39 0.38 0.38 0.37 0.36 0.36 0.38 0.40
0.39 0.39 0.38 0.37 0.36 0.37 0.38 0.37 0.36 0.36 0.36 0.37 0.38 0.38 0.39 0.37 0.36 0.36 0.36 0.38
0.38 0.37 0.38 0.36 0.36 0.36 0.37 0.38 0.39 0.38 0.37 0.36 0.36 0.36 2.74 2.75 2.74 2.75 2.74 2.76
2.74 2.73 2.74 2.75 2.74 2.75 2.75 2.74 2.73 2.74 2.75 2.74 2.74 2.74 2.76 2.73 2.75 2.75 2.75 2.76
2.74 2.75 2.73 2.74 2.76 2.75 2.76 2.74 2.75 2.74 2.74 2.75 2.75 2.75 2.75 2.76 2.74 2.76 2.75 2.75
2.74 2.74 2.73 2.75 2.75 2.74 2.76 2.75 2.75 2.73 2.74 2.76 2.75 2.75 2.74 2.75 2.73 2.74 2.75 2.76
2.74 2.76 2.75 2.72 2.74 2.75 2.74 2.75 2.74 2.75 2.73 2.73 2.74 2.75 2.74 2.75 2.74 2.75 2.75 2.72
2.75 2.74 2.76 2.74 2.73 2.73 2.74 2.75 2.74 2.76 2.74 2.75 2.73 2.73 2.74 2.75 2.74 2.75 2.75 2.74
2.73 2.74 2.75 2.75 2.75 2.74 2.74 2.73 2.74 2.74 2.75 2.74 2.77 2.75 2.73 2.73 2.75 2.75 2.73 2.75
2.74 2.73 2.74 2.76 2.75 2.75 2.74 2.75 2.73 2.73 2.75 2.75 2.75 2.73 2.74 2.74 2.73 2.74 2.75 2.75
2.75 2.74 2.75 2.72 2.74 2.75 2.74 2.73 2.74 2.74 2.75 2.75 2.75 2.75 2.74 2.72 2.74 2.75 2.74 2.75
2.74 2.75 2.72 2.74 2.74 2.75 2.73 2.76 2.74 2.73 2.74 2.74 2.75 2.74 2.75 2.74 2.73 2.72 2.74 2.74
2.75 2.73 2.75 2.73 2.72 2.73 2.74 2.75 2.74 2.75 2.74 2.73 2.74 2.74 2.75 2.76 2.73 2.75 2.74 2.73
2.74 2.75 2.75 2.74 2.76 2.74 2.73 2.73 2.75 2.75 2.74 2.74 2.74 2.73 2.74 2.74 2.75 2.74 2.76 2.74
2.72 2.73 2.74 2.74 2.75 2.74 2.75 2.74 2.73 2.75 2.74 2.75 2.73 2.75 2.72 2.73 2.75 2.75 2.76 2.73
```

Fig. 17 EDMSP

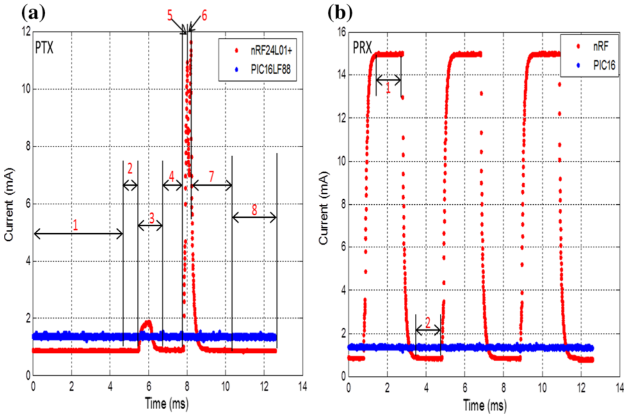


Fig. 18 PTX and PRX measurements

Windows API. Many configurable parameters can be set via EDMSP GUI such as port number, baud rate, data bit and stop bit. Once the selected COM port receives data, the number of received data will be recorded and shown on the configuration table, in the meantime the data values will also be shown in the display window. Since EDMSP supports multi-type display functions, the received data can be presented in decimal value, hex value, voltage value, current value and even power consumption value when the corresponding display check-box is selected. Other specific functions can also be easily extended to EDMSP according to users' requirements. Further, all the displayed energy values can be printed out into a TXT file for the convenience of subsequent data processing and evaluation.

4.2.4 Measurements

In this part, MEMD and EDMSP are applied for the measurements and calibration of the iHop@Node testbed. The current consumption measurements of iHop@Node are presented in Fig. 18 on both PTX and PRX devices. Since the microcontroller and transceiver are typically the most power consuming parts, the measurement results are mainly focused on these two parts in order to simply the investigations.

Figure 18a represents the process of PTX device under Enhanced ShockBurst mode, where the iHop@Node testbed is configured as 2 Mbps and 0 dBm output power. Each operation step in the figure is marked with a corresponding number for the clarity of illustration.

Interval 1 The PIC16 microcontroller is in active mode, while the nRF transceiver is in power down mode. The microcontroller demands about 1.4 mA current consumption, and the transceiver consumes about 0.8–0.9 mA current. During the whole experiment, the PIC16 microcontroller runs in active mode all the time.

Interval 2 and 4 During interval 2, the microcontroller sends commands via SPI to the transceiver to configure it from the power down mode to the standby mode. For interval 4, the microcontroller sends a 10 byte payload to the nRF transceiver's TXFIFO also by SPI. Both intervals show that SPI communication will not increase the current consumption of PIC16 microcontroller.

Interval 3 After reception of SPI commands from microcontroller, the crystal of nRF transceiver starts up and the transceiver enters into standby mode from power down mode. The maximum current consumption for this interval can reach about 1.83 mA.

Interval 5 The CE (Chip Enable) signal is set high for at least 10 μs, which triggers the transceiver first into the transition state TX_Settling. After 130 μs in this TX_Settling

Table 2 Measurements of iHop@Node (oscilloscope: Tek 1012B)

iHop@Node	Vout (scope)	Vout (MEMD)	Error of Vout	Current (scope)	Current (MEMD)	Current datasheet
PIC16 (sleep)	0.048 V	0.047 V	0.001 V; 2.08 %	0.13 mA	0.12 mA	0.5 μA
PIC16 (Active)	0.36 V	0.35 V	0.01 V; 2.80 %	1.399 mA	1.34 mA	1.2 mA
PIC16 (SPI)	0.36 V	0.35 V	0.01 V; 2.80 %	1.399 mA	1.34 mA	N/A
PIC16 (UART)	0.36 V	0.35 V	0.01 V; 2.80 %	1.399 mA	1.34 mA	N/A
PIC (instruction)	0.36 V	0.35 V	0.01 V; 2.80 %	1.399 mA	1.34 mA	N/A
nRF (power down)	0.23 V	0.21 V	0.02 V; 8.70 %	0.89 mA	0.82 mA	0.9 μA
nRF (standby)	0.24 V	0.22 V	0.02 V; 8.33 %	0.93 mA	0.85 mA	26 μA
nRF (startup) max	0.48 V	0.47 V	0.01 V; 2.08 %	1.86 mA	1.83 mA	0.4 mA (average)
nRF (TX_Settling) max	2.24 V	2.23 V	0.01 V; 0.45 %	8.70 mA	8.66 mA	8.0 mA (average)
nRF (TX@0dBm)	2.84 V	2.80 V	0.04 V; 1.41 %	11.03 mA	10.88 mA	11.3 mA
nRF (RX_Settling) max	2.76 V	2.75 V	0.01 V; 0.36 %	10.72 mA	10.68 mA	8.9 mA (average)
nRF (RX@2Mbps)	3.85 V	3.82 V	0.03 V; 0.78 %	14.96 mA	14.84 mA	13.5 mA

state, the transceiver will automatically enter into the TX mode for packet transmission. An average current consumption for this 130 μs settling state is given in the product specification. Based on the measurement results, there are actually two steps consuming different currents in this transition state. In the first 50 μs, the increase rate of current consumption is slower than the last 80 μs, which is probably because in the first 50 μs the transceiver only performs packet assembly, while for the next 80 μs many internal components of the transceiver are turned on for over the air packet transmission. Therefore, a much more rapid increase in current consumption can be detected during the second step.

