Intel Serv Robotics (2009) 2:139–151
DOI 10.1007/s11370-009-0040-0

ORIGINAL RESEARCH PAPER

Real-time control architecture using Xenomai for intelligent service robots in USN environments

Byoung Wook Choi · Dong Gwan Shin · Jeong Ho Park · Soo Yeong Yi · Seet Gerald

Received: 22 February 2009 / Accepted: 8 April 2009 / Published online: 30 April 2009

Abstract This paper describes the implementation of a dual-kernel software architecture, based on standard Linux and real-time embedded Linux, for real-time control of service robots in ubiquitous sensor network environments. Mobile robots are used in active service for the assisted living of elderly people, monitoring their mental and physiological data with wireless sensor nodes. The data collected from sensor nodes are routed back to a sink node through multi-hop communication. The moving sink node installed on the main controller of the robot collects data and transmits it to the main controller. To be able to handle emergency situations, the robot needs to satisfy real-time requirements when processing the data collected, and invoking tasks to execute. This paper realizes a multi-hop sensor network and proposes real-time software architecture based on Xenomai. The real-time tasks were implemented, with priority, to rapidly respond to urgent sensor data. In order to validate the deterministic response of the proposed system, the performance measurements for the delay in handling the sensed data transmission and the trajectory control with a feedback loop were evaluated on the non real-time standard Linux.

Keywords Service robot · Real-time embedded Linux · Wireless sensor and actor network · Assisted living

B. W. Choi (✉) · S. Y. Yi
Department of Electrical Engineering, Seoul National University of Technology, 172 Gongreoung 2-dong, Nowon-gu, Seoul 139-743, South Korea
e-mail: bwchoi@snut.ac.kr

S. Y. Yi
e-mail: suylee@snut.ac.kr

D. G. Shin
Robotstar Co., Ltd., 119-38, Sasa-dong, Sangnok-gu, Ansan, Gyeonggi-do 426-220, South Korea
e-mail: shingun@robostar.co.kr

J. H. Park
DasaRobot Co., Ltd., 401 Yakdae-dong, Wonmi-gu, Bucheon, Gyeonggi-do 420-734, South Korea
e-mail: jhpark@dasarobot.com

S. Gerald
School of Mechanical and Aerospace Engineering, Nanyang Technological University, Nanyang Ave., Singapore 639798, Singapore
e-mail: mglseet@ntu.edu.sg

1 Introduction

Recent advances in wireless communication, embedded systems, and ubiquitous computing have enabled the development of small sensor nodes that are of low cost, low power, and multifunctional. A ubiquitous sensor network (USN) is composed of a large number of sensor nodes which sense, collect, correlate, and aggregate data, concurrently. The ability of a USN to sense phenomenon such as temperature, light, pressure, or movement makes it a promising sensing technique for many applications. USNs have been used in numerous applications including military applications, environmental monitoring, health applications, home automation, and smart environments [1–4].

Due to the problem of an aging society, there are proposals to use a USN to aid senior citizens in hospitals and silver towns. As the population ages, there will be an increasing demand on health care resources. Approximately one-third of all health care expenditures are directed to the population over 65. Almost 25% of this population have one or more deficits in capability, necessary for successful daily life, and reside in a customised or assisted living facility [5]. New and improved devices and applications are expected to emerge. UIUC has suggested the I-Living System, which uses open software and hardware to provide daily life

support for elderly people [6,7]. They are involved in the process of developing a wireless-based software infrastructure with sensing, localisation, monitoring, wireless communication, and event/data management. AlarmNet, a system which provides healthcare service by monitoring the vital signs and activity patterns of residents in a home via sensors, has also been proposed [8]. A Health Smart Home was designed by the Faculty of Medicine to validate the feasibility of such a system [9].

Realization of these USN applications requires wireless ad hoc networking techniques. Although many protocols and algorithms have been proposed for USNs in recent years, they are not well suited for the unique features and application requirements of sensor networks [1]. Since the sensor nodes are usually scattered and densely deployed in a sensor field, multi-hop communication is typically adopted by using neighbour nodes. Data collected in sensor nodes are routed back to a central entity. The central entity performs the functions of data collection, coordination and transmission to central controller, e.g. human interaction. Sometimes, the central entity is referred to as a *sink node*. The sink node collects and transmits the data to the main controller for in-depth analysis such as monitoring, alarming, and forming a database. This type of architecture is referred to as semi-automated architecture since the sink node collects and transmits data to the main controller, where the appropriated action is initiated [10].

There is much ongoing research on utilizing sensor networks to advance the development of mobile service robots. A wireless sensor network can extend the sensory perception of people and robots far beyond their normal range. Although there have been some research efforts related to USNs with robots, most of them are focused on navigation and mapping algorithms. None of the studies reported were related to the conduct of active services utilizing USNs. The mobile robots can perform more active service based on the data collected by sensor networks, such as responding to emergency situations. The proposed application in this paper is an example of an integrated sensor/actor node since the robot is capable of both sensing and acting. This is sometimes referred to as a wireless sensor and actor network (WSAN), as it consists of a group of sensors and actors linked by a wireless medium to perform distributed sensing and acting tasks. The design of a WSAN featuring node mobility has been investigated for control applications [11]. A typical example of integrated sensor/actor nodes in WSANs is the autonomous battlefield robot called the *robotic mule* [10]. In another WSAN effort, a mobile sensor platform for a mobile distributed sensor network for real-time target detection, recognition, and tracking was developed by Sandia National Laboratories [12].

In this paper, a sink node was deployed into a main robot controller and became the moving sink node used for aggregating sensor data. The robot should act on its own sensor readings as well as the sink node data received from the network. Coordinating the behaviour of the mobile robot was an important issue as the tasks that need to be performed evolve with time. For instance, when applied to active assisted living, coordinated actions should be initiated to react as soon as possible to events detected by sensors, as in the monitoring of an elderly person's behaviour, such as a fall [3,11]. Thus, real-time control architecture is strongly required for active service and to achieve real-time communications to remote care providers [13].

In order to achieve real-time requirements, real-time operating systems (OS) are needed. The proprietary OS provides an integrated development environment, a multi-tasking environment, and third party tools. As a consequence of dramatic improvements in hardware computing power and free software quality, Linux can currently be considered as a soft real-time OS. Recently, the possibility of using standard Linux for embedded real-time applications in robotics was evaluated [14]. However, standard Linux is not suitable for hard real-time applications as required in feedback control or reactive control. There are two approaches to satisfy real-time performance in the Linux domain. One is a dual-kernel approach, and the other is a fully pre-emptive Linux kernel with a real-time scheduler. Open-source Linux extensions based on the dual-kernel approach are mainly RTAI [15] and Xenomai [16]. Most commercial real-time embedded Linux is based on fully pre-emptive Linux.

