A Software Defined Radio Gateway for Wireless Sensor Networks

Roberto de Matos*, Tiago Rogério Mück†, Antônio Augusto Fröhlich† and Leandro Buss Becker*
Federal University of Santa Catarina (UFSC)
Florianópolis - Brazil
*Department of Automation and Systems (DAS)
Email: {rmatos,lbecker}@das.ufsc.br
†Laboratory for Software and Hardware Integration (LISHA)
Email: {tiago,guto}@lisha.ufsc.br

***Abstract*—The fast proliferation of Wireless Sensors Networks has triggered new fields of research related to communication reliability, implying the need to ensure connectivity among heterogeneous WSNs and the outside world. Thereby it is possible to enhance the availability of the data being transmitted in these networks. Due to the very limited power and computational resources of WSN nodes, communication with other networks is typically made through external gateways. This paper proposes the use of Software Defined Radio (SDR) to allow changes in the physical layer of the gateways in order to increase the flexibility in WSN communications. To exemplify our proposal we developed a gateway for 802.15.4 and 802.11b wireless networks. This example is used to evaluate critical parameters in such scenario, such as the impact of latency caused by the physical layer reconfiguration.**

I. Introduction

The popularization of Wireless Sensor Networks (WSN) has allowed the monitoring of many environments, such as home security, industrial fault diagnosis, chemical and biological detection or medical and environmental monitoring. The proliferation of these networks has triggered new field of research related to communication reliability, implying the need to ensure connectivity among heterogeneous WSNs and the outside world. Thereby it is possible to enhance the availability of the data being transmitted in these networks, as in many application scenarios this data should be made available in the outside world.

The very limited power and computational resources of WSN nodes require efficient operation and power management, which are not available in traditional communication radios and high overhead protocols, like IEEE 802.11 and TCP/IP [1], [2]. Thus, proprietary protocols and low-power radios are used in WSNs, such as 802.15.4 and proprietary solutions, i.e. CC1000 (433MHz or 900MHz), which integrates Berkley's Mote Mica2. Therefore, it is necessary to use of gateways to interconnect WSNs themselves or connect WSNs and Area Networks (WAN, LAN or Cell phone networks) [3], [1].

Research in WSN gateways has focused on aid and to allow flexibility to project with a bigger number of protocols. [1] provides a modular application layer WSN gateway framework that allows the gateway to be deployed in heterogeneous network environments and enables remote upgrading. [4] presents a communication architecture that adapts to a wide range of underlying communication mechanisms, from the MAC layer to the transport layer, without requiring any changes to applications or protocols. [3] presents the architecture of the sensor gateway for web-based management.

All these schemes allow a combination of some protocols requiring, or not, changes to the applications. However, the flexibility of these architectures can be used only on software layers of the communication protocols, so they are limited by the physical layers defined at project time. This paper proposes the use of Software Defined Radio (SDR) to allow changes in the physical layer of the gateways in order to increase the flexibility in WSN communications. The SDR can provide significant benefits as gateways or base stations for sensor networks [5]. To exemplify our proposal we developed a gateway for 802.15.4 and 802.11b wireless networks. This example is used to evaluate critical parameters in such scenario, such as the impact of latency caused by the physical layer reconfiguration.

The rest of this paper is organized as follows: Section II presents an Overview of SDR, which covers the GNU Radio toolkit and the Universal Software Radio Peripheral (USRP). Section III presents our proposal of SDR-gateway for Wireless Sensor Networks. Section IV presents an evaluation of the proposed gateway. Concluding remarks and future work are given in Section V.

II. Software Defined Radio

The Software Defined Radio is a technique for building wireless communication systems [6]. The goal is to get the software as close to the antenna and use it to filter, modulate, demodulate and other stages of transmition and reception paths. The main advantage is the flexibility of the physical layer, which allows communication with other radios in different frequencies or modulation only by changing the software. The ideal SDR eliminates almost all hardware, only an ADC takes samples from the antenna to the software. However, there aren't ADCs fast enough to sample all the bandwidth used. Thus, more hardware is needed to down-convert the chosen band for the ADCs to sample it.

Despite the software technology employed in radios guarantees a much quicker development cycle and in-field functions upgrades with complete modification, the extreme flexibility and on-the-fly reconfiguration of SDR spend more energy relative to fixed function ASICs, thus if the physical layer flexibility is not important, this cost is unnecessary. Nowadays, the limiting factors of the physical layer flexibility in high performance systems are the requirement of FPGA or ASIC to support the high data rate and A/D performance.

