Control of Mobile Robots Through Wireless Sensor Networks

Ricardo S. Souza[1], Lucio Agostinho[1], Fábio Teixeira[1], Diego Rodrigues[1]
Leonardo Olivi[2], Eliane G. Guimarães[2] and Eleri Cardozo[1]

[1]School of Electrical and Computer Engineering, Unicamp, Campinas, SP, Brazil

[2]Information Technology Center Renato Archer, Campinas, SP, Brazil

Abstract. *A trend in today's mobile robotics is to perform the control of mobile robots in a distributed fashion. This control scheme requires a wireless communication network connecting the robot to the other existing devices in the environment. WiFi and Bluetooth are examples of such networks. Ad-hoc networks such as wireless sensor networks (WSN) can be deployed instantly and offer a wide coverage by employing a large number of communicating nodes. In addition to provide a communication link to the robots, a WSN can aid the robot's navigation, localization, and also enhance its sensory capabilities. This paper presents a network architecture for supporting the control and communication of mobile robots through WSNs.*

Keywords*: Wireless Sensor Networks, Networked Robotics, Mobile robotics.*

1. Introduction

Wireless sensor networks (WSN) are ad-hoc networks composed of small multifunctional nodes that are able to communicate among them in short distances [Akyildiz et al. 2002]. Each node has limited memory and processing power and can have multiple sensing capabilities (e.g., temperature, light, humidity and acceleration). A structured WSN has its sensor nodes placed in well known positions, usually employing a regular geometry. An unstructured WSN has its nodes deployed in arbitrary geometries.

WSNs are becoming widespread for purposes such as area monitoring, precision agriculture, and wild life protection, usually providing extensive field coverage. WSNs can also be deployed instantly in order to manage unforeseen situations such as natural or man-provoked disasters. Mobile robots are interesting complements to WSNs. In fact, WSNs and mobile robotics have been combined for different purposes. A WSN can extend the mobile robots' sensory capabilities by supplying the mobile robots with information about the environment. This information can be employed by the mobile robots for localization [Peng et al. 2009], mapping, navigation [Popa et al. 2009] [Yao and Gupta 2010], mission planning, among other purposes. In addition, mobile robots can enhance WSN applications by deploying, replacing and acquiring data from sensor nodes [Tekdas et al. 2009], supplying the location of sensor nodes [Sichitiu and Ramadurai 2004], and providing gateway services to the sensor network.

We are particularly interested in deploying mobile robots on environments already monitored by unstructured WSNs. The mobile robots are equipped with sensor nodes and are capable of communicating with the network. Our main objective is to use already deployed WSNs to access and control the robots.

Our starting point is a platform for networked robotics developed by our group and reported in [Cardozo et al. 2010]. Firstly, we evaluate the restrictions imposed by WSNs on networked robotics applications in cases when the robots must rely solely on this network for control and communication purposes. Examples of such restrictions include bandwidth and message size restrictions, and routing restrictions. Then we present the design and implementation of WSN capabilities added to this platform. The design preserves the platform main features such as security, access from multiple programming languages, and low overhead. Evaluations of the overheads were conducted in both simulated and real environments. We compare the platform's performance using WSN and WiFi networks and finally present a networked robotics application as a case study.

The paper is organized as follows. Section 2 presents a brief review of the integration of mobile robots and WSNs as related works. Section 3 introduces our mobile robotics platform. Section 4 presents the platform extensions for operating through WSNs. Section 5 presents performance evaluations comparing the platform overheads when operating over wireless sensor and WiFi networks, and describes an application of mobile robotics over WSNs. Finally, Section 6 concludes the paper and presents some current and future works.

2. Related Works

There are many possibilities for integrating mobile robotics and WSNs. One of the most usual application is to apply WSNs for aiding the navigation of mobile robots.

A solution presented in [Popa et al. 2009] performs motion control of a mobile robot based on information gathered from WSNs, through a sensor node attached to the robot that controls its navigation. This node communicates with another node attached to a PC through the WSN. The PC monitors other nodes in the environment and receives the robot's position and motion. The robot sends periodic beacons to allow the WSN to track its position. The robot position is calculated using a trilateration method. This implies that at least three WSN nodes must know their positions as well as their distances to the robot. This distance is inferred from RSSI (Radio Signal Strength Indicator).

In [Yao and Gupta 2010] a different solution for the navigation problem is presented. A distributed planning framework is proposed where "obstacles" detected by the WSN are taking into account in determining feasible paths for mobile robots. In other words, a roadmap is generated by the wireless sensor network to guide robots through the environment. Sensors must be equipped with laser rangefinders capable of detecting obstacles and build a map of their sensing region. The local maps are combined to form a map of the monitored area. This map is then used to generate roadmaps for the robot. To combine the local maps created by the sensors the environment must be equipped with landmarks, known as relay points. These relay points are later used to stitch the local maps together.

An indoor navigation system for autonomous mobile robots using WSN is presented in [Fu et al. 2009]. This system is based on a pre-deployed sensor network with at least three nodes per room with known locations. These nodes communicate with the robot transmitting their positions. The robot is then capable of determine its own position using RSSI.