Even though both approaches are useful to achieve real-time performance, the focus of this paper is on the open source approach, implying the dual-kernel approach. In the dual-kernel approach, a standard Linux kernel is run as the lowest priority task by the microkernel and Linux system calls can be used. Both the RTAI and Xenomai were worthy of consideration for the proposed application. Xenomai was chosen as it had a better structure, available for a larger number of platforms, and provides a set of emulation layers which may prove useful when porting to large systems [16]. Xenomai is a real-time development framework that cooperates with the Linux kernel in order to provide a pervasive, interface-agnostic, hard real-time support to user-space applications seamlessly integrated into the GNU/Linux environment. In performance measurements on four different OSs, Barbalace et al. demonstrated that open-source software such as RTAI and Xenomai are suitable for hard real-time applications [17]. The open and modular approach that uses Xenomai over the IEEE 1394 real-time device driver has been described and its real-time performance has been evaluated for a mobile robot platform [18].

The proposed system is a WSAN, where the service robot is an actor that provides assisted living service while the mobile sink node aggregates the sensing data. This paper documents the following engineering R&D tasks of smart

sensing, wireless transmission, monitoring, and real-time computing for the service robots working in USN environments.

- A network routing algorithm of multi-hop from sensor nodes to a moving sink node was implemented by user data defined in TinyOS data packet structures, for lower power consumption.
- A real-time control architecture, based on a dual-kernel approach (Xenomai + standard Linux), was developed to provide real-time response by prioritizing tasks during emergency situations for the user.
- A real-time serial interface between the sink node, embedded in the robot controller, and the main robot controller was implemented to provide real-time handling of the sensing data aggregated from sensor fields.
- A prototype of an assisted living system for the elderly was implemented on a mobile robot, to prove the feasibility of the proposed system.
- The real-time performance was examined for the proposed architecture depending on the workload in handling sensor data in a WSAN as well as in applying the tracking control for the mobile robot.

2 System structure

A service system for assisted living people in USN environments needs sensor nodes to sense phenomenon, a sink node to aggregate data, and send them to the main controller. The main controller is implemented on a service robot for active service, to react to emergency situations, and serve as a remote monitoring system. An efficient network routing algorithm from the sensor nodes to the sink node is required, together with a real-time software architecture to process data in order of task priority.

In order to meet such requirements, Fig. 1 shows the overall structure of active services for elderly people with service robots in USN environments. The proposed system consists of *sensor fields*, an integrated sink node, and a remote monitoring server. The sensor nodes are usually scattered to sense phenomenon in *sensor fields*. The role of sensor nodes is to collect data and route it back to the sink node. The integrated sink node is a mobile robot with a sink node installed to collect data. The robot should perform appropriate actions based on the collected data; for example, moving to the sensor node where an event is triggered and monitoring the behaviour of elderly people using a camera installed on the mobile robot. All actions are performed in the tasks scheduled by the real-time operating system. The robot controller also transmits collected information to a remote monitoring server via wireless LAN for other services, such as alerting a doctor or to monitor the people remotely.

Whenever an event occurs, event information always passes through the sensor nodes within one hop from the sink. Thus, these sensor nodes may have a higher load of relaying packets. However, the robot, an integrated sensor/actor node, is able to move into the event area. This implies that relaying sensor nodes also reconfigure in response to each event. The relaying load becomes evenly distributed between all nodes. As a result, since event information is transmitted locally through sensor nodes around the event area, sensors that are far from the event area do not function as relaying nodes, resulting in network resource conservation by the proposed application. Therefore, wireless sensor networks with mobile sinks have many advantages over static sensor networks [18].

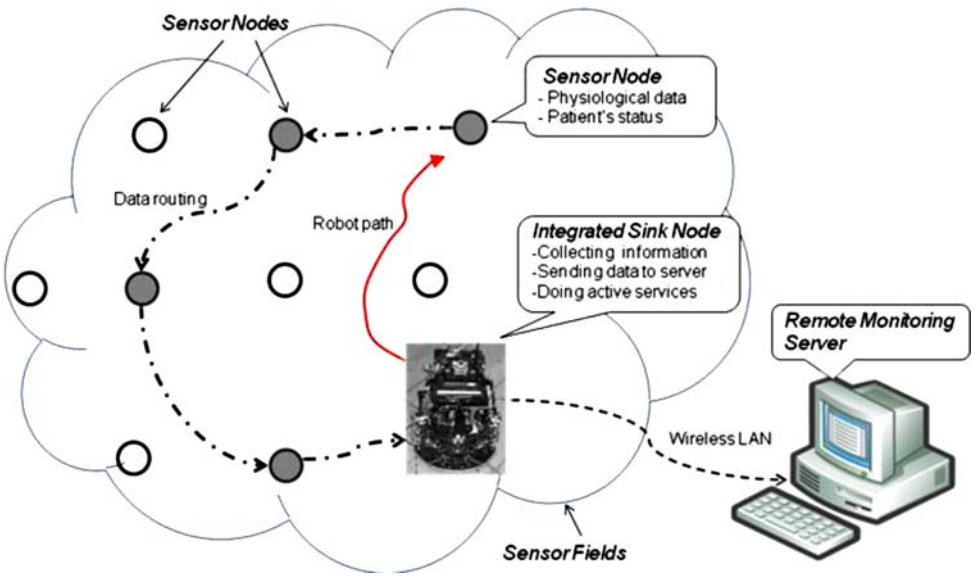

Fig. 1 System structure of active services for the elderly people

The sensor node used consisted of a control board called the H-mote and a daughter sensor board, as shown in Fig. 2 [19], and also two microprocessors: a micro-controller and a RF chip. The processing unit of the H-mote was an 8 MHz TI MSP430 micro-controller. The open-source operating system TinyOS was used [20]. The RF chip was a CC2420 with 2.4 GHz IEEE 802.15.4 compliant RF transceiver. The H-mote provided the connection to a sensor board and a USB interface for integrating other controllers. The logical connection, however, was achieved by the RS-232 interface. Therefore, the same sensor control board could be used for both sensor nodes and the sink node.

In order to provide more active service, a mobile robot was needed. The robot utilized in this paper is shown in Fig. 3. The robot had various sensors to perform autonomous navigation, such as ultrasonic sensors, position sensor devices (PSD), a CMOS camera, and a localizing sensor. A wireless communication device was also installed to communicate with the remote monitoring system. An Intel/IBM compatible i386 single board computer was used as the main controller. In this case, a sink node could be interfaced to become a mobile sink node. The sink node transmitted data to the main board through the USB interface.

The data gathered on the control board is processed according to predetermined priorities. For example, vital signs such as the pulse rate are processed first, while other less important data are processed later. When the system determines an emergency situation has occurred, the robot approaches the user to capture visual images and send them to the system. The collected sensor and visual data of the elderly, on urgent occasions, are sent to the monitoring system using a broadband WLAN communication network. For such a system to perform efficiently, real-time processing of the data is essential. If the data processing is delayed due to rescheduling of the operating systems, a quick response cannot take place, leaving the user in a dangerous situation.