Many architectures and frameworks of SDR are available. One of the most used is the GNU Radio, and it was the choice to develop this work, because it's free, and there is a high active community effort; it works with many hardware platforms, including the USRP (Universal Software Radio Peripheral). The USRP is a low-cost and flexible platform for software defined radios. In fact, there are better SDR platforms than GNU Radio and USRP2 for commercial embedded system applications. However, the most available platform and free code come from the combination of GNU Radio project and USRP hardware, which allow a rapid development and experimentation with SDR. The next two subsections give an overview about GNU Radio and USRP, respectively.

A. GNU Radio Overview

GNU Radio is a free software toolkit for the development of Software Defined Radios [6]. It was started with the funding of John Gilmore and it has been developing since 1998, with a complete reformulation in 2004. The simple use and high performance for Digital Signal Processing (DSP) are achieved by the hybrid nature of the framework, which uses C++ to write the core of the signal processing components (Processing Blocks) and Python to connect these components in a data flow graph to make filters, modulators, demodulators and other structures that compose the radios.

Processing Blocks are the components that act in the data stream and are divided into three types: Normal, Sources, and Sinks. Most of them are Normal, which have input, output and are responsible by signal processing. The Sources have only output and they provide the data stream, always starting the flow graph. While the Sinks have only input and finish the flow graph, the role of these blocks is consume the processed stream.

All the Processing Blocks have the I/O signature, work method and forecast method. The first defines the minimum and the maximum numbers of input and/or outputs and the size of data types. The work method acts on the input and produces the output. It is the core of the digital signal processing algorithm. The forecast method helps the scheduler decide when the work method should be called. It is used to estimate the input requirements of a block to produce a output. The two forecast method parameters are; the number of output items to produce each output stream and an integer vector that saves the number of input items required on each input stream [7].

The GNU Radio provides two main structures: the Flow Graph Mechanism (FGP) and the Scheduler. The first is responsible for the data flow abstraction, in other words, the sequence of signal processing blocks and connections between them. Figure 1 shows a typical GNU Radio flow graph. The *connect method* of GNU Radio specifies how the output streams of a block connects to the inputs of other blocks, and it is used to construct the flow graph, which will be automatically created to process the data stream. The process details to create the flow graph are hidden from the user, where the main function of FGP is the buffer allocation algorithm, which defines the size of the buffer between the input and output. This buffer allocation function considers the input and output stream sizes and the relative rate of consumption and production.

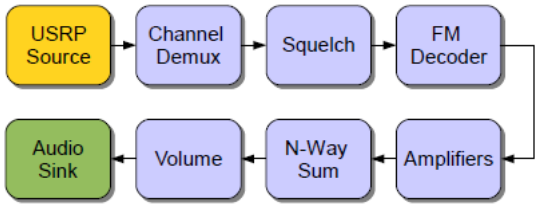

Fig. 1. A typical GNU Radio flow graph of Narrowband FM receiver

The Scheduler is implemented as a single thread and executes the flow graph created by the FGP. Each block is executed sequentially and its input requirements and input buffers are analyzed to determine if the work function should be called. If there isn't sufficient input, the Scheduler skips to the next block.

B. USRP Overview

The Universal Software Radio Peripheral (USRP) is a custom-made, low-cost and flexible platform developed by Matt Ettus [8] for GNU Radio project. It is basically composed of ADCs, DACs, an FPGA, slots for daughterboards (RF front ends) and a communication interface, which connect the GNU Radio framework with the RF world. The daughterboards have the function of down-converting from received carrier frequency to intermediate frequency and the inverse for the transmitted signal.

There are two versions of USRP board, which are compatible with all daughterboards, Table I [6] shows the differences. The main problem of the USRP1 is the interface to connect with PC, the USB2 standard has a maximum rate of 60 MB/s, but the real rate is lower and USRP1 is limited to transfer at most 32MB/s, thus the USB2 can support only 8MS/s (complex signal - 2 bytes I and 2 bytes Q channels), which achieve a sampling window of 8MHz. While the ADC has 64MS/s and the DAC has 128MS/s, these sample rates can achieve an effective bandwidth of around 32 MHz for receiver and allow generating signals of around 60 MHz for transmitter. For this reason, in the FPGA, Digital Down/UP Converters (DDC) are implemented to decimate the samples and reduce the data rate to send over the USB2. The decimation rate can be controlled by the GNU Radio. Figure 2 shows the internal structure of DDC and figure 3 shows a high level view of RX path implementation in the FPGA.