WSNs are also applied to solve localization problems. In [Peng et al. 2009], a

system that calculates the localization of a moving target traveling along an area covered by a WSN. In this scenario a mobile robot is equipped with a sensor node, turning it into a mobile sensor node to the WSN. It is assumed that all static nodes of the sensor network have known positions. When the robot moves through the WSN its position is determined using trilateration based on the position of the static nodes and RSSI readings between the mobile and the static nodes.

Another possibility when combining WSN and mobile robots is to use the robot to collect or aggregate information from sensor nodes. A system that uses mobile robot to harvest information from sensor nodes is presented in [Tekdas et al. 2009]. This solution becomes interesting when monitored areas are far from each other and their information need to be combined. In this scenario the use of a mobile robot that can collect information from one area and then move to the next area to collect the information is more interesting (e.g., in terms of energy savings) than deploying nodes to connect the two regions.

A mobile node moving within a WSN can also be used to localize the nodes of a deployed network, as reported in [Sichitiu and Ramadurai 2004]. In this work a mobile robot equipped with a sensor node performs random walks within a WSN. The robot sends periodic beacons with its position that are received by the nearby nodes. Using these beacons the sensor nodes are capable of inferring the distance to the robot and use such distances to construct and maintain their own position estimates.

In [Meger et al. 2009] a different system for mapping a WSN is presented. In this case, several sensor nodes equipped with cameras are deployed on the environment without communication among the nodes. A mobile robot, equipped with a sensor node and a known landmark that the cameras are able to detect, navigates through the environment communicating its position to the nodes. Using the landmark, the nodes are able to determine the distance between them and the robot, and with the robot's position information, the nodes are able to determine their own positions.

Solutions for the integration of WSN and mobile robotics usually are developed to solve specific problems in specific scenarios. Navigation strategies employing WSNs usually rely on the fact that the positions of all network nodes are well known or can be inferred. The solutions for the localization problem often employ RSSI readings, which are well documented as unreliable in dynamic environments, to determine node or robot positions [Menegatti et al. 2009] [Peng et al. 2009].

3. A Networked Robotics Platform

The WSN architecture presented in this paper is based on a networked robotics platform that allows control of mobile robots over the network [Cardozo et al. 2010]. The platform employs open protocols, services, and security solutions adopted by modern distributed applications to grant access to mobile robots through the Internet and private networks. The platform is divided into four basic packages: Front-End, Protocol Handler, Management and Embedded, as shown in Figure 1.

The Embedded package aggregates microservers that run on the robot's internal processors. Although large mobile robots have enough processing power to run full featured servers, this is not the case for small and mid sized robots. Low processing power imposes constraints in some design decisions regarding the microservers. Real time, security, and interaction with the robot's hardware are examples of such constraints.

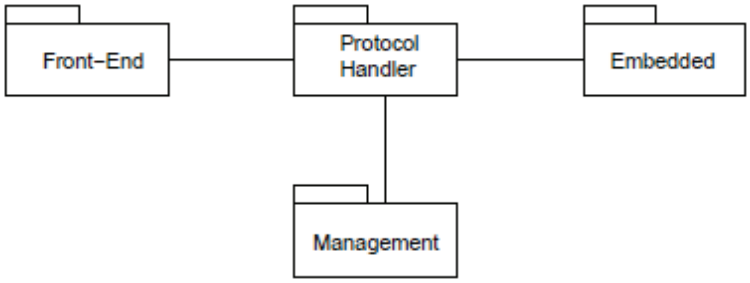

Figure 1. The packages of the networked platform and their relationships.

On the client side, applications interact with the microservers using well known protocols, such as HTTP (Hypertext Transfer Protocol) and SIP (Session Initiation Protocol). These protocols are favored for three reasons: i) they are simple, text-based protocols; ii) proxies simplify the processing of these protocols at the server-side; iii) they have powerful client applications available (e.g., Web browser for HTTP and Internet communicators for SIP).

In our communication model the client-microserver interaction is based on the REST (Representational State Transfer) interaction pattern [Richardson and Ruby 2007]. REST employs messages to inspect and change the state of an object maintained by a Web server. The object is a data structure representing a logical entity (e.g., a database entry) or physical entity (e.g., a robot). Messages can be based on any protocol, HTTP and SOAP (Simple Object Access Protocol) being the most used ones. Our platform favors REST messages directly over HTTP for simplicity on the microserver side. As such, messages already defined for HTTP are employed in the interaction, mainly GET, PUT, POST and DELETE for retrieving, creating, updating, and removing object states, respectively.

Another important aspect is the interaction of the microservers with the robot's hardware. Usually a robotic framework such as ARIA [Mobile Robotics 2010] or Player [Player 2010] is used as an intermediator. These frameworks are responsible for all interaction with the robot's hardware. For instance, an HTTP POST message carrying a new state of the robot (e.g., a new pose) must be translated into instructions that can be interpreted by the robot. Robotic frameworks usually are called from C/C++. In order to process a request, the microserver calls the proper C/C++ functions, collects the returned data, and assembles an HTTP response to the client carrying a XML (Extensible Markup Language) document as reply.