Interval 6 With the ESB mode, the nRF transceiver will automatically enter into RX_ACK mode waiting for the acknowledgement packet. A transition state RX_Settling is therefore required to make the transceiver from PTX to PRX. Since the current consumption in RX_Settling is lower than that consumed in TX and RX states, so the consumed current during interval 6 will first reduce and then increase.

Interval 7 and 8 After the successful of reception ACK frame, the nRF transceiver automatically returns to standby mode. During interval 8, the microcontroller sends a command via SPI to configure the transceiver into power down mode.

Figure 18b represents the process of periodical listening of PRX device.

Interval 1 The transceiver is in RX mode for channel listening and is ready to receive possible incoming packets. The microcontroller works is in active mode.

Interval 2 CE is set low by the microcontroller, and the transceiver returns to standby mode.

Note that the nRF24L01+ transceiver is integrated into a module with an onboard 3.3 V regulator, RP-SMA 2.4 GHz antenna and some peripheral circuits, so as shown in Table 2, the measured current values of nRF24L01+ are much greater than those listed in the product specification. This is especially true in power down and standby modes, because significant current loads are required by other hardware components on the module except for the nRF24L01+ chip itself. Table 2 also shows that SPI and UART communications in the microcontroller do not consume extra energy. This is also true for the instruction execution of microcontroller, where the instructions under investigation are AND, OR, XOR, ADD, DIV, MUL, WHILE and FOR loop [54]. Besides, the measured current consumption in the sleep mode of microcontroller is found to be significantly different (three orders of magnitude) from the values in the specification. This discrepancy could be related to some other scenarios: (1) The enable/disable of interrupt, SPI, ADC and Timer functions on the MCU chip (about 0.2 mA current consumption can be saved if these functions are disabled). (2) The enable/disable of watchdog function. (3) The configuration of I/O ports.

In addition, considering other popular sensor motes, such as Telos, Shimmer Node, Tmote (TelosB), MicaZ, Mica2 and N@L, the measurement range of the proposed MEMD and EDMSP is still available to use.

4.2.5 Calibration

After measuring energy consumption, the measurement results are used for testbed operation calibration. Measurements provide detailed time duration information on each operation state of the iHop@Node, including chip enable signal duration time, delay of interrupt, duration of transition states, SPI communication time, etc. So with the calibrated model, an ESB mode-based model is simulated via EMrise_SS and is compared with the real world measurement model. Detailed information is presented in Fig. 19. In this experiment, the packet-based data rate is configured to 2Mbps, and the packet format contains a 5 byte payload while other packet fields are maintained as default values (1 byte preamble, 1 byte CRC, 5 bytes address, and 9 bits packet control field). The experiment shows that the calibrated simulation model can provide very close results to the measurements, so more trusted and accurate energy consumption is able to be predicted when calculated with the realistic measurement energy data.

The second case study is to evaluate iHop@Node maximum continuous packet stream data rate for transmission and reception.

The maximum transmission packet stream rate can be achieved via following steps. Firstly, single packet-based data rate is configured as 2Mbps. Secondly, SB mode is used rather than ESB mode, because SB can send three packets continuously without waiting for the ACK frame after each packet transmission. For the packet format, each packet contains a 32 byte payload and default values for the other overheads (1 byte preamble, 1 byte CRC and 5 bytes address). In this case, one transmission process contains three continuous packet sending, and this transmission mechanism is used in both simulation and experiment. The detailed result is presented in Fig. 20.

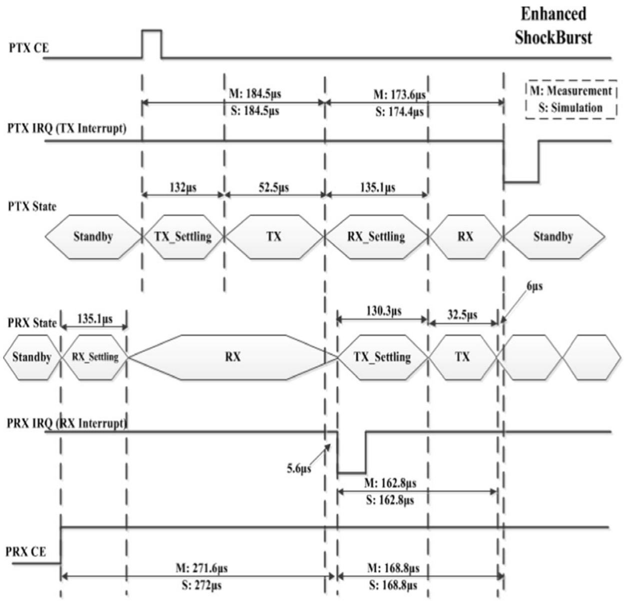


Fig. 19 Calibrated ESB simulation model and corresponding measurements

According to the results, the continuous packet stream rate for simulation and experiment can be calculated respectively as follows.