TABLE I
USRP VERSIONS DIFFERENCES

	USRP1	**USRP2**
Interface	USB 2.0	Gigabit Ethernet
FPGA	Altera EP1C12	Xilinx Spartan3 2K
RF Bandwidth to/from host	8MHz @ 16bits	25MHz @ 16bits
Cost	700	1400
ADC Samples	12-bit,64 MS/s	14-bit,100 MS/s
DAC Samples Daughterboard	14-bit, 128 MS/s	16-bit, 400 MS/s
capacity	2 TX, 2 RX	1 TX, 1 RX
SRAM	None	1 Megabyte
Power	6V, 3A	6V, 3A

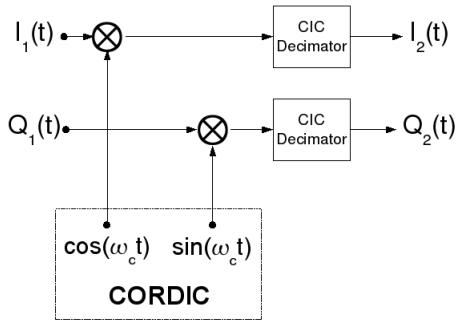

Fig. 2. DDC implemented in USRP's FPGA

The USRP2 was improved in this aspect by replacing the USB by a Gigabit Ethernet interface, with a maximum transfer rate of 125 MB/s, allowing a maximum sampling window up to 25 MHz. Other improvements are the increase of ADC and DAC sample rates and a bigger and faster FPGA. On other hand, the daughterboard capacity fell by half and the price increased by two.

Due to the bigger USRP2 FPGA, more functions can be implemented in hardware, that joins the reconfiguration flexibility and the performance required for some applications. Besides the size of the FPGA, another important change occurred in the logic of control. The USRP1 spreads the logic control between the FPGA and the USB controller, such as adjustment of daughterboards cannot be controlled by the FPGA, the USB controller uses the I2C bus, which has no connection to the FPGA. On the other hand, the USRP2 centralizes all control

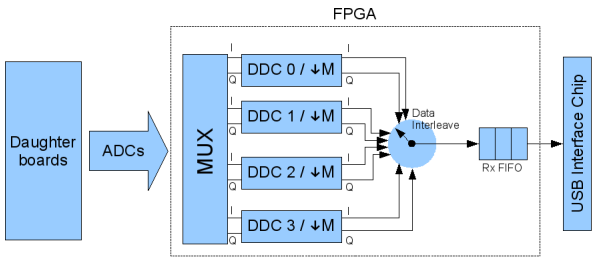

Fig. 3. Overview of RX path

functions with the aeMB softcore processor in the FPGA, except its owner configuration, which is done by a CPLD (Xilinx XC9572).

The second version doesn't turn the first completely obsolete for Gnu Radio applications. Meanwhile, the USRP2 is more flexible, powerful and should be used more for deep researches in the future.

III. SDR AS GATEWAY ON WSN

Due to rapid growth in the number of nodes of WSNs, the low capacity of processing and storage, it is required the dissemination of large amount of data generated by these networks. One of the most direct ways of doing it is to use a gateway to translate or adapt the information between WSNs and Area Network, such as WAN, LAN or Cellphone Network. Normally there should be available a distinct network interface for each of the used physical layers. However, we can use a SDR to allow more flexibility for Gateway role, since physical layer can be modified to interact with many kinds of networks and to create different topologies.

Thus, some nodes of a WSN can be replaced by SDRs with a careful analysis about the necessary flexibility of the network and the high cost for the additional hardware resources and power consumption. With the SDR technology, the same equipment can be different behaviors within the WSN. Figure 4 shows three possible scenarios: Gateway, Bridge and SDR Node. Any of the behaviors can be changed to another dynamically only with software modification.

The Gateway is the more common behavior, which intermediates communication between a WSN and another Area network [1], [4], [9]. As a Gateway, the SDR has a interface configured with WSN technology and another with Area network standard. The messages exchanged between the two networks should be translated according to the protocols of the upper layers.