The robot operations available for the users, through a HTTP microserver, are divided into different categories of services. Table 1 shows the categories of services that a typical mobile robot requires.

In order to facilitate the access to the robotic resources, the Front End package supplies APIs (Application Programming Interface) for several programming languages (Java, C/C++, Python, C#, Matlab, and LabView). The APIs generate HTTP requests and parse the XML reply transparently to the clients. An overview of the platform components

Service	Definition
Move Service	Handles movement related operations. (e.g., setVel, setHeading, getPosition)
Range Service	Handles sensory related operations. (e.g., getSonarReadings, getLaserReadings)
Video Service	Handles video and image capture related operations. (e.g., getPicture, getVideo)
Camera Service	Handles camera configuration related operations. (e.g., setPan, setTilt)
Info Service	Handles robot information operations. (e.g., getRobotSizes, hasGripper)

Table 1. Platform service categories.

is shown in Figure 2. All interaction with the robot is mediated by the Proxy Agent and the Security Agent (part of the Protocol Handler package). These components allow communication only from authenticated and authorized users and applications.

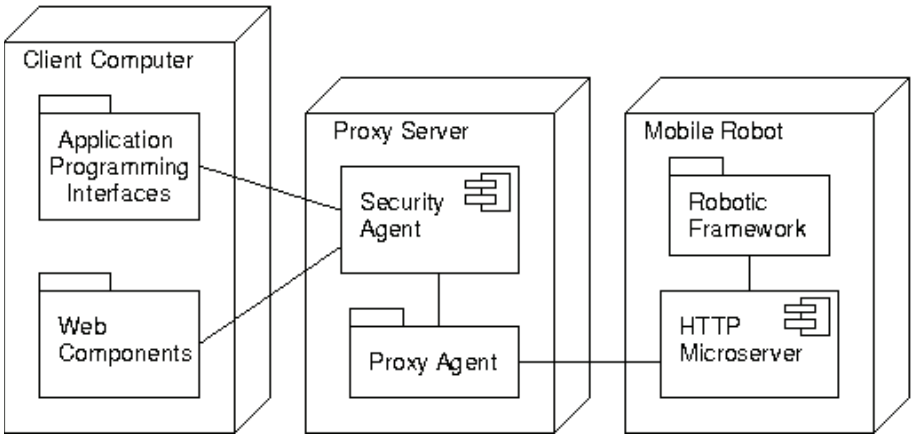

Figure 2. The components of the networked robotics platform. Management components are not shown.

4. Wireless Sensor Network Architecture

Last section introduced a mobile robotics platform that grants access to robotic resources over a network, more specifically, the robot's communication model within the platform. This section presents a WSN extension to provide communication between the robot and the platform server over IEEE 802.15.4, instead of IEEE 802.11 (WiFi).

In order to change the radio technology from 802.11 to 802.15.4 some aspects must be taken into account. Since sensor nodes are small devices with limited processing, storage, and power capabilities, several constraints are imposed. One important limitation is the severe reduction on package payload size. For instance, on the TinyOS [TinyOs 2010] implementation of the 802.15.4 standard the package payload must not exceed 28 bytes. At the same time, the change in technology must remain transparent to both client and robot sides. This implies that the communication model presented in Section 3 (XML over HTTP) must remain unchanged. Meanwhile, the WSN

extensions must be able to provide at least the robot's main functionalities. We assume that the robot's main functionalities are provided by the Move, Range and Info services (Table 1).

The solution relies on two WSN gateways, one running on the platform's proxy server, and a second one running on the mobile robots as shown in Figure 3. These gateways communicate through a WSN protocol.

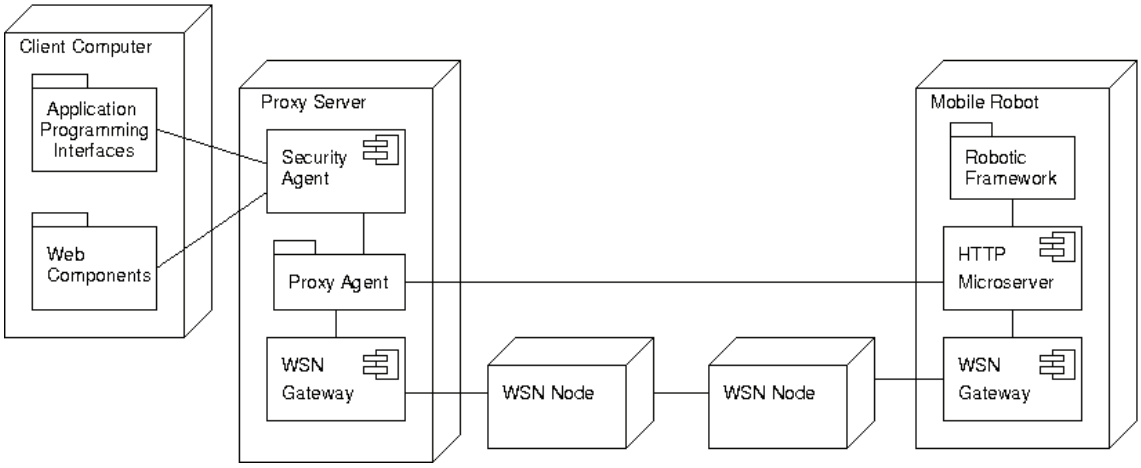

Figure 3. The platform extended architecture for WSN support.