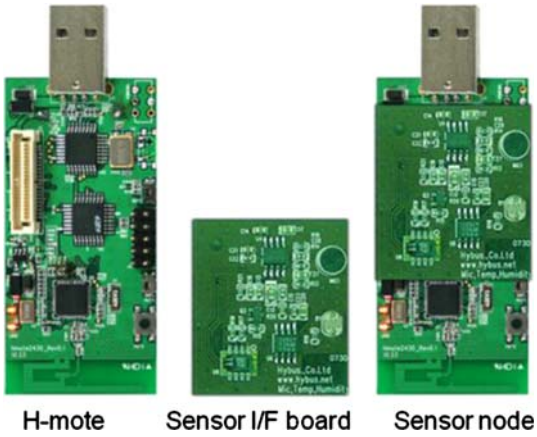

Fig. 2 A sensor node consisting of the H-mote and the sensor I/F board

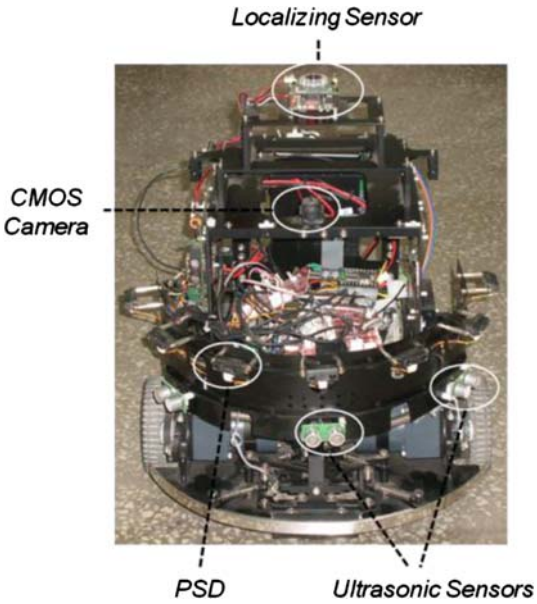

Fig. 3 A two-wheeled mobile robot

3 Multi-hop network

This paper focuses on a specific type of WSAN. The deployment diagram of the proposed system is shown in Fig. 4. The system consists of a group of sensors and a mobile sink node where the data are aggregated periodically and transferred to the robot controller. A multi-hop networking technique would clearly be convenient for the distributed network nodes in trying to locate the most efficient route between the sensors and the sink node. In this section, a routing algorithm is implemented based on a set of ad hoc data packets using TinyOS. The implementation is based on the TinyOS SurgeTelos routing algorithm [21]. It only supports the IEEE 802.15.4 PHY/RF and part of the MAC for the CC2420 RF device. It was modified so that the mobile sink node would be required to update the periodic routing.

The configuration of the network routing between the sensor nodes and the sink node must be set before the transmission of the data. This means that the network routing must be fixed according to the movement of the elderly people or the mobile robot. TinyOS includes library components that provide ad hoc multi-hop routes for sensor network applications and uses a packet structure for network routing and data transmission. In TinyOS, a message buffer defined as *TOS_Msg* is depicted in Fig. 5. The buffer contains a header, data frame, and packet metadata. *TOS_Msg* is a fixed size structure whose size is defined by the maximum data; the default is 29 bytes.

In this paper, two separate packets were used in the network routing and data transmission functions, thus enabling

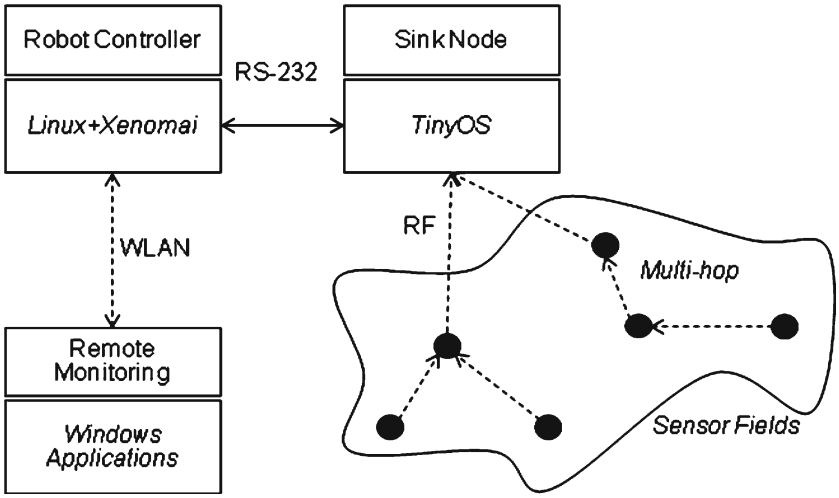

Fig. 4 The deployment diagram of the proposed system

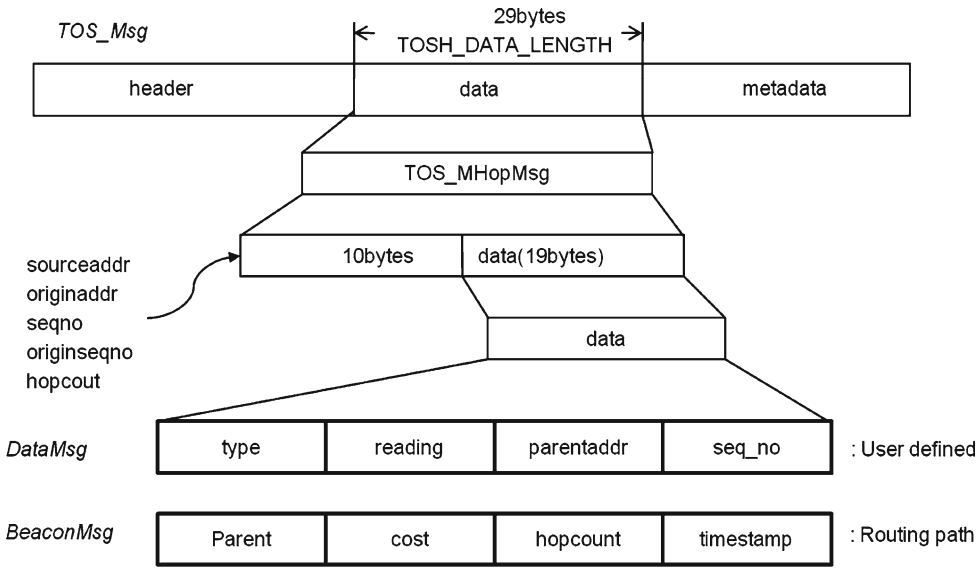

Fig. 5 TinyOS packet structure

efficient data processing. *BeaconMsg* was used for network routing with the sink node, and *DataMsg* was used to transmit data from the sensor nodes. The periods of time for messages with *BeaconMsg* and *DataMsg* were 4,096 and 800 ms, respectively. In Fig. 5, *Parent* is the ID number of the parent node of the receiving node, *cost* describes the link cost from the sink node to the parent node, and *hopcount* represents the number of hops from the sink node to the receiving node. The receiving node configured the data route to the sink node with such information. In the other direction, *DataMsg* was a message structure used to transmit data from the sensor node to the sink node.