Measurement:

$$\textit{Number of bits to send}: (39 * 3 * 8) * 3 = 2808 \text{ bits} = 2808 * 10^{-6}\,\text{Mbit}$$

$$\textit{Transmission time}: 5532\ \mu\text{s} = 5532 * 10^{-6}\ \text{s}$$

$$\textit{Continuous packet stream rate}: \left(2808 * 10^{-6}\right)/\left(5532 * 10^{-6}\right) = 0.5076\ \text{Mbps}$$

Simulation:

$$\textit{Number of bits to send}: (39 * 3 * 8) * 3 = 2808 \text{ bits} = 2808 * 10^{-6}\ \text{Mbit}$$

$$\textit{Transmission time}: 5495.9\ \mu\text{s} = 5495.9 * 10^{-6}\text{s}$$

$$\textit{Continuous packet stream rate}: \left(2808 * 10^{-6}\right)/\left(5495.9 * 10^{-6}\right) = 0.5109\ \text{Mbps}$$

However, based on the measurements, the receiver cannot receive the packet stream at this 0.5 Mbps rate without losing packets, because the SPI payload reading speed is slower than its payload writing speed. This means that before reading out all payloads from the three RXFIFO spaces (each RXFIFO contains 32 bytes payload), new data packets will continuously come from the transmitter to make the RXFIFOs full. Then, the transceiver cannot receive any new data packets with the saturated RXFIFOs, which will lead to packet loss. In order to make sure that the receiver can successfully receive all the packets without

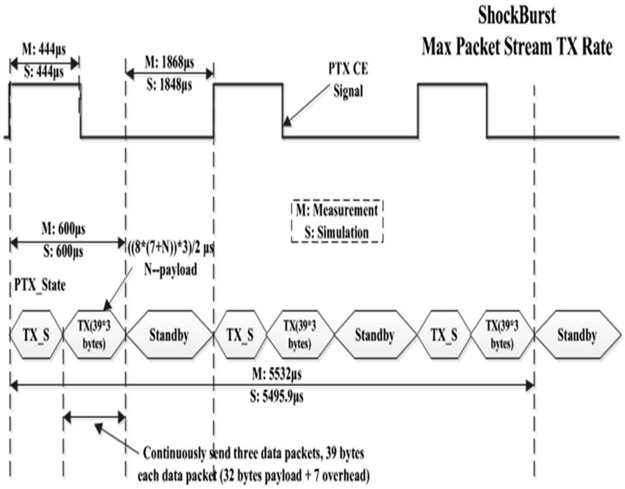


Fig. 20 Maximum TX rate of packet stream

any loss, the receiver should first read out all the packets from RXFIFOs and have PRX_IRQ (RX interrupt) cleared before the transmitter finishes the first data packet sending during each transmission process. This is to guarantee that the next PRX_IRQ can be triggered, and that in its interrupt service routine the PIC16 microcontroller can continuously read out all the packets from RXFIFOs and then clear PRX_IRQ for the preparation of the new incoming packets. Thus, measurements are run to find the maximum packet stream reception rate, and the experimental measurement is compared with its corresponding simulation result shown in Fig. 21.

Based on the results, the maximum packet stream reception rate is calculated respectively as follows.

Measurement:

$$\begin{aligned}&\mathit{Number\ of\ bits\ to\ send}:\\&(39*3*8)*3=2808\ \text{bits}=2808*10^{-6}\ \text{Mbit}\end{aligned}$$

$$\begin{aligned}&\mathit{Transmission\ time}:\\&7100\ \mu\text{s}=7100*10^{-6}\ \text{s}\end{aligned}$$

$$\begin{aligned}&\mathit{Continuous\ packet\ stream\ rate}:\\&\left(2808*10^{-6}\right)/\left(7100*10^{-6}\right)=0.395\ \text{Mbps}\end{aligned}$$

Simulation:

$$\begin{aligned}&\mathit{Number\ of\ bits\ to\ send}:\\&(39*3*8)*3=2808\ \text{bits}=2808*10^{-6}\ \text{Mbit}\end{aligned}$$

$$\mathit{Transmission\ time}=7038\ \mu\text{s}=7038*10^{-6}\ \text{s}$$

$$\begin{aligned}&\mathit{Continuous\ packet\ stream\ rate}:\\&\left(2808*10^{-6}\right)/\left(7038*10^{-6}\right)=0.399\ \text{Mbps}\end{aligned}$$

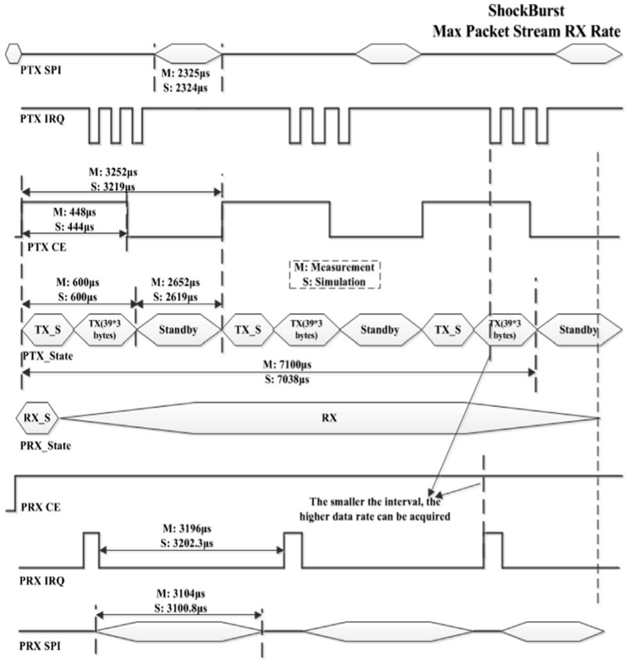


Fig. 21 Maximum RX rate of packet stream

Despite the fact that reception rate is lower than transmission rate, it is high enough to guarantee a much better performance of many sensor network application scenarios.

4.3 EMrise_OpS (EMrise Optimization System)

In this section, iMASKO [55], a generic GA-based optimization framework, is integrated into EMrise system and regarded as EMrise_OpS. Due to the global search and intelligent properties of genetic algorithms (GAs), the framework is able to automatically and effectively fine tune hundreds of possible solutions to find the best tradeoff solution. Before the description of EMrise_OpS, a brief introduction of genetic algorithm is first presented as follows.

4.3.1 Genetic algorithms (GAs)

Genetic algorithms (GAs) are stochastic search engines that mimic natural selection and biological evolution processes. GAs are considered to be highly efficient techniques, since their search is based on a whole population of individuals in parallel calculations rather than a single point, which improves the chance of achieving the global optimum solution and helps to avoid local stationary points. Further, the use of a fitness function for evaluation can extend GAs to any kinds of continuous or discrete optimization problems. As a domain-independent search technique, GAs are ideal for applications where domain knowledge and derivative information are difficult or impossible to provide [56]. In other words, tedious and knowledge-based processes can be greatly simplified, which is of especial interest for inexperienced designers, since only the fitness function and corresponding fitness levels influence the whole search process [57, 58]. Finally, some advantages of GAs are summarized and listed in Table 3.

The use of GAs in wireless sensor network solutions has been proved to be very successful for a number of works. The most common use of GAs is how to achieve an energy-aware network under specific routing conditions, which includes the finding of proper cluster heads for data aggregation in the network; the selection of the optimal path to reduce hops (and thus transmission energy); and at the same time the reduction of channel contention. Such energy management efforts can be found in [59–61].

4.3.2 Architecture of the EMrise_OpS

With the same architecture of iMASKO, EMrise_OpS is able to quickly and efficiently explore simulation-based results or the theoretical model-based results optimization

Table 3 Advantages of GAs compared to the conventional methods

Advantage of GAs
Parallelism, efficiency, reliability, easily modified and adaptable to different problems
Large and wide solution space search ability
Non-knowledge based optimization process
Use of fitness function for evaluation
Easy to discover global optimum, avoid trapping in local optima
Capable of multi objective optimization, can return a suite of potential solutions
Good choice for large-scale/wide variety of optimization problems

on energy management purpose. In this section, EMrise_OpS is applied to automatically fine-tune the results from SystemC-based simulation of EMrise_SS.