4.1. The WSN Gateways

The WSN gateways are responsible for providing the major functionalities of the networked robotics platform through a WSN. At the platform side, the WSN gateway is implemented as a Java HTTP microserver with the same CGI (Common Gateway Interface) interface as the microserver running on the robot. The Proxy Agent redirects an HTTP request to the robot or to the WSN gateway, according to the address mapped for the resource (e.g., request to resource */PioneerRobot* can be proxyed to robot's microserver or to the WSN gateway at the same node).

At the robot side, the WSN gateway is a Java program that reads messages sent to the robot's WSN node. If the message requires an operation on the robot (e.g., to set an speed), the gateway generates an HTTP request to the robot's microserver and forwards the reply to the client (in this case the gateway running on the platform side). If the operation does not involve robot operations (e.g., the reading of a nearby sensor node), the gateway performs the operation directly through the WSN.

4.2. The WSN Protocol

The limit on message length imposed by the WSN (28 bytes in our case) demands a protocol able to fragment the messages commonly carried through HTTP and formatted in XML. Common requests and replies fit in a single WSN message. For instance, to set a pose of the robot it is required two integer numbers (coordinates) and a floating point number (orientation). Setting operations return a single true/false response indicating success/failure. Range sensor readings, on the other hand, return a large amount of data: apart from the robot pose, for each sensor it contains its position in a local (robot's)

reference frame, the obstacle detected in both local and global (inertial) reference frames, and the absolute distance read. For example, a robot with a laser rangefinder able to perform 180 readings, results in a 16 Kbytes XML document with the readings data.

Clearly, XML is not a practical format for WSNs due to its low efficiency in terms of information coding. For WSNs, we replaced the platform's XML-based protocol in benefit of a more efficient protocol. The protocol defines message types for requests and replies. Each request message contains two fixed fields (message type and operation), plus a set of variable length fields with the operation parameters. The number and length of parameters are inferred from the operation type. Listing 1 shows the format of a request message in the NesC [Gay et al. 2003] notation.

```
typedef nx_struct OperationRequestMsg {
        nx_uint8_t msgType;
        nx_uint16_t operation;
        nx_int16_t par1;
        nx_int16_t par2;
        nx_int16_t par3;
} MoveMsg;
```

Listing 1. Format of a request message generated by the WSN gateway at the platform side.

As an example, Listing 2 shows the command to read all the sixteen sonars of the robot as submitted to the WSN gateway (top) and the corresponding generated message to be carried over the WSN to the gateway running on the robot (bottom).

```
http://RobotWSN/range.cgi?op=getSonarReadings&range=0:15:1

Message <OperationRequestMsg>
  [msgType=0x2]
  [operation=0x21]
  [par1=0x0]
  [par2=0x10]
  [par3=0x1]
```

Listing 2. Range request submitted to the platform and the corresponding WSN request message.

Reply messages are defined for each type of operation. Most reply messages fit on a single WSN packet, e.g., operations returning a true/false indication, a single numerical quantity (speed, orientation, etc.), or a few quantities (e.g., robot pose). In case of multiple sensor readings, the protocol defines two messages, a control message and a sensor reading message. Once the WSN gateway at the robot side receives a XML document containing the sensor readings, it parses this document and computes the number of readings. It then generates a control message shown in Listing 3.

```
typedef nx_struct ControlMsg {
        nx_uint16_t numMsgs;
        nx_int16_t robotPosX;
        nx_int16_t robotPosY;
        nx_int16_t robotPosTh;
} ControlMsg;
```

Listing 3. Control message format.

The control message carries the robot's pose when the readings were taken and the number of sensors read. After the control message, the WSN gateway at the robot

side generates sensor messages containing only the sensor's ID and distance reading, plus a message ID, as shown in Listing 4.

```
typedef nx_struct RobotSensorMsg {
        nx_uint8_t msgId;
        nx_uint8_t id;
        nx_uint16_t reading;
} RobotSensorMsg;
```

Listing 4. Sensor message format.

After receiving the messages containing the readings of all sensors, the WSN gateway at the platform side reassembles the XML document as specified by the platform. This process requires the pose of the robot (present in the control message) and the positions of the sensors in the local reference frame (dependent of the robot type). In order to work with different robot types, the WSN gateway can rely on a configuration file that supply the sensor positions for the robot type (types can be identified by the robot's node ID or the gateway can query the robot type through the Info service).