The routing algorithm provided by the TinyOS uses minimum cost forwarding to set the route with the minimum link cost between the nodes. The link cost is calculated using the received signal strength indication (RSSI) and link quality indication (LQI). Figure 6a shows the diagram of the algorithm based on the minimum link cost. In the diagram, each circle indicates nodes, and values on the links represent the cost required to transmit a data packet through the related link. The number 0 is the sink node which collects the data, 6 is the sensor node that senses phenomenon, and numbers 1–5 are the relay nodes within the network route transferring the data to the next node. In semi-automated architecture, whenever an event occurs event information always passes through the sensor nodes within multi-hop to the sink node. Thus, these sensor nodes become the relay nodes. Naturally, there is no distinct difference between relay and sensor nodes, and any node can be the sensor node when sensing data.

Each node goes through initialization for network routing. Besides the sink nodes, other nodes cannot create *BeaconMsg*, which is used independently for network routing. The algorithm to find the minimum energy route is described in Fig. 6b. When the sink node transmits *BeaconMsg*, other sensor nodes around the sink node receive the data and

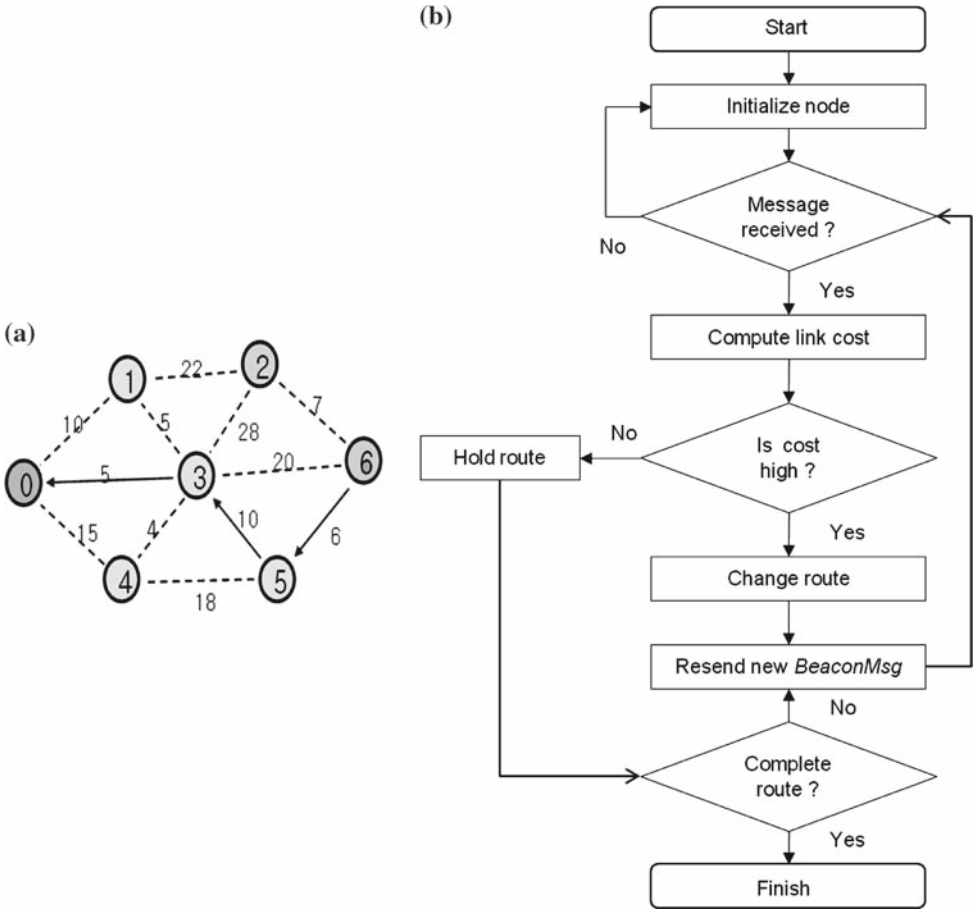

Fig. 6 The algorithm of network routing for minimum energy consumption. **a** The measured cost of power on the routes. **b** The flowchart of the routing algorithm

compute the link cost to the parent node. The evaluated cost for the receiving node is compared with that of the previous one, and the route is established for the one which costs less. After this process, *BeaconMsg* is adjusted and sent again; this process is repeated until the data reach the final node. By using such a process, the data route with minimum link cost can be found, and sensor nodes are able to transmit data to the sink node in the form of *DataMsg* packets. For example, in Fig. 6a the data route from 6 to 0 with the multi-hop of 5 and 3 was established as the minimum energy route.

In this application, the sensor nodes are static, whereas the actuator node, or the mobile robot, is moving. Since the sink node installed in the mobile robot is also moving, it becomes a mobile sink node. Even though sink mobility can improve network performance in areas such as energy efficiency and throughput, the network routing needs to be reconfigured when the robot moves. Bi et al. [18] suggested an autonomous moving strategy for mobile sinks in data-gathering applications. The movement of the robot requires quick and regular reconfiguration of the network routing, but such frequent reconfiguration reduces the transmission rate. Therefore, an adequate time distribution depending on the speed of the robot is needed to prevent this. Along with these routing results, the position of the sensor node used by the user can be found, giving the robot access to the position where the event occurred.

4 Real-time control architecture using Xenomai

In recent years, dramatic improvements in hardware computing power and free software quality have generated much interest in the possibility of using standard Linux for embedded real-time applications in robotics. Laboratory tests showed that standard Linux could be used for embedded real-time robotics [14]. Moreover, a new implementation of the scheduler has been provided in Linux kernel 2.6. However, standard Linux is not yet suitable for hard real-time applications as required in feedback control, especially in trajectory control for robotics. Therefore, a way of adding to Linux the possibility of defining real-time tasks that ensures control within a deterministic response time needs to be considered. This feature is provided by the open-source Linux extension Xenomai. The real-time extension of Xenomai is achieved through a dual-kernel approach. Therefore, it is necessary for the hardware resources to be shared by Linux and the additional component. This was achieved through the

adaptive domain environment for operating systems (Adeos) [22]. To implement this requirement, several patches of Linux, Adeos, and Xenomai were ported to the target board.

4.1 Dual-kernal approach

The implementation to allow deterministic response times regardless of the standard Linux implementation is described in Fig. 7. Adeos is a resource virtualization layer available to run several operating systems on a single hardware platform. Adeos enables multiple entities called domains to exist simultaneously on the same machine. These domains do not necessarily see each other, but all of them see Adeos. Two domains were ported on the same hardware platform: One was the real-time embedded Linux Xenomai 2.3, and the other was the non real-time standard Linux 2.6.17. Kernel space application was handled first depending on the priority, and the standard Linux kernel was scheduled with the lowest priority. The Adeos interface was directly exposed to the hardware abstraction layer (HAL) underlying the Xenomai core. Therefore, most of the requests for Adeos services were issued from HAL allowing predictable interrupt latencies in the lowest micro-second level range to Xenomai no matter what the standard Linux was undergoing.

As shown in Fig. 7, Xenomai allows running real-time applications both in kernel space and user space. All threads managed by Xenomai can be identified from the real-time nucleus. Adeos ensured that events were dispatched in an orderly manner to the various client domains according to their respective priority. The events were incoming external interrupts, system calls issued by Linux applications, and other system events triggered by the kernel code. Xenomai threads were running over the context of the highest priority domain, but the Linux kernel was considered as the lowest priority domain. Therefore, the real-time applications could be performed in the context of Xenomai as well as standard Linux-based application as the non real-time applications, insuring the convenience and extension of development. These are the main advantages when using Xenomai for real-time applications. Functions developed in Linux can be used without any modification.