The Bridge behavior ensures the convergence between different WSNs [5], normally it is useful when an old structure is installed and new-technology nodes are introduced in the network or when neighboring and incompatible networks could benefit themselves by increasing the coverage through the intercommunication.

Another configuration yet unexplored is here named SDR Node, which acts like a common network node and can be reconfigured to supply the connectivity in case of faults. If a WSN cluster loses communication with other clusters, the SDR node can try other settings to reconnect its cluster. For example, in Figure 4 if a area problem causes the fail of nodes 7, 8, 9 and SN2, the nodes SN1 and 1 up to 6 will be isolated. In this case, with a coordinated action, the SDR nodes (SN1 and SN3) can reconfigure themselves with a long-range PHY layer to keep connection between the two new clusters and to ensure that data of isolated nodes will be arrived at gateway (SG1). Figure 5 shows the reconfigured scenery.

Even with a modular [1] or adaptive [4] structure, all these behaviors can only be achieved dynamically by using a flexible physical layer. Our research analyzes the time latency impact

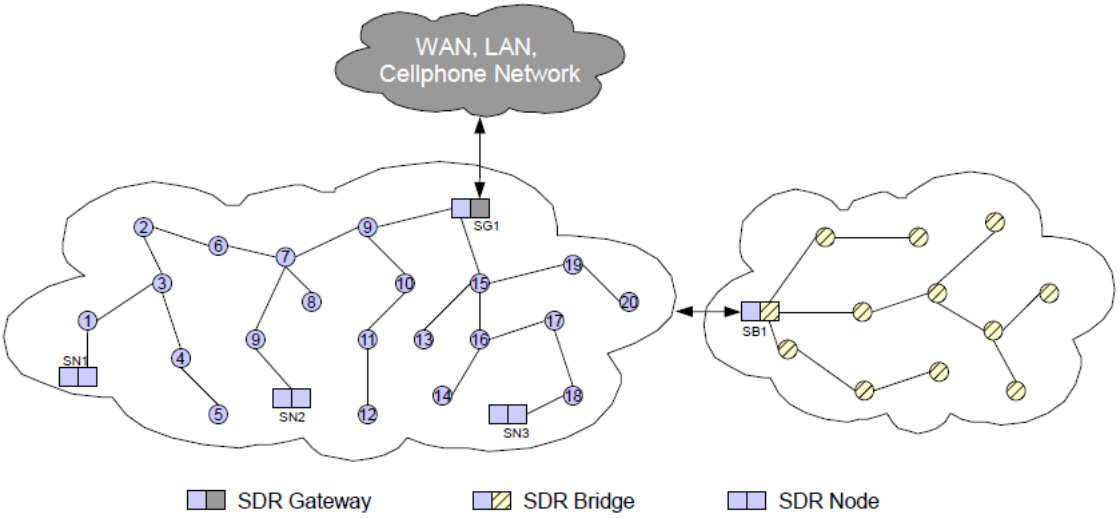

Fig. 4. Three possible scenarios using SDR on WSN

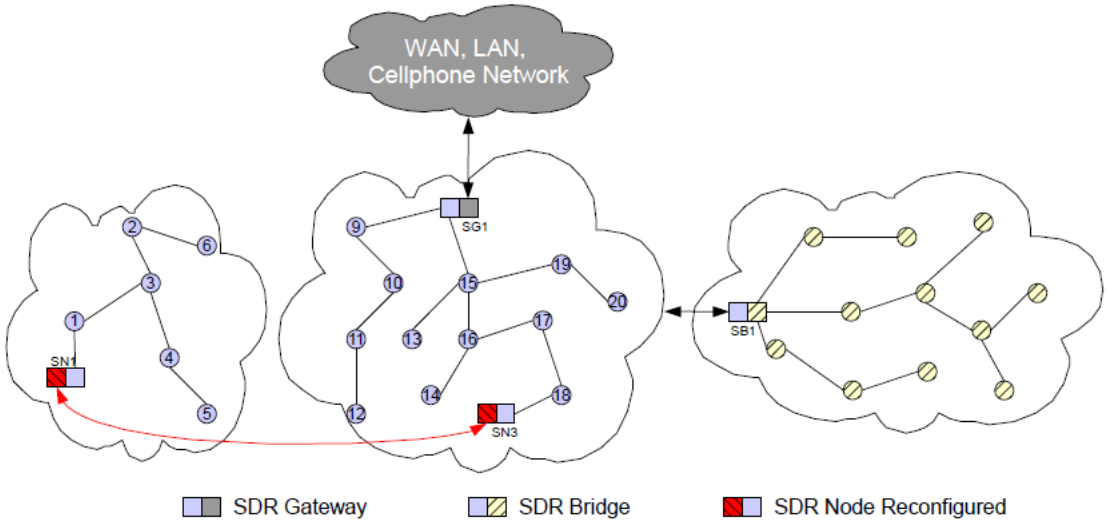

Fig. 5. SDR Node reconfigured to ensure connectivity

of the reconfiguration between two physical layers extensively used by WSN and gateways projects, which are 802.15.4 [5] and 802.11b. For evaluate a full physical layer reconfiguration, we mounted the setup of Figure 6, where was used a USRP2 and RFX2400 daughter board, which has frequency range from 2.3 to 2.9 GHz. MicaZ was used to generate the 802.15.4 packets and another USRP2 to generate the 802.11b packets.