4.3. An Enhanced Range Service

An important feature of WSNs is the ability to acquire different environmental informations (e.g., temperature, humidity, light). These sensory functions were incorporated by the platform's Range Service. With this enhancement the robot is now capable of collecting environmental information from nodes in its communication radius.

The WSN sensor reading requests follow the range service pattern. Therefore, when a WSN sensor reading request arrives at the WSN gateway at the robot side, it generates a regular WSN reading request that the nodes can interpret and reply. This enhancement remains transparent to the client applications and can be employed even if the robot is connected through both WiFi and WSN networks.

The WSN sensor readings can be targeted to only one sensor or to a region nearby the robot. If the client wants to read a specific sensor the request must specify the node's ID. When the reading request is for a region, the robot's gateway will broadcast the reading request and wait for replies from its neighbors until a timeout occurs. When this timeout is reached a reply is sent to the client with the readings from the nodes that responded the broadcast. Figure 4 shows the system during a region reading and Figure 5 shows a sequence diagram for the entire process.

5. Results

This section presents the tests employed to evaluate the implementation. All the sensor network implementations were written in NesC 1.3.1 programming language for the TinyOS 2.1.1 operating system.

5.1. Simulations

Simulations were conducted using the TOSSIM, the TinyOS simulator, to evaluate some scalability aspects. The goal was to evaluate the impact that the network size would have on the robot-server comunication. A simple set operation was used to make this evaluation. The simulation scenario started when the message arrives on the server gateway, the operation request is processed and sent through the network to the robot. and ended when

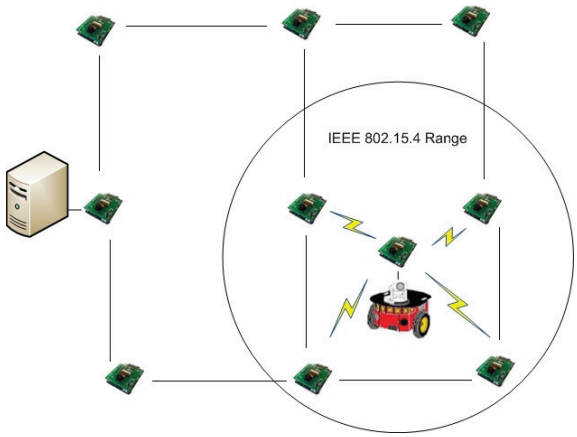

Figure 4. Robot performing a region sensors reading.

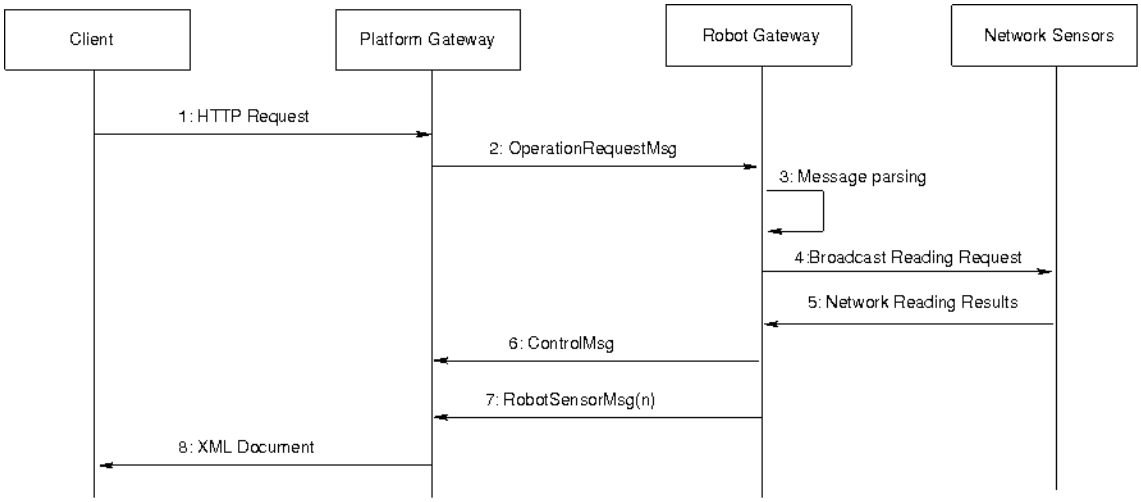

Figure 5. Sequence Diagram for a sensor region reading.

the response packet is delivered by the server gateway. To evaluate the scalability of the networking solution, simulations were conducted with networks with 9, 50, 100 and 150 nodes (Figure 6).

In all simulations, the robot was deployed as far from the server as possible, so that the packets would have be routed through the most number of motes. The operation was conducted sixty times for each configuration. Times were measured from the assemble of the message to the confirmation that the operation was successfull. The network was setup as a grid. Average RTT (Round Trip Time) for all the simulated scenarios are presented in Table 2.

Network Size	Average Number of Hops	Average RTT (ms)
9 nodes (3x3)	4 hops	79
50 nodes (5x10)	14 hops	220
100 nodes (5x20)	22 hops	389
150 nodes (5x30)	33 hops	517

Table 2. Simulation Average RTT.