In practical use, the optimization of related parameters in the design space can help find the application-specific solution to achieve the most optimal network performance for the given scenario before real deployment. By using EMrise_OpS, only two things are required: (1) A set of input parameters in the design space. (2) A user defined fitness function. As long as the system/model under evaluation can return the required data metrics to the fitness function, the detailed implementation of the system/model (simulators/emulators/mathematical models) can be ignored, which is of especial interest for non-experts in wireless sensor network design. Finally, the configuration of the GA optimization process is very simple on EMrise_OpS for users, since the interfaces of the corresponding algorithm parameters are provided and the configurations can be done via a MATLAB-based GUI or command lines.

A brief workflow of EMrise_OpS is presented in Fig. 22. Before starting the optimization, relevant parameters are fed to the simulation environment and to the GA evaluation engine respectively. Once the corresponding metrics have been generated from simulation and processed by the fitness function, the GA evaluation engine will decide whether the final result can satisfy the criteria of the given task. If so, the solution can be considered and applied in real applications. Otherwise, another cycle of the parameter tuning will start until the qualified solution for the task is found.

In Table 4, the typical optimization steps of EMrise_OpS are given by easily identified syntax: generate, evaluate and compare. The parameter space ***X*** could be a set of communication strategies/protocols, different configurations of a given protocol, or different application scenarios as long as the parameters that under the investigation can be represented as the qualified individuals in GAs. For the performance space ***P***, it contains the metrics that can reflect network performance (e.g., energy consumption, network reliability, network delay). For the comprehensive and overall optimization for the given problem, the metrics returned by the fitness function are separated and independent for the synchronous multi-objective optimization.

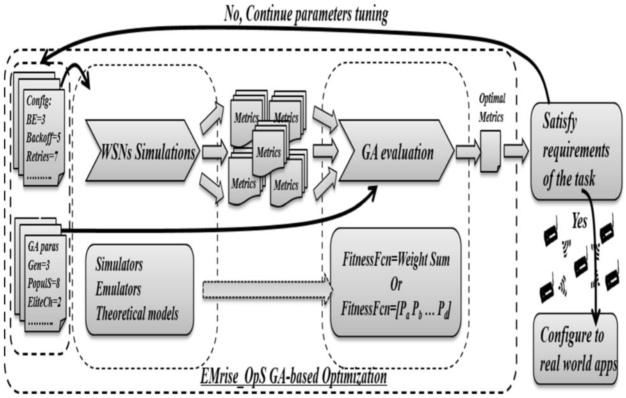


Fig. 22 Workflow of EMrise_OpS

4.3.3 *Multi-objective GA for Pareto-front optimization*

As previously mentioned, the performance optimization for the specific application scenario is always an overall and comprehensive process which must consider multiple objectives at the same time. The evaluation engine of EMrise_OpS returns a set of performance metrics ***p*** that need to be optimized/improved simultaneously and is given as:

$$\min_{(X)} \mathbf{p} = [p_1, p_2, p_3, \ldots, p_n] \tag{4}$$

where X is the parameter space, but the solutions that can simultaneously optimize every metric are difficult to find.

However, the GA based multi-objective optimization provides an alternative, since the parallelizable characteristics of GA (represented by the population size) can help evaluate many different sets of solutions in the parameter space simultaneously, which greatly improves the efficiency. The selection of the best individuals is based on the Pareto-front ($\partial F \prec$, F-solution space), which can be described as a solution $\mathbf{f} = [f_1 .. f_n]$ dominating $\mathbf{f}^* = [f_1^* .. f_n^*]$. This condition is true if each parameter of $\mathbf{f}$ is not greater than the corresponding parameter in $\mathbf{f}^*$ and there

Table 4 Pseudo code of EMrise_OpS optimization process

```
EMrise_OpS Optimization Process
Input initial solution: X, P ## X-parameters space, P-performance space
assume that: Xbest ← X, Pbest ← P
## Starting optimization until condition met
while stop condition not met then do #condition (e.g., max generation exceeded)
    Xnext ← generate(X) #GA operations: selection, crossover, mutation
    ## Evaluations are proceeded in parallel
    $P_{next}$ ← evaluate($X_{next}$) #evaluate results from SystemC simulations
    △cost ← compare($P_{next}$, P) #△cost = $P_{next}$ - P
    if △cost ≤ 0
        X ← $X_{next}$, P ← $P_{next}$
        if (P ≤ Pbest)
            Xbest ← X, Pbest ← P
        end if
    end if
end while ## when condition is met
return/output: Xbest, Pbest
```

is at least one parameter that is less, *i.e.* $\mathbf{f}_i \leq \mathbf{f}_i^*$ for each i and $\mathbf{f}_i < \mathbf{f}_i^*$ for some i. This is presented as $\mathbf{f} \prec \mathbf{f}^*$ to mean $\mathbf{f}$ dominates $\mathbf{f}^*$, and the total description in mathematics can be given as follows:

$$\forall_{(i \in \{1,\ldots,n\})}(f_i \leq f_i^*) \wedge \exists_{(i \in \{1,\ldots,n\})}(f_i < f_i^*) \quad (5)$$

In other words, the Pareto-front is part of the boundary of performance space. On this boundary, no solution is better in criteria that makes at least one performance metric better without making other metrics worse. The Pareto front captures the trade-offs between competing performance metrics and identifies the solutions that are non-dominated, as shown in Fig. 23.

In order to satisfy the simultaneous evaluation requirements on complex design problems and system development, if multi-objective GA optimization only focuses on a specific condition, the satisfying results are hardly to achieve. Therefore, a multi-scenario optimization method is proposed, which has already solved many engineering problems like [62, 63]. The use of this multi-scenario based multi-objective optimization is to find a robust solution which can give the optimal performance for all possible case scenarios and it can be presented as:

$$\min_{(X)} \mathbf{p} = [p_{a1}, p_{a2}, \ldots, p_{an}, \ldots, p_{m1}, p_{m2}, \ldots, p_{mn}] \quad (6)$$

where X is the vector of design parameters, p denotes a set of related performance metrics of several scenarios which are returned by EMrise_OpS evaluation engine from the simulations, p_{a1} to p_{an} and p_{m1} to p_{mn} represent the performance metrics of scenario a and scenario m respectively.

Take two primary objectives, energy and reliability (packet loss), as the example for performance optimization. If these two metrics need to be evaluated under different scenarios before practical use, the following four-dimensional (4D) front in Fig. 24 is able to transpose the concept for the case of multiple scenarios and help explore trade-offs for the metrics. In this 4D coordinate, the dotted lines plotted in quadrant one and four are the Pareto fronts. A_Metric2 and B_Metric1 axes are reversed to make the plot clearer, and quadrant two and three are used as the auxiliary quadrant to help locate the rectangle tradeoff markers. In the meantime, the rectangles are signs of metrics balance for different scenario requirements according to the decision maker.