4.2 Real-time architecture for the sink node

In the proposed architecture, the mobile service robot becomes an actor integrated with the sink node in a WSAN that processes all incoming data and performs active service. The sink node gathers the data from the sensor nodes and analyses it before issuing it to the actor node. Therefore, the following roles are required in addition to intrinsic service robot functions:

- Sink Node: Gather the data received from the sensor node and transmit them to the main controller.
- Active Agent: Provide an active service of the robot depending on the gathered data for the user.
- Gateway: Transmit the data to remote monitoring system via WLAN for further use.

In order to perform such tasks, the service robot must be equipped with the proper computing devices. For this study, the Intel/IBM compatible i386 single board computer (SBC) was chosen as the main controller for the robot, with standard Linux and real-time Linux Xenomai running together as described in Fig. 7. For the sink node installed on the mobile robot, the CC2420 was used to receive data from the sensor nodes and to transmit data to the main controller. The sink node needs to route the network frequently to provide a minimum energy data transmission in a multi-hop communication environment. In addition, the sink node must gather data sent by sensor nodes, and then send the data to the actor in order of importance in real-time.

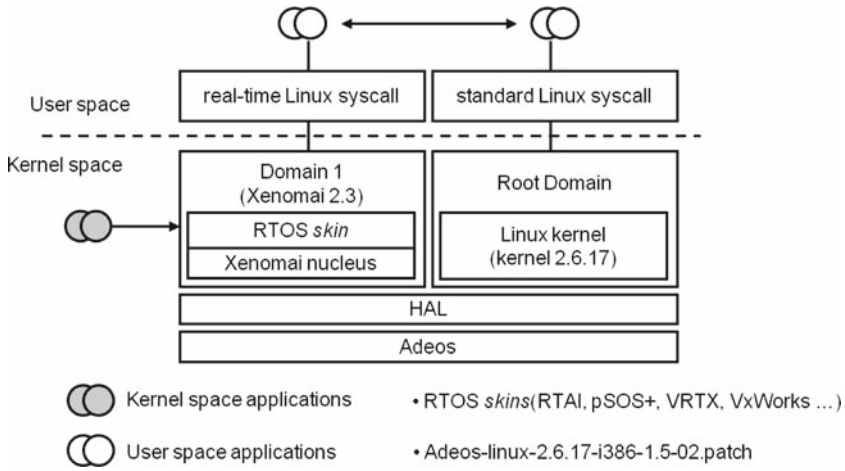

Fig. 7 Real-time control software architecture based on Xenomai

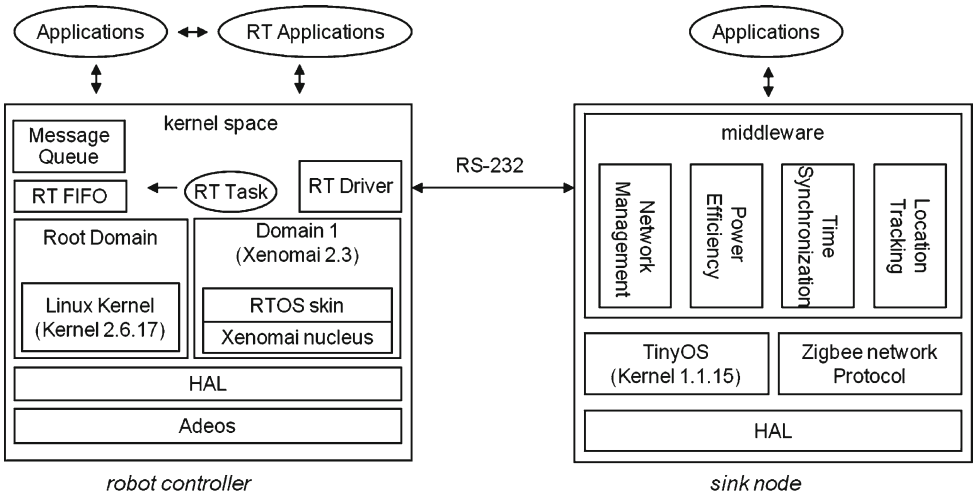

Fig. 8 The real-time sink node architecture

The service robot in the proposed system exchanged information with the rest of the WSAN through serial transmissions of the main controller and the sink node. The data in the sensor nodes are transmitted to the main controller using the RS-232 interface. Therefore, a real-time (denoted RT for device driver or task) device driver needed to be realized for the serial devices in Xenomai platform. The real-time driver model (RTDM) provides a unified interface of real-time user-space applications and the hard real time system [16,23]. Xenomai was the first real-time Linux extension to support the new RTDM revision. It acts as a mediator between an application requesting a service from a certain device and the device driver offering it. HAL is an optional indirection layer which may be added when further abstraction is required. In the Xenomai project, real-time device drivers have been ported for open-source projects such as RT_COM for serial transmission, RT_NET for network performance over Ethernet, RT_CAN for CAN (Controller Area Network) controllers, RT_FireWire for IEEE1394 specification, etc. [22,24]. Blocking capability of a function was ruled out to ensure real-time performance. Xenomai comes with a reference driver for the UART 16550A chips conforming to the serial profile. In this paper, RT_COM was applied as a real-time serial driver to the motor control system. RT_COM was used to interface with other sensor devices in the robot as well.

The real-time sink node architecture can be seen in Fig. 8, fulfilling the requirements for active services in a WSAN. To perform the real-time data collecting and processing, the real-time software architecture based on Xenomai was ported on the SBC, the RT device driver of RT_COM was implemented to communicate to the sink node through the RS-232 interface, RT tasks in kernel space were designed to process incoming data and conduct real-time control with high priority, and finally, both RT applications and NRT applications were implanted to do other functions in the user space with low priority. The data communication between tasks from the kernel space to the user space was achieved with RT FIFO (real-time FIFO mechanism) and message queue. In terms of hardware, the sink node and sensor nodes have the same structure, and both nodes use TinyOS to efficiently manage limited resource. TinyOS is an appropriate operating system to provide an efficient performance within a limited hardware memory capacity, since it is an event-based status switching operating system.

5 Implementation and performance measurement

For performance measurement, the proposed system was implemented with the minimum functions. The purpose of this experiment was to show the feasibility of a dual-kernel approach for the service robot in a USN environment in the sense of real-time performance. The focus was on the real-time performance for both handling data gathered from the WSAN and performing trajectory control. The number of RT and NRT tasks and the levels of priorities depend upon the application of the service robot. Therefore, the practical temporal constraints of RT tasks were not handled. In general, interrupt latency and rescheduling time should be considered for evaluating real-time performance. The performance measurements for the measured delays for VxWorks, Xenomai, RTAI and Linux have been addressed by Barbalace et al. [17]. Therefore, measurements were concentrated on the performance for the effect of context switching in terms of the number of tasks running simultaneously for both real-time Linux and standard Linux.