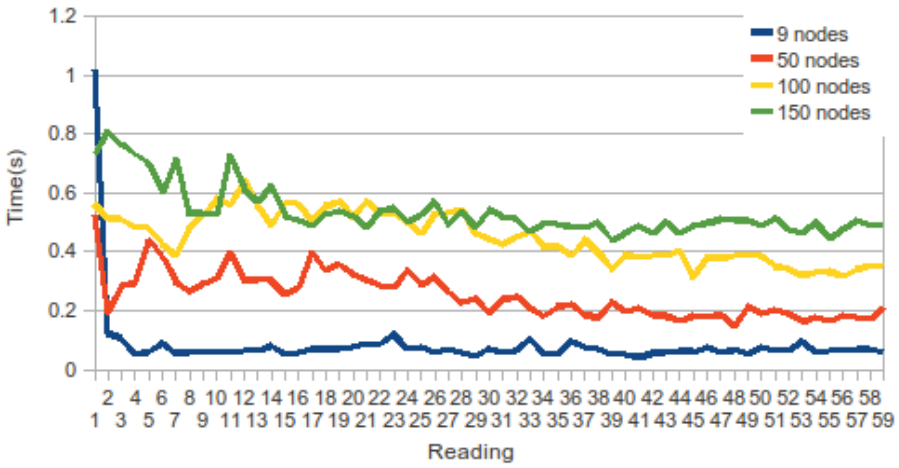

Figure 6. Simulations results for 9, 50, 100 and 150 nodes.

The simulations shown an expected and regular increase on RTT times for the different network sizes. Considering that the range of one node is approximately 25 meters, with 150 nodes a robot can cover a region up to $94000m^2$ with an average response time of 517ms. With these results we conclude that the system is scalable for sub-second latencies, and can be applied even to large coverage areas.

An interesting observation about the simulation results is that for large configurations the RTT tend to decrease over time. This is an indication that the network is constantly adapting its routing tables trying to establish better packet routes.

5.2. Experimental Setup

Experiments were conducted using the Mensic's iMote2 sensor platform with a low-power PXA271 Xscale processor at 416MHz with 265kB SRAM, 32MB Flash and 32MB SDRAM. The iMote2 is also equipped with an integrated CC2420 IEEE 802.15.4 radio transceiver from Texas Instruments. The radio supports a 250kb/s data rate with 16 channels in the 2.4 GHz band.

To communicate the iMote2 with the platform server and the robot the IIB2400 Interface Board was used. To perform the sensory functions the iMote2s were fitted with the ITS400 Basic Sensor Board. This sensor board is equipped with a 3-axis accelerometer, analog and digital temperature, humidity, and light sensors. For power supply it was used the IBB2400 Battery Board, which holds three AAA batteries each. Figure 7 illustrates these devices.

One requirement for the WSN was that it should not be device specific. This means that any TinyOS-based device should be able to integrate the network. To fulfill this requirement, we rely only on native TinyOS communication and routing protocols. The Collection Tree Protocol (CTP) [Gnawali et al. 2009] is a multi-hop tree-based protocol developed for WSNs. In the CTP algorithm routes are traced based on Expected Transmissions (ETX), an estimate of the number of transmissions it takes for a node to deliver a packet. Each node maintains and exchanges information only about neighbors

and the tree root on the network. This means that only the root route is maintained by every node on the network. In our WSN architecture two route trees are formed, one rooted on the node attached to server and one on the node attached to the robot. This means that independent routes are maintained on the network making the network more dynamic and fault tolerant.

A Pioneer P3-DX mobile robot was used in our experiments. The robot has a Pentium III processor with 512 MB of RAM and runs the Xubuntu 8.04 LTS operating system. The platform's Proxy Server is connected via USB to the WSN through an interface board. The robot is connected to the WSN also through an interface board packed with a sensor board (as such, the robot is a WSN mobile node). The rest of the network consists of sensor nodes fed by battery boards. Figure 8 shows the mobile robot fitted on it's top with a Mensic's iMote2 interface and sensor boards.

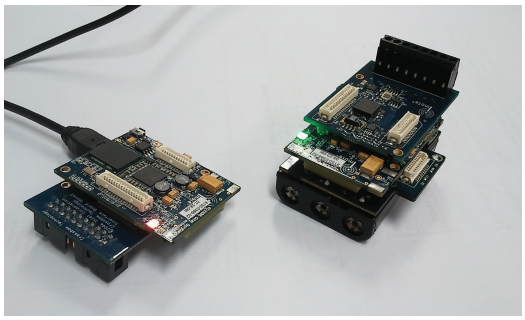

Figure 7. Interface Board (left) and a wireless sensor node (right).

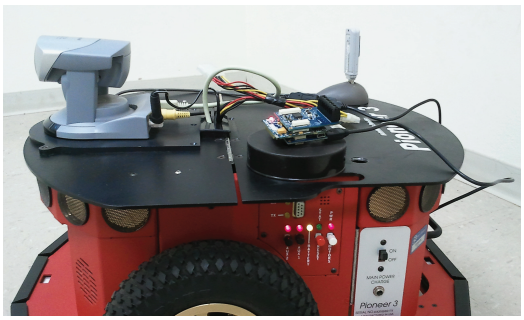

Figure 8. Mobile Robot fitted with iMote2.