4.3.4 GUI of EMrise_OpS and generic use of EMrise_OpS

A MATLAB based GUI has been developed for EMrise_SS and EMrise_OpS to link simulation and optimization, which facilitates configurations on both sides and makes the evaluation process visualizable. The GUI is shown in Fig. 25, where all the corresponding parameters can be set easily. Except for this multiple and detailed configuration interfaces support, EMrise_OpS is very generic of use. In this work, the fitness function uses EMrise_SS simulation results. However, the fitness function in EMrise_OpS can be of multiple types as long as it provides parameter space inputs and performance metrics outputs. Thus, results from other well-known WSN simulators such as NS-2, OMNeT++ and Prowler can also be used under the evaluation of EMrise_OpS, even if the

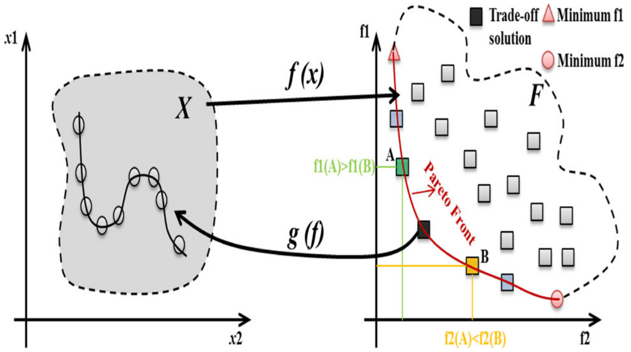


Fig. 23 Pareto-front optimality

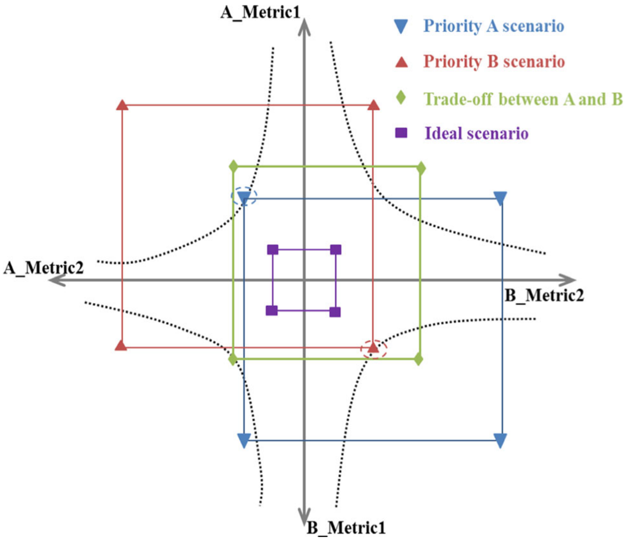


Fig. 24 4D Plot for multi-objective based multi-scenario optimization [58]

detailed implementation and knowledge of such simulations are unknown. Therefore, EMrise_OpS GUI also provides path loading to help select different types of fitness functions (in the mid-left of GUI) according to each specific case.

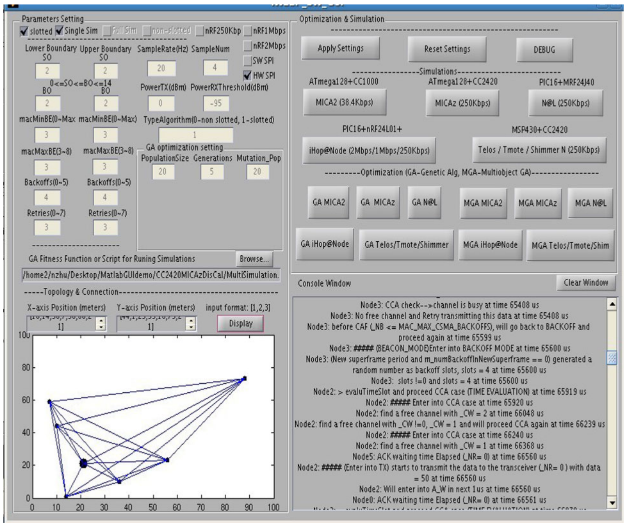
4.3.5 Case studies on multi-scenario and multi-objective optimization

The main goal of this test case is to verify the capability of the EMrise_OpS framework under the conditions of multi-scenario and multi-objective optimization. The case applied in this experiment is the typical wireless body area networks (WBANs) [64], which are widely used in medical and health care applications, as shown in Fig. 26. In the experiments, both Telos node and iHop@Node based networks are under investigation, the typical payload length in the packet is set to 2 bytes, and each simulation is set to 2 s. Fixed seed values are also adopted to mitigate the influence of the starting time of data sensing.

For such a network, each sensor node represents a specific scenario, and the use of different configurations of the unslotted CSMA/CA algorithm under this multi-scenario case could cause different performances at each sensor node. Note that during the algorithm tuning process, the same algorithm parameter combination is used to configure all sensor nodes. As mentioned previously, performance optimization problems always involve multiple objectives to be met simultaneously in each specific scenario. These objectives are usually conflicting such as achieving maximum network reliability and at the same time minimizing the energy consumption. Therefore, usually there is no single solution to a multi-objective optimization problem, in addition to the use of the Pareto front method to find a set of mathematically equal solutions.

In the test, two objectives (energy and packet loss) are subjected to optimization within the framework to achieve the solution trade-offs. In the multi-scenario situation of the network, the fitness function of each individual is composed of two pairs of energy consumption and packet loss rate for two different sensor nodes. The detailed fitness function is given as:

$$\min_{(X)} \mathbf{p} = [EnergyNode1, PktLossProNode1, EnergyNode2, PktLossProNode2] \tag{7}$$


Parameters Setting
☑ slotted ☑ Single Sim □ Full Sim □ non-slotted □ nRF250Kbp □ nRF1Mbps □ nRF2Mbps □ SW SPI ☑ HW SPI
Lower Boundary SO: 2
Upper Boundary SO: 2
0<=SO<=BO<=14
Lower Boundary BO: 2
Upper Boundary BO: 2
SampleRate(Hz): 20
SampleNum: 4
PowerTX(dBm): 0
PowerRXThreshold(dBm): -95
macMinBE(0~Max: 3
macMinBE(0~Max): 3
TypeAlgorithm(0-non slotted, 1-slotted): 1
macMaxBE(3~8): 3
macMaxBE(3~8): 3
GA optimization setting
PopulationSize: 20
Generations: 5
Mutation_Pop: 20
Backoffs(0~5): 4
Backoffs(0~5): 4
Retries(0~7): 3
Retries(0~7): 3
GA Fitness Function or Script for Runing Simulations
/home2/nzhu/Desktop/MatlabGUIdemo/CC2420MICAzDisCal/MultiSimulation.