5.1 Realization of the real-time control software

The implementation to allow deterministic response based on Xenomai is shown in Fig. 8. For the real-time realization, Xenomai 2.3 was applied to the Adeos architecture. The standard Linux of kernel 2.6.17 was ported to the main board for

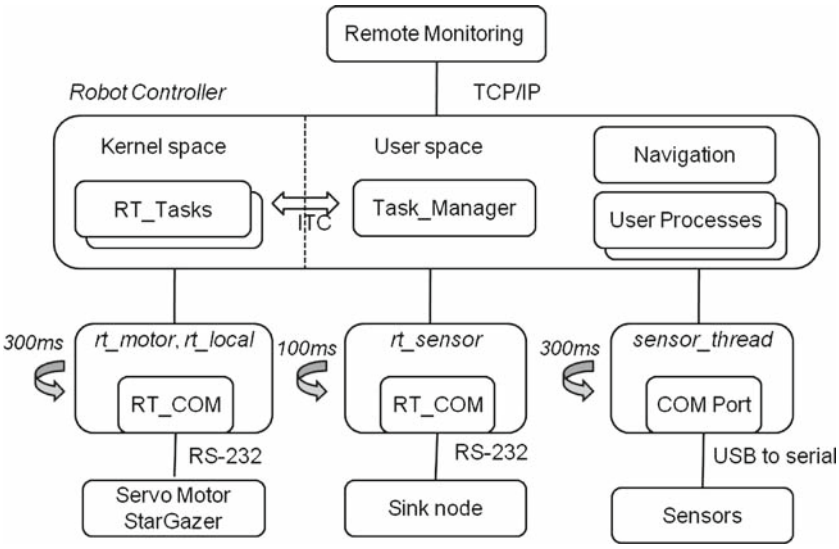

Fig. 9 The realization of real-time software in the robot

general application software as the root domain, and the sink node used CC2420 to route data back from the sensor nodes to the main controller. The data transfer between the sink node and the main controller was performed over RS-232 serial communication.

For the mobile robot, the implemented robot tasks were simple because the robotics-related algorithms were not the scope of this paper. The robot had various sensors such as ultrasonic sensors and PSDs, but these were connected with a USB-to-serial converter due to the lack of serial ports of the controller used in this paper. Therefore, the set of tasks was implemented in the standard Linux as threads. For autonomous navigation to the position of the elderly (for example, finding the position of the elderly people using the position of the issued sensor), a StarGazer [25] was adopted. The interface for the motor control board and localisation sensor was serial communication so that real-time tasks through RT_COM for achieving motor control and handling position data were developed.

The robot was mainly used as the actor. The software modules needed to meet the requirement for the mobile robot were simply displayed as several processes in user space. The simplified overall implementation in the robot is shown in Fig. 9. The general inter-task communication (ITC) mechanisms were used to achieve communication between RT tasks and NRT standard threads. The tasks requiring real-time performance, such like *rt_motor, rt_local* and *rt_sensor*, were implemented in kernel space with real-time priority. Localisation based on the StarGazer needed to be updated within specified periodic time intervals to conduct trajectory control. In order to rapidly respond to urgent sensor data handle, the real-time tasks needed to be implemented with priorities for the sink node. Other tasks related with general applications of the robot were implemented in user space as standard Linux threads. Even though user processes can be implemented in kernel space, debugging and extensions with other off-the-shelf libraries are very difficult. Therefore, the dual-kernel approach is useful to implement complicated robot applications. For NRT tasks such as *sensor_thread* and user processes, there is no guarantee they will be executed within the specified period.

5.2 Performance measurement of handling sensor data

This measurement aimed to show the efficiency of RT tasks in handling the sink node data through RT_COM. It was not focused on evaluating the efficiency for multi-hop communication. For this experiment, a test program *produce_sine* was created in a sensor node, which processed the sampled data in a 100 ms range to transmit 180 degrees for one cycle of a sine curve of magnitude 4. Although very simple, this program allowed the testing of features of interest. Since the data were periodically sampled and gathered in the main controller, the sink node gathering data from the sensor node transmitted data to the main board through the RS-232 interface at a speed of 57,600 bps. As described in Fig. 9, the RT task *rt_sensor* was designed as the periodic task of 100 ms to read the received data through RT_COM. The received data were sent to *data_task* running in user space by message queue where data were processed into a plot. *nrt_data_task* was used to evaluate non real-time performance. *nrt_data_task,* implemented in user space as standard Linux application, reads data through a standard serial interface.

The measured times are reported in Fig. 10. For RT tasks, it took approximately 18 s to draw the data from the time the data were received from the sensor node to handing the data in user space to *data_task,* regardless of the number of tasks running simultaneously. The reason for such a result was that when the RT serial device driver was used to process the data and the priority of the handling task

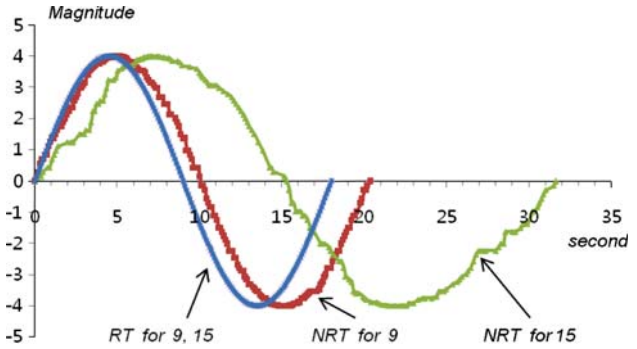

Fig. 10 Measured performance for the delay according to the number of workload tasks

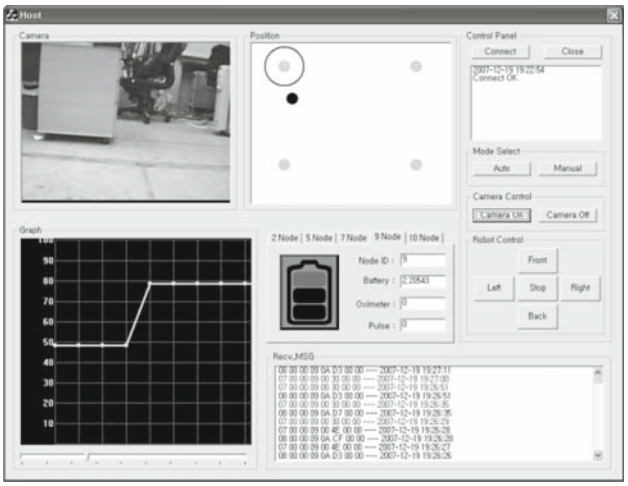

Fig. 11 GUI of the remote monitoring system

was high, it prevented context switching by standard Linux processes. However, for NRT performance the measured time increased when the workload increased. In the proposed software architecture, the data processing was performed in the necessary time even if the workload increased. On the other hand, in the NRT environment the data processing revealed time delays depending upon the number of workload tasks.