The first evaluation was conducted in a one hop scenario. This means that the robot and the gateway node can communicate directly. The iMote2 has an approximated range of 25 meters. This scenario can be seen as if the robot and the platform server are in the same room. A second evaluation was conducted with a multi-hop scenario with 3 hops. The robot is deployed far from the server in an environment with multiple nodes. Figure 9 presents the end-to-end communication latencies for the single-hop and multi-hop WSN scenarios compared with Wi-Fi. The average RTT for each scenario is presented in Table 3. It is important to point out that the simulations results show only the network overhead while the experimental results take into account not only the network but also the platform overhead due to HTTP proxying and XML parsing.

Scenario	Average RTT (ms)
WiFi	19 ± 27
Single Hop	75 ± 31.5
Multi Hop	108 ± 11.4

Table 3. Average RTT, plus standard deviation, for WiFi and WSN.

As expected, the WSN performance is lower when compared with a WiFi network. In a single-hop structure the average time to perform a complete operation was 75ms. When using multiple hops this average time rose up to approximately 110ms. Using WiFi it took approximately 19ms to perform the same operation. Since we are dealing with

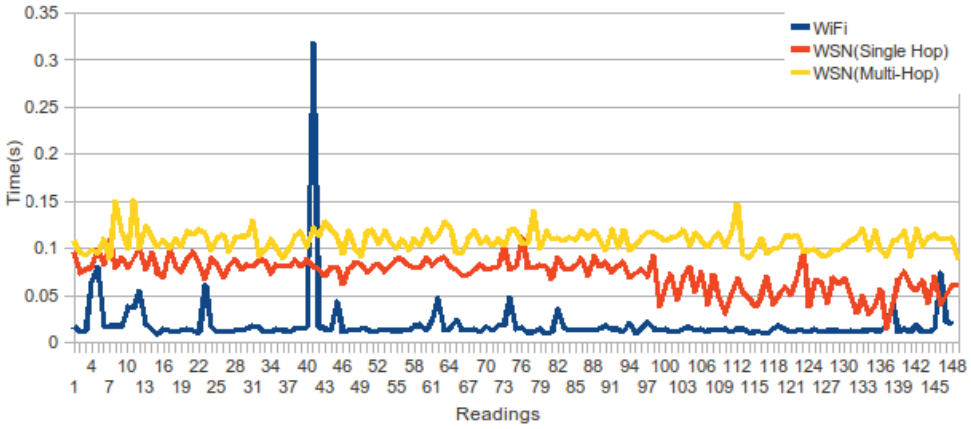

Figure 9. End-to-end communication latencies for WSN and WiFi networks.

small, limited devices these results show that the solution can be applied to teleoperation and navigation at low speed applications.

5.3. Case Study

A network robotics application was implemented for testing the proposed WSN architecture in a real scenario. The Potential Fields is an autonomous navigation algorithm that, given a goal position, is able to navigate the robot safely through the environment [Barraquand et al 1992]. The algorithm uses a combination of repulsive fields generated by the detected obstacles and an attractive field towards the goal. The resulting field is used to update the robots direction and speed.

The algorithm was executed on a client computer using both infrastructures in order to evaluate the performance of the WSN network. The proposed scenario was that the robot should reach a goal position across a room and come back to the starting point. Between the robot and the goal position there was an obstacle that needed to be avoided.

Figures 10 and 11 presents the experimental results from the real experiments.The path followed by the robot is presented in Figure 10. The path using the regular WiFi network is shown in Figure 11. In both executions the obstacle was the same, a square box located in front of the robot. The black dots represent the robot's sonar readings, the diference in the figures is due to the sonar sensor uncertainty and the robot's odometry error.

Although the algorithm has executed correctly and the robot did reach its goals in both networks, the paths taken by the robot were different. When using the Wi-Fi network the navigation application can interact faster and more times with the robot then when using the WSN, due to different network technologies latencies. This is illustrated in areas 1 and 2 of Figures 10 and 11. Beacause of the WSN higher latency when the robot is getting closer to the obstacle the algorithm takes longer to perceive and avoid it. It also takes longer for the algorithm to notice that there are no more obstacles in the direction of the goal. Despite the lower performance, the WSN allowed the correct conduction of the robot, being an interesting option for outdoor mobile robotics.

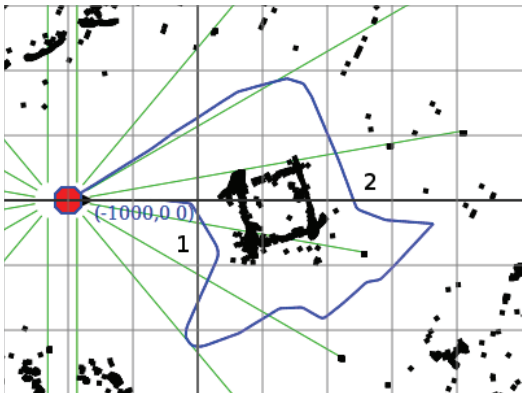

Figure 10. Robot path using a wireless sensor network.