-----Topology & Connection-----
X-axis Position (meters): 1]
Y-axis Position (meters): 1]
input format: [1,2,3]

Console Window

```
Node3: CCA check-->channel is busy at time 65408 us
Node3: No free channel and Retry transmitting this data at time 65408 us
Node3: before CAF (_NB <= MAC_MAX_CSMA_BACKOFFS), will go back to BACKOFF and
proceed again at time 65599 us
Node3: ##### (BEACON_MODE)Enter into BACKOFF MODE at time 65600 us
Node3: (New superframe period and m_numBackoffInNewSuperframe == 0) generated a
random number as backoff slots, slots = 4 at time 65600 us
Node3:  slots !=0 and slots = 4 at time 65600 us
Node2: > evaluTimeSlot and proceed CCA case (TIME EVALUATION) at time 65919 us
Node2: ##### Enter into CCA case at time 65920 us
Node2: find a free channel with _CW = 2 at time 66048 us
Node2: find a free channel with _CW !=0, _CW = 1 and will proceed CCA again at time 66239 us
Node2: ##### Enter into CCA case at time 66240 us
Node2: find a free channel with _CW = 1 at time 66368 us
Node5: ACK waiting time Elapsed (_NR= 0) at time 66560 us
Node2: ##### (Enter into TX) starts to transmit the data to the transceiver (_NR= 0 ) with data
= 50 at time 66560 us
Node2: Will enter into A_W in next 1us at time 66560 us
Node0: ACK waiting time Elapsed (_NR= 0) at time 66561 us
```

Fig. 25 MATLAB-based GUI

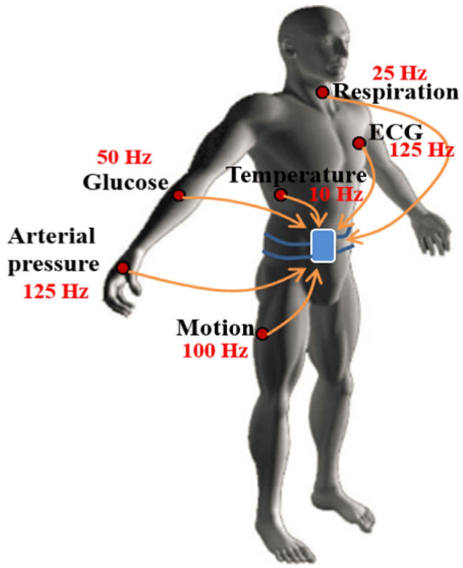


Fig. 26 Typical WBANs for medical and health care

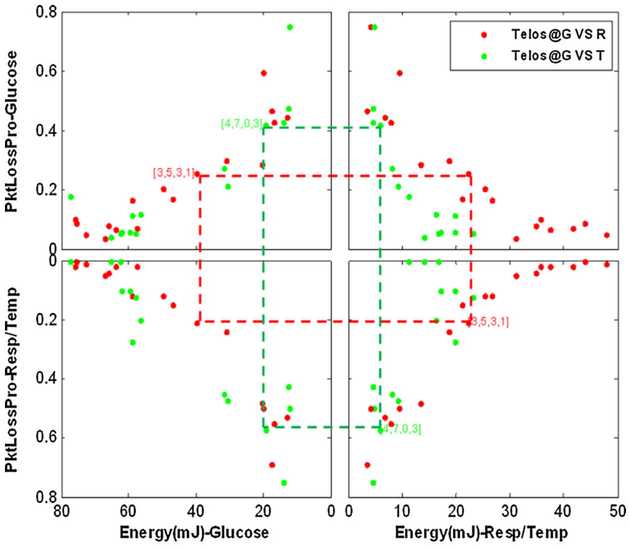


Fig. 27 Energy and packet loss Pareto front (Glucose vs Respiration/Temperature)

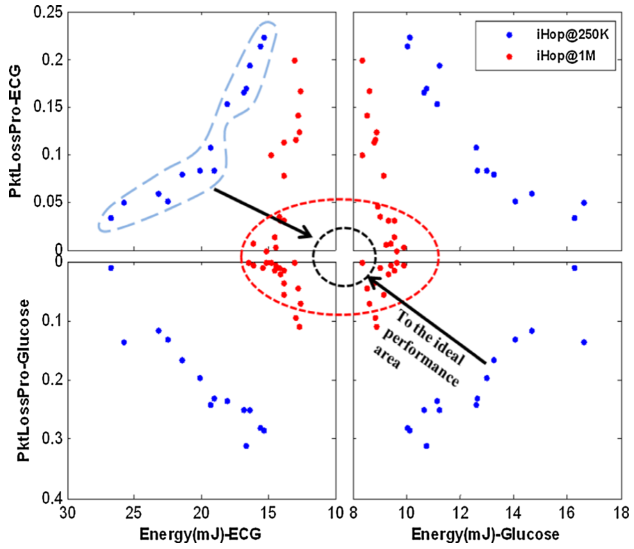


Fig. 28 Energy and packet loss Pareto front (ECG vs Glucose @ 1 Mb and 250 Kb)

where X is the parameter space of the unslotted algorithm (four parameters). Node1 and Node2 are different nodes selected from ECG, Arterial, Motion, Glucose, Respiration and Temperature sensor nodes. The multi-objective genetic algorithm was implemented by MATLAB, integrated into the EMrise_OpS framework and was used to generate the Pareto front for energy consumption and packet loss probability. The GA was configured with 60 initial integer individuals, 30 generations, scattered crossover, 0.8 crossover fraction as well as 2 elite children. Figure 27 shows the obtained four-dimensional Pareto front from the Telos mote based network.

In the Figure, the node energy consumption and network packet loss rate of the glucose sensor node are compared with the corresponding performance metrics of the respiration and temperature nodes respectively. According to the problem requirements, a good tradeoff solution for both energy and reliability can be manually selected by the decision maker based on their knowledge and intuitive experience (e.g., dashed rectangle in the Fig. 24). Algorithm configurations are also presented in the Figure ([*macMinBE, macMaxBE, macMaxFrameBackoffs, macMaxFrameRetries*] = [1, 3, 5]). On the other hand, do note that the best tradeoff solution for two specific cases might not guarantee the best tradeoff for the other two cases. Therefore, if an overall tradeoff solution is required for the whole network then a more comprehensive optimization on all the possible scenarios needs to be evaluated.

In another case study, the performance metrics (energy, reliability) of the highest sample rate based ECG node are also tested and compared with the glucose node by using iHop@Node. Two scenarios were under investigation, which were 1 Mbps based high data rate and 250 Kbps based low data rate. Figure 28 shows the 4D Pareto front.

The results in Fig. 28 show that the balance of tradeoffs between energy consumption and packet loss still needs to be found on the Pareto front for both data rate cases. As compared to the commonly used 250 Kbps low data rate, the high data rate based network can provide both lower energy consumption and packet loss, which brings it closer to the ideal performance area. This is because the high data rate reduces the channel occupation condition, so more packets can be successfully transmitted. For energy saving, this comes from two parts. Firstly, with good channel conditions under high data rate, mechanisms like repeated channel check and retransmission in the unslotted algorithm are no longer necessary, since almost all the packets can be successfully transmitted with one attempt, and energy is saved from protocol overhead. Secondly, despite the fact that a high data rate case consumes more current during the packet and ACK frame transmission, the time spent in both transmissions are greatly reduced due to the high data rate, and ultimately a large amount of energy can be saved from this part.

5 Conclusion

In this book chapter, the design and implementation of a house-build platform EMrise is presented for the energy management and evaluation of WSNs. Benefiting from SystemC-based simulation, EMrise is able to support energy consumption evaluation and exploration of heterogeneous sensor network at system-level and transaction-level. With a cost-effective and flexible hardware measurement platform, simulation/emulation models can be calibrated and verified for the accurate energy prediction. In addition, a generic genetic algorithm-based optimization framework is also integrated in the EMrise for the fast, multi-objective and multi-scenario energy optimization.