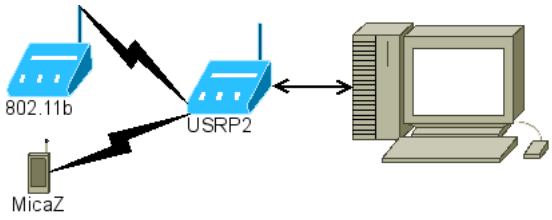

Fig. 6. Evaluation setup

Despite the quick and flexible development, the GNU Radio data flow model imposes some limitations for implementing simultaneous radio chains. Both paths share the same USRP2 block, thus they must be on the same flow graph. But the GNU Radio model requires both paths to be running at the same time, and stopping one of the radio chains blocks the entire flow graph. We were able to overcome this by implementing a demux using stall blocks. These stall blocks can simply pass the inputs directly to the outputs or they can consume all the inputs and generate no outputs, thus we can easily switch between 802.11b and 802.15.4 block chain without blocking the entire flow graph. The figure 7 shows the implemented structure in GNU Radio and the *demux* details.

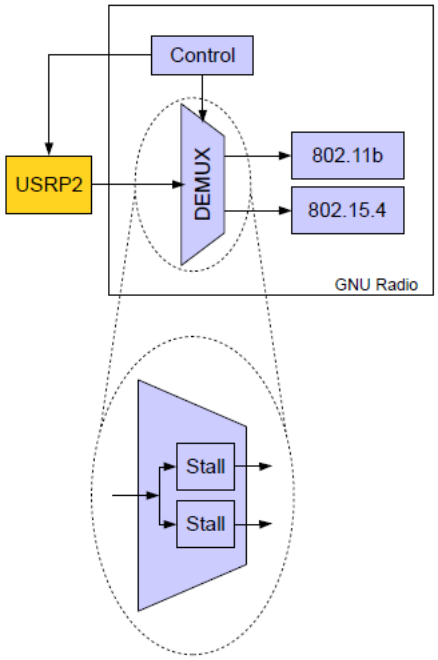

Fig. 7. Structure implemented in GNU Radio and Demux detail

IV. Evaluation of the Proposed Approach

To achieve the equipment flexibility proposed in Section III is necessary the evaluation of critical parameters, such as the impact of latency caused by the physical layer reconfiguration. Table II shows the results of four kinds of time latency for physical layers reconfiguration, which are described below:

- T_{init} is the time to create the flow graph and make all connections, which only occurs at start-up of system.
- T_{conf} is the time to configure the USRP2 for the new physical layer, where are set a new center frequency and decimation rate, which are the hardware parameters that vary between 802.11b and 802.15.4
- T_{lock} is the time to stop the flow graph responsible for implementing the physical layer.
- T_{unlock} is the time to release the flow graph again.