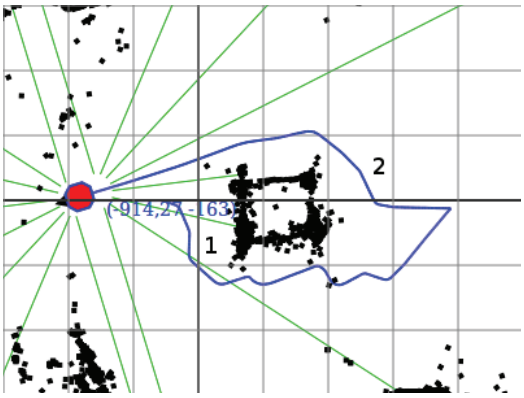

Figure 11. Robot path using a WiFi network.

6. Conclusion

This paper presented a WSN architecture that supports access and control of mobile robots on environments were traditional networks such as WiFi are not available. In addition, this architecture enhances the robot's sensory capabilities by allowing mobile robotics applications to access environmental data such as temperature and humidity.

Both simulations and real world experiments were conducted in order to evaluate the performance of the proposed architecture. Through simulations it was possible to conclude that the solution is scalable and can be applied to WSN covering wide areas. The experiments with the mobile robot compared the WSN and the WiFi performances. Using the Potential Fields algorithm the solution was tested in a real robotic application and it was demonstrated that a WSN is a good choice when WiFi networks are not available in the field.

Future works include the porting of the WSN system from TinyOS to embedded Linux OS, implementing a discovery algorithm to identify robots entering and leaving the network area, and port the microserver architecture to the embedded Linux OS, therefore eliminating the robotic framework overhead. We are also investigating WSN routing algorithms more suitable for dynamic topologies as the case where a set of WSN nodes are mobile nodes.

References

Akyildiz, I., Su, W., Sankarasubraniam, Y., and Cayirci, E. (2002), "A survey on sensor networks", *IEEE Communications Magazine*, Vol. 40, No. 8, August.

Barraquand, J., Langlois, B. and Latombe, J. (1992), " Numerical Potential Field Techniques for Robot Path Planning," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 22, No. 2, March.

Cardozo, E., Guimarães, E., Rocha, L., Souza, R., Paolieri, F. and Pinho, F. (2010) "A Platform for Networked Robotics", IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, October.

Fu, S., Yang, G. and Hou, Z. (2009), "An Indoor Navigation System for Autonomous Mobile Robot using Wireless Sensor Network", IEEE International Conference on Networking, Sensing and Control, Okayama, Japan.

Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E. and Culler, D. (2003), "The nesC Language: A Holistic Approach to Networked Embedded Systems", ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation, San Diego, USA.

Gnawali, O., Fonseca, R., Jamieson, K., Moss, D. and Levis, P. (2009), Collection Tree Protocol, 7th ACM Conference on Embedded Networked Sensor Systems, Berkeley, USA.

Meger, D., Marinakis, D., Refleitis, I. and Dudek, G. (2009), "Inferring a Probability Distribution Function for the Pose of a Sensor Network using a Mobile Robot", IEEE International Conference on Robotics and Automation, Kobe, Japan.

Menegatti, E., Zanella, A., Zilli, S., Zorzi, F. and Pagello, E. (2009), "Range-Only SLAM with a Mobile Robot and a Wireless Sensor Networks", IEEE International Conference on Robotics and Automation, Kobe, Japan.

Mobile Robotics Inc. (2010), Advanced Robot Interface for Applications (ARIA), `http://robots.mobilerobots.com/wiki/ARIA`.

Peng, E., Meng, M. and Liang, H. (2009), "An Experimental System of Mobile Robot's Self-Location Based on WSN", IEEE International Conference on Automation and Logistics, Shenyang, China.

Player Project Web Site (2010), `http://playerstage.sourceforge.net/`, 2010.

Popa, M., Marcu, M. and Popa, A. (2009), "Wireless sensory control for mobile robot navigation", 7th International Symposium on Intelligent Systems and Informatics, Tokyo, Japan.

Richardson L. and Ruby, S. (2007), "RESTful Web Services", O'Reilly.

Sichitiu M. and Ramadurai, V. (2004), "Localization of Wireless Sensor Networks with a Mobile Beacon", IEEE International Conference on Mobile Ad-hoc and Sensor Systems, Fort Lauderdale, USA.

Tekdas, O., Isler, V., Lim, J. and Terzis, A. (2009), "Using Mobile Robots To Harvest Data From Sensor Fields", *IEEE Wireless Communications*, Vol 16, No. 1, February.

TinyOS Project Web Site (2010), `http://www.tinyos.net/`.

Yao, Z. and Gupta, K. (2010), "Distributed Roadmaps for Robot Navigation in Sensor Networks", IEEE International Conference on Robotics and Automation, Alaska, USA.