5.3 The remote monitoring system

Another important part of the software is the graphical user interface (GUI) of the remote monitoring system, which is executed on either Windows or Linux (see Fig. 11). In order to use sensor data to form a health record database for the user, to gain a visual image of the user during an emergency situation, or to move the mobile robot to a certain location according to a situation, a monitoring management system is required. This consists of three graphical windows and many other control views. The graphical data view can be used to display data such as the pulse signals of the person. Through the camera view transmitted from the robot, the visual status of the elderly person can be examined. Also, other information about the WSAN can be shown, such as the remaining battery power for each node and communication packets.

Information on the position of the sensor nodes is useful as the robot is able to move to the sensor node that issued sensor data near the person. The demonstration of the unique feature of an active service is displayed in the top-middle window labelled "Position". The four grey dots in the window show the position of each node, and the circled dot indicates the position of the elderly person. When the sensor node triggered data to indicate the abnormal status of the person, the data were transmitted to the sink node. Next, the data were relayed to the remote monitoring system by the robot controller through TCP/IP as shown in Fig. 9. Using the routing information within the relay nodes and position data of each node allowed the robot to move to the position of the sensor node. Another mode is tele-operation. The robot can be interactively controlled to move to another position and determine the condition of the person through voice chatting or visual camera by using control buttons made in GUI.

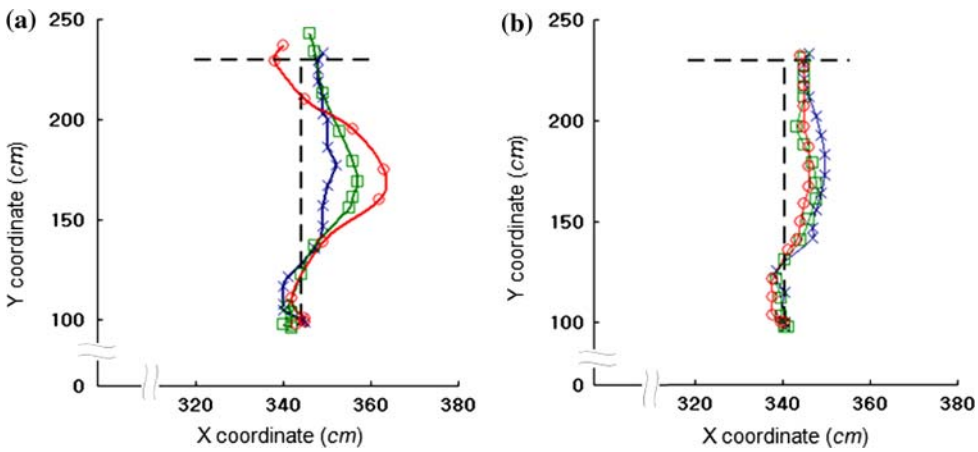

Fig. 12 The trajectories of the robot in Cartesian space. **a** Measured performance for NRT tasks. **b** Measured performance for RT tasks

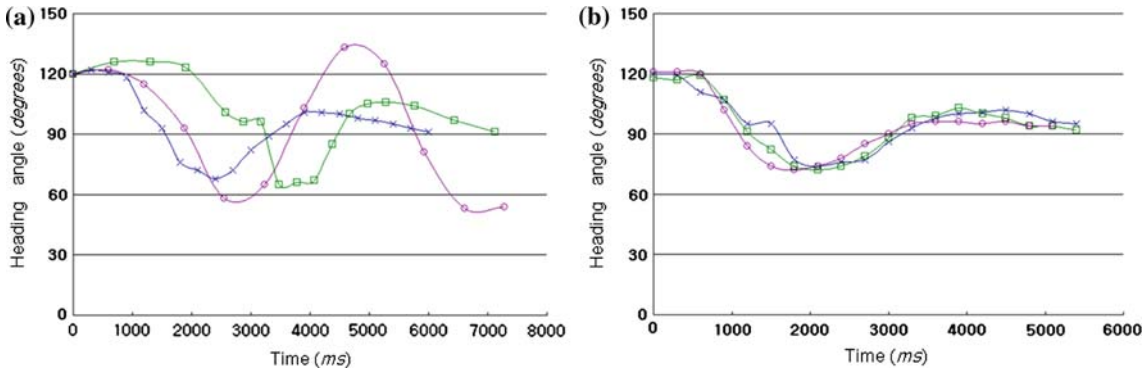

Fig. 13 Heading angles of the robot. **a** Measured performance for NRT tasks. **b** Measured performance for RT tasks

5.4 Performance measurement of controlling the robot

In the first round of tests, the focus was on the difference in delay time when handling sensing data in RT and NRT software environments. In this case, the real-time performance for aggregating sensor data could be examined even though those were gathered from the sensor nodes. Here, the control performance was put into focus by driving the robot straight forward. Since the robot was an actor in this study, the control performance was also important in performing the active services. The mobile robot was controlled from the Cartesian coordinates (344, 100) to (344, 230), measured in cm, while heading angle was targeted from 120 degrees to 90 degrees. In the robotics of mobile robots, the position and heading angle are major control arguments. The feedback control loop should be performed with a control loop time of 300 ms to follow the reference command. Complicated algorithm was not adopted as the purpose of this test was to verify the real-time characteristics, not to show the performance of the control algorithm. Although Bruzzone et al. [14] evaluated the possibility of standard Linux for embedded real-time application in robotics and showed that a large variety of applications could be implemented, the current version of Linux is not yet suitable for hard real-time applications, as required in feedback control and for this case.

In Figs. 12–14, all results are reported according to the number of workload tasks, where symbols denote the number of workload tasks (multiplication symbol: no other task, open square: 5 tasks, open circle: 7 tasks). Figure 12 shows the robot trajectories in Cartesian space, Fig. 13 describes the heading angle of the robot in the trajectory, and Fig. 15 indicates the performed control loop cycle. Table 1 shows the detailed results. The control loop cycle time in Fig. 14a varied when workload increased. The delay of the control loop in Fig. 14b created the delay in total execution time shown in Fig. 13b. On the other hand, the control loop in Fig. 14b was maintained at 300 ms. Results imply that the proposed architecture is feasible for hard real-time control,

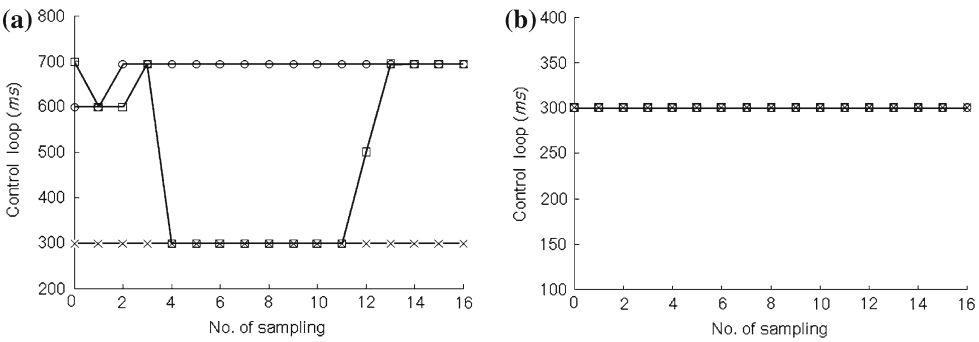

Fig. 14 The sampling times in the control loop of the robot. **a** Measured performance for NRT tasks. **b** Measured performance for RT tasks

Table 1 Tracking control performance for NRT tasks depending on workload tasks

No. of tasks	Actual distance/reference distance (cm)	Working time (ms)	Actual time/reference time (ms)
0	132/130	6,300	300/300
5	143/130	7,803	300–600/300
7	140/130	7,960	500–600/300

as required in feedback control. The trajectories and heading angles of the robot were incorrectly controlled in the case of a NRT software environment, as shown in Figs. 12a and 13a. The results shown in Figs. 12b and 13b maintained the control goal regardless of the number of workload tasks, even though the trajectory did not follow the reference command exactly. The final control error resulted from the slippery floor and the poor resolution of localization sensors adopted in this paper. This is another problem that was beyond the scope of this paper; improving the control performance itself is a topic for future work.