TABLE II
Time of Initialization and Configuration

	802.15.4	**802.11b**
T_{init}	98.72 ms	4.2 ms
T_{conf}	1.04 ms	0.94 ms
T_{lock}	12.9 µs	19.1 µs
T_{unlock}	11.9 µs	25.1 µs

Thus the times to exchange between physical layers are given by the following formulas:

$$T_{PHY1 \rightarrow PHY2} = T_{lock\_PHY1} + T_{conf\_PHY2} + T_{unlock\_PHY2}$$

The final values of table III were obtained as described above. The time to exchange between PHY layer 802.15.4 by the 802.11b is $0.98\,\mathrm{ms}$ and the inverse spends $1.07\,\mathrm{ms}$. The really relevant latency is the T_{conf}, which makes sense, due the configuration of USRP2 hardware depends on the drivers and the communication channel, it should bear more interferences of intrinsic system latencies.

TABLE III
Time to Exchange the Physical Layer

802.15.4 → 802.11	**802.11 → 802.15.4**
0.98 ms	1.07 ms

Despite the exchange time is not being large and not impeding the SDR gateway, a SDR running in a dedicated system has many advantages in this context. Without the interferences of a generic processing environment, like a Personal Computer (PC) used by the GNU Radio, the exchange of PHY layer should be faster. In fact, there are better SDR platforms than GNU Radio and USRP2 for commercial embedded system applications, such as Bell Labs Programmable Radio Platform [10] or Embedded SDR [11]. However, even these embedded platforms consume more resources than simple nodes to give more connectivity and flexibility, thus the research to determine the gateway numbers and their locations in the network can be used to balance this trade-off [12].

The magnitude of latency to exchange physical layers allows the creation of a multi-PHY architecture for providing services to upper layers. This flexibility can be used directly by application or by Data Link and Network layers to meet the communications requirements set by the user. Moreover, this architecture can be useful to provide vertical mobility within the various networks available in order to equalize the best data transfer rate, range and power savings.

V. Conclusions and Future Work

In this paper, we suggest the use of SDR nodes to have more versatility in WSN gateways, which can assume different behaviors to increase flexibility and connectivity of WSNs. Three scenarios were shown and discussed, but there are other applications in WSN that can use the versatility of SDR.

Furthermore, we used a Software Defined Radio technology to present the analyses of time latency of dynamic exchange between 802.15.4 PHY and 802.11b PHY, which can be extrapolated for any other physical layers. Our experiment showed that even for a entire physical reconfiguration the magnitude of latency is not prohibitive.

As future work, we want to improve the PHY layer exchange structure to allow the integration with novels WSN architectures and upper layers. Another interesting field of research is the search of low-power SDR to employ in real WSN applications. Moreover the algorithms to determine connectivity problems and to coordinate the PHY layers modifications are great research challenges.

REFERENCES

[1] L. Wu, J. Riihijarvi, and P. Mahonen, "A modular wireless sensor network gateway design," Aug. 2007, pp. 882–886.

[2] X. Luo, K. Zheng, Y. Pan, and Z. Wu, "A tcp/ip implementation for wireless sensor networks," vol. 7, Oct. 2004, pp. 6081–6086 vol.7.

[3] K. il Hwang, J. In, N. Park, and D. seop Eom, "A design and implementation of wireless sensor gateway for efficient querying and managing through world wide web," *Consumer Electronics, IEEE Transactions on*, vol. 49, no. 4, pp. 1090–1097, Nov. 2003.

[4] A. Dunkels, F. Österlind, and Z. He, "An adaptive communication architecture for wireless sensor networks," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2007, pp. 335–349.

[5] T. Schmid, T. Dreier, and M. B. Srivastava, "Software Radio Implementation of Shortrange Wireless Standards for Sensor Networking," in *Conference On Embedded Networked Sensor Systems*, 2006.

[6] E. Blossom, "Gnu radio," http://www.gnu.org/software/gnuradio, 2009.

[7] D. Shen, "Writing a signal processing block for gnu radio," https://radioware.nd.edu/documentation/advanced-gnuradio/writing-a-signal-processing-block-for-gnu-radio-part-i, Tech. Rep., Jun 2005.

[8] M. Ettus, "Universal software radio peripheral," http://www.ettus.com/, 2009.

[9] P. Song, C. Chen, K. Li, and L. Sui, "The design and realization of embedded gateway based on wsn," vol. 4, Dec. 2008, pp. 32–36.

[10] B. Ackland, D. Raychaudhuri, M. Bushnell, C. Rose, and I. Seskar, "High performance cognitive radio platform with integrated physical and network layer capabilities," http://www.winlab.rutgers.edu/pub/docs/NeTS-ProWiN1.pdf, Tech. Rep., Jul 2005.

[11] P. Balister, T