For the future better work, more detailed simulation models and communication protocols models will be integrated into EMrise for more realistic energy evaluation. The re-design of hardware platform in a smaller size will also be necessary for the practical use on more sensor node measurements. Besides, the integration of other optimization algorithms will help the exploration of more efficient methods of achieving energy-aware and manageable WSNs.

References

1. Lorincz, K., Chen, B., Challen, G. W., Chowdhury, A. R., Patel, S., Bonato, P., & Welsh, M. (2009). Mercury: A wearable sensor network platform for high-fidelity motion analysis. In *Proceedings of the 7th ACM conference on embedded networked sensor systems* (*SenSys'09*) (pp. 183–196)
2. Lloret, J., Bosch, I., Sendra, S., & Serrano, A. (2011). A wireless sensor network for vineyard monitoring that uses image processing. *Sensors, 11*(6), 6165–6196.
3. Yeh, L.-W., Wang, Y.-C., & Tseng, Y.-C. (2009). iPower: an energy conservation system for intelligent buildings by wireless sensor networks. *International Journal of Sensor Networks, 5*(1), 1–10.
4. Mikhaylov, K., Tervonen, J., Heikkila, J., & Kansakoski, J. (2012, April). Wireless sensor networks in industrial environment: Real-life evaluation results. In *2nd Baltic congress on future internet communications* (*BCFIC 2012*) (pp. 1–7).
5. Durisic, M. P., Tafa, Z., Dimic, G. & Milutinovic, V. (2012, June). A survey of military applications of wireless sensor networks. In *2012 Mediterranean conference on embedded computing* (*MECO 2012*) (pp. 196–199).
6. Fall, K. & Varadhan, K. (2011, November) The ns manual (formerly ns notes and documentation). http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf
7. OMNeT++ network simulation framework. http://www.omnetpp.org/.
8. Simon, G., Volgyesi, P., Maroti, M. & Ledeczi, A. (2003, March) Simulation-based optimization of communication protocols for large-scale wireless sensor networks. In *Proceedings of 2003 IEEE aerospace conference* (Vol. 3, pp. 1339–1346).
9. Levis, P., Lee, N., Welsh, M. & Culler, D. (2003) 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).
10. Polley, J., Blazakis, D., McGee, J., Rusk, D., & Baras, J. S. (2004, October). ATEMU: A fine-grained sensor network simulator. In *2004 first annual IEEE communications society conference on sensor and ad hoc communications and networks* (*IEEE SECON'04*) (pp. 145–152).
11. Titzer, B. L., Lee, D. K., & Palsberg, J. (2006, April). Avrora: Scalable sensor network simulation with precise timing. In *Processing of fourth international symposium on information sensor networks* (*IPSN'05*) (pp. 477–482).
12. Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., et al. (2005). *Tinyos: An operating system for sensor networks. Ambient intelligence* (pp. 115–148). Heidelberg: Springer.
13. Crossbow Technology Inc. MicaZ datasheet. http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf.
14. Barboni, L. & Valle, M. (2008, August). Experimental analysis of wireless sensor nodes current consumption. In *The second international conference on sensor technologies and applications, SENSORCOMM 2008* (pp. 401–406).
15. Landsiedel, O., Wehrle, K., & Gotz, S. (2005). Accurate prediction of power consumption in sensor networks. In *The second IEEE workshop on embedded networked sensors* (pp. 37–44).
16. Crossbow Technology Inc. MICA2 datasheet. https://www.eol.ucar.edu/rtf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf
17. Crossbow Technology Inc. TelosB datasheet. http://www.willow.co.uk/TelosB_Datasheet.pdf.
18. Texas Instruments Inc. (2013). ADC10664 datasheet. http://www.ti.com/lit/ds/symlink/adc10664.pdf.
19. Hurni, P., Nyffenegger, B., Braun, T., & Hergenroeder, A. (2011). On the accuracy of software-based energy estimation techniques. *Lecture notes in computer science* (Vol. 6567, pp. 49–64).
20. Hergenroder, A., Wilke, J., & Meier, D. (2010). Distributed energy measurements in WSN testbeds with a sensor node management device (SNMD). In *International conference on architecture of computing systems* (*ARCS*) (pp. 1–7).
21. Texas Instruments Inc. (2011). MSP430 datasheet. www.ti.com/lit/ds/symlink/msp430f1611.pdf.
22. Mackensen, E., Lai, M. & Wendt, T. M. (2012, October) Bluetooth low energy (ble) based wireless sensors. In *2012 IEEE Sensors* (pp. 1–4).
23. Zhang, J., Orlik, P. V., Sahinoglu, Z., Molisch, A. F., & Kinney, P. (2009). UWB systems for wireless sensor networks. *Proceedings of the IEEE, 97*(2), 313–331.
24. Buratti, C., Conti, A., Dardari, D., & Verdone, R. (2009). An overview on wireless sensor networks technology and evolution. *Sensors, 9*(9), 6869–6896.
25. Nanhao, Z. H. U., & O'Connor, I. (2013, July) Performance evaluations of unslotted CSMA/CA algorithm at high data rate WSNs scenario. In *The 9th IEEE international wireless communications and mobile computing conference* (*IWCMC 2013*) (pp. 406–411).
26. Castagnetti, A., Pegatoquet, A., Belleudy, C., & Auguin, M. (2012). A framework for modeling and simulating energy harvesting WSN nodes with efficient power management policies. *EURASIP Journal on Embedded Systems 2012, 1*, 8.
27. Ye, W., Heidemann, J., & Estrin, D. (2004). Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking, 12*(3), 493–506.
28. Van Dam, T., & Langendoen, K. (2003). An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems* (*SenSys'03*) (pp. 171–180).
29. IEEE Computer Society. (2006). Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANS). In *IEEE Std 802.15.4-2006*.
30. Accellera Systems Initiative. About SystemC. http://www.accellera.org/downloads/standards/systemc/about_systemc/
31. Prayati, A., Antonopoulos, Ch., Stoyanova, T., Koulamas, C., & Papadopoulos, G. (2010). A modeling approach on the TelosB

WSN platform power consumption. *Journal of Systems and Software, 83*(8), 1355–1363.

32. Agarwal, R., Martinez-Catala, R. V., Harte, S., Segard, C., & O'Flynn, B. (2008, August). Modeling power in multi-functionality sensor network applications. In *The second international conference on sensor technologies and applications, SENSORCOMM 2008*, (pp. 507–512).
33. Sivanandam, S. N., & Deepa, S. N. (2007). *Introduction to genetic algorithms, chapter 2 genetic algorithms* (pp. 15–36). Heidelberg: Springer Publishing Company, Incorporated.
34. Moteiv Corporation. (2006). Tmote sky datasheet. http://www.eecs.harvard.edu/konrad/projects/shimmer/references/tmote-sky-datasheet.pdf
35. Shimmer Research. Shimmer—wireless sensor platform for wearable applications. http://www.shimmer-research.com/
36. Zhu, N. (2013). Simulation and optimization of energy consumption on wireless sensor networks. Ph.D. Thesis, EEA Dept., Ecole Centrale de Lyon.
37. Zhu, N., & O'connor, I. (2013 April). Energy measurements and evaluations on high data rate and ultra low power WSN node. In *IEEE international conference on networking, sensing and control* (*ICNSC'13*) (pp. 232–236).
38. Nordic Semiconductor Inc. (2008). nRF24l01+ product specification. http://www.nordicsemi.com/eng/content/download/2726/34069/file/nRF24L01P_Product_Specification_1_0.pdf.
39. Fummi, F., Quaglia, D., & Stefanni, F. (2008, September). A SystemC-based framework for modeling and simulation of networked embedded systems. In *Forum on specification, verification and design languages* (*FDL'08*) (pp. 49–54).
40. Hiner, J., Shenoy, A., Lysecky, R., Lysecky, S., & Ross, A. G. (2010, June). Transaction-level modeling for sensor networks using SystemC. In *2010 IEEE international conference on sensor networks, ubiquitous, and trustworthy computing* (*SUTC'10*) (pp. 197–204).
41. Microchip Technology Inc. (2010). Microchip MIWI wireless networking protocol stack. Application Note. http://ww1.microchip.com/downloads/en/AppNotes/AN1066%20%20MiWi%20App%20Note.pdf.
42. Hauer, J.-H. (2009). Tkn15.4: An IEEE 802.15.4 MAC implementation for tinyos2. TKN technical report TKN-08-003. http://www.tkn.tuberlin.de/fileadmin/fg112/Papers/TKN154.pdf.
43. Nordic Semiconductor Inc. 2.4 GHz RF ultra low power 2.4 GHz RF ICS/solutions. http://www.nordicsemi.com/eng/Products/2.4GHz-RF.
44. Henderson, T. *Free space model*. http://www.isi.edu/nsnam/ns/doc/node217.html
45. Jovanov, E., Milenkovic, A., Otto, C., & de Groen, P. C. (2005). A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *Journal of NeuroEngineering and Rehabilitation, 2*(6), 1–10.
46. Ge, Y., Liang, L., Ni, W., Wai, A. A. P. & Feng, G. (2012). A measurement study and implication for architecture design in wireless body area networks. In *2012 IEEE international conference on pervasive computing and communications workshops* (*PERCOM'12*) (pp. 799–804).
47. Microchip Technology Inc. (2005) Pic16f87/88 data sheet. http://ww1.microchip.com/downloads/en/devicedoc/30487c.pdf
48. Microchip Technology Inc. PIC18F2525/2620/4525/4620 datasheet. http://ww1.microchip.com/downloads/en/devicedoc/39626b.pdf.
49. Future Technology Devices International Limited. (2010). FT232R USB UART IC. http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf.
50. Hergenroder, A., Wilke, J., & Meier, D. (2010, February). Distributed energy measurements in WSN testbeds with a sensor node management device (SNMD). In *23rd international conference on architecture of computing systems* (*ARCS'10*) (pp. 1–7).
51. Hergenroder, A., Horneber, J., Meier, D., Armbruster, P., & Zitterbart, M. (2009) Distributed energy measurements in wireless sensor networks. In *Proceedings of the 7th ACM conference on embedded networked sensor systems* (*SenSys'09*) (pp. 299–300).
52. Selavo, L., Zhou, G., & Stankovic, J. A. (2006, October). Seemote: In-situ visualization and logging device for wireless sensor networks. In *3rd international conference on broadband communications, networks and systems* (*BROADNETS'06*) (pp. 1–9).
53. Spekreijse, R. A communication class for serial port. http://www.codeguru.com/cpp/in/network/serialcommunications/article.php/c2483/Acommunication-class-for-serial-port.htm.
54. Morton, G. (2004). MSP430 competitive benchmarking. Application report. http://www.gaw.ru/pdf/TI/app/msp430/slaa205.pdf
55. Zhu, N., & O'Connor, I. (2013). iMASKO: A genetic algorithm based optimization framework for wireless sensor networks. *Journal of Sensor and Actuator Networks., 2*(4), 675–699.
56. Sivanandam, S. N., & Deepa, S. N. (2007). *Introduction to genetic algorithms, chapter 10 applications of genetic algorithm* (pp. 317–402). Heidelberg: Springer Publishing Company, Incorporated.
57. Chipperfield, A., Fleming, P., Pohlheim, H., & Fonseca, C. Genetic algorithm toolbox user's guide. http://crystalgate.shef.ac.uk/code/manual.pdf.
58. Ferreira, F. F. (2012). Architectural exploration methods and tools for heterogeneous 3D-IC. Ph.D. Thesis, EEA Dept., Ecole Centrale de Lyon.
59. Hussain, S., Matin, A. W., & Islam, O. (2007). Genetic algorithm for hierarchical wireless sensor networks. *Journal of Network, 2*(5), 87–97.
60. Sudha, N., Valarmathi, M. L., & Neyandar, T. C. (2011). Optimizing energy in WSN using evolutionary algorithm. In *IJCA proceedings on international conference on VLSI, communications and instrumentation* (*ICVCI'11*) (Vol. 12, pp. 26–29).
61. Liu, J.-L., & Ravishankar, C. V. (2011). LEACH-GA: Genetic algorithm-based energy-efficient adaptive clustering protocol for wireless sensor networks. *International Journal of Machine Learning and Computing, 1*(1), 79–85.
62. Bhatti, M. S., Kapoor, D., Kalia, R. K., Reddy, A. S., & Thukral, A. K. (2011). Rsm and ann modeling for electrocoagulation of copper from simulated wastewater: Multi objective optimization using genetic algorithm approach. *Desalination, 274*(1), 74–80.
63. Frantz, F., Labrak, L., & O'Connor, I. (2012). 3D IC floorplanning: Automating optimization settings and exploring new thermal-aware management techniques. *Microelectronics Journal, 43*(6), 423–432.
64. Latre, B., Braem, B., Moerman, I., Blondia, C., & Demeester, P. (2011). A survey on wireless body area networks. *Wireless Networks, 17*(1), 1–18.

Nanhao Zhu is currently a senior researcher and expert on communication network, sensor network and internet of things at CETC Group, GCI Science and Technology Co., LTD. Canton, China. He completed his Ph.D. in Wireless Sensor Networks at Ecole Centrale de Lyon, France, in 2013. He also received his M.S. and B.S. degrees from Communication University of China in 2010 and 2008, respectively. His research interests include wireless sensor networks, wireless body area networks, embedded system, future telecommunication technologies and data visualization. He has also served as the reviewer for some international conferences.

Athanasios V. Vasilakos is currently a professor with the Lulea University of Technology, Sweden. He has authored or coauthored over 200 technical papers in major international journals and conferences. He is author/co-author of five books and 20 book chapters in the area of communications. He has served as general chair/technical program committee chair for many international conferences. He served or is serving as an editor and/or guest editor for many technical journals, such as the IEEE Transactions on Network and Service Management, the IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, the IEEE Transactions on Information Technology in Biomedicine, the IEEE Transactions on Computers, the ACM Transactions on Autonomous and Adaptive Systems, IEEE JSAC special issues of May 2009, January 2011, and March 2011, IEEE Communications Magazine, ACM/Springer Wireless Networks (WINET), and ACM/Springer Mobile Networks and Applications (MONET). He is founding editor-in-chief of the International Journal of Adaptive and Autonomous Communications Systems (IJAACS) and the International Journal of Arts and Technology (IJART). He is general chair of the Council of Computing of the European Alliances for Innovation.