6 Conclusion

This paper proposed a real-time control software architecture for mobile robots in WSANs for health care or assisted living services for elderly persons. A WSAN consists of a robot, a sink node fitted onto the robot controller, and sensor nodes to sense phenomenon such as temperature, pressure and movement. The robot became an actor for performing active services, and the sink node collecting the sensing data became a moving sink node. To facilitate the moving sink node, the network routing algorithm for multi-hop was conducted to find the minimum energy cost using the TinyOS. The real-time software architecture was based on a dual-kernel approach, where Xenomai and standard Linux are running together on top of the Adeos environment. In order to interface the sink node, the real-time serial device driver was also ported. For realization of the health care system, a remote monitoring system was introduced without an explicit server. As system calls can be used with two kernels, the functions can be easily performed for the gateway to the outside, the controller with real-time performance, and the sink node.

In order to verify the feasibility and real-time performance, two features were focused upon: sensor network data handling, and control performance of the robot with a feedback loop robot corresponding to the number of workloads. Based upon the performance measurement results, the following was observed:

- The performance of the proposed system, in terms of delay, was good at both handling sensing data transmission and performing trajectory control with the feedback loop.
- Both a real-time task and real-time serial device were required to satisfy real-time performance.
- Xenomai was better structured and available for a larger number of platforms; the flexibility of the proposed system was applicable to both real-time and non real-time applications.
- The robot performed well as the moving sink node and the actor in the WSAN.

Though there were many research initiatives involving sensor networks introduced [1], more development should be encouraged in solutions to living assistance for elderly persons with mobile robots in a WSAN as follows:

- Real-time requirements and decomposition of RT and NTR tasks should be considered.
- Robot control problems, including localization, should be overcome.
- Non-tactile sensors to sense the psychological data of the elderly person must be developed.
- More practical services implemented in the context of WSANs are needed for commercial solutions, such as visual tracking, voice chatting, and rich databases for clinical purposes.

Acknowledgments This work is the outcome of a Manpower Development Program for Energy & Resources supported by the Ministry of Knowledge and Economy (MKE).

References

1. Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E (2002) Wireless sensor networks: a survey. Comput Netw 38:393–422. doi:10.1016/S1389-1286(01)00302-4
2. Manley ED, Deogun JS (2007) Location learning for smart homes. In: Proceedings of IEEE international conference on advanced information networking and application workshops, pp 787–792
3. Hou J, Wang Q, Ball L, Birge S, Caccamo M, Cheah CF, Gilbert E, Gunter C, Gunter E, Lee CG, Karahalios K, Nam MY, Nitya N, Rohit C, Sha L, Shin W, Yu Y, Zeng Z (2007) PAS: a wireless-enabled, sensor-integrated personal assistance system for independent and assisted living. In: Proceedings of joint workshop on High Confidence Medical Devices, Software, and Systems (HCMDSS) and Medical Device Plug-and-Play (MD PnP) interoperability (HCMDSS/MD PnP'07)
4. Diamond SM, Ceruti MG (2007) Application of wireless sensor network to military information integration. In: Proceedings of IEEE international conference on industrial informatics, pp 317–322
5. Berenson UI (2006) Public policy lecture: quality, chronic care, and developments in physician payments. In: Presented in Am Geriatric Soc annual meeting
6. Assisted living project. http://lion.cs.uiuc.edu/assistedliving/technical.html
7. Wang Q, Shin W (2006) I-Living: an open system architecture for assisted living. In: Proceedings of IEEE International Conference on Systems Man and Cybernetics (ICSMC '06), pp 4268–4275
8. AlarmNet. http://www.security.honeywell.com/hsce/solutions/alarmnet/index.html
9. Noury N, Herve T, Rialle V, Virone G, Mercie E, Morey G, Moro A, Porcheron T (2000) Monitoring behavior in home using a smart fall sensor. In: Proceedings of IEEE-EMBS special topic conference on microtechnologies in medicine and biology, pp 607–610
10. Akyildiz IF, Kasimoglu IH (2004) Wireless sensor and actor: research challenges. Ad Hoc Netw 2:351–367. doi:10.1016/j.adhoc.2004.04.003
11. Xia F, Tian YC, Li Y, Sun Y (2007) Wireless sensor/actuator network design for mobile control applications. Sensors 7:2157–2173. doi:10.3390/s7102157

12. Sandia National Laboratories. Visited in March 2008 http://www.sandia.ov/isrc
13. Oh S, Schenato L, Chen P, Sastry S (2007) Tracking and coordination of multiple agents using sensor networks: system design, algorithms, and experiments. In: Proceedings of the IEEE, vol 95, pp 234–254
14. Bruzzone G, Caccia M, Ravera G, Bertone A (2009) Standard Linux for embedded real-time robotics and manufacturing control systems, robotics and computer-integrated manufacturing. Corrected Proof, available online 15 January 2008 (in press)
15. Home Page RTAI. http://www.rtai.org
16. Xenomai Home Page. http://www.Xenomai.org
17. Barbalace A, Luchetta A, Manduchi G, Moro M, Soppelsa A, Taliercio C (2007) Performance comparison of VxWorks, Linux, RTAI and Xenomai in a hard real-time application. In: Proceedings of IEEE-NPSS real-time conference, pp 1–5
18. Bi Y, Sun L, Ma J, Li N, Khan IA, Chen C (2007) HUMS: an autonomous moving strategy for mobile sinks in data-gathering sensor networks. EURASIP J Wirel Commun Netw 1–15. doi:10.1155/2007/64574
19. Hybus Home Page. http://www.hybus.net
20. Tiny OS. http://www.tinyos.net
21. Serge Telos. http://www.tinyos.net/tinyos-1.x/apps/SurgeTelos
22. Adeos Home Page. http://home.gna.org/adeos
23. Kiszka J (2005) The real-time driver model and first applications. In: Proceedings of 7th real-time Linux workshop
24. Kiszka J, Wagner B, Zhang Y, Broenink J (2005) RTnet—a flexible hard real-time networking framework: In: Proceedings of 10th IEEE international conference on emerging technologies and factory automation
25. Home Page HAGISONIC. http://www.hagisonic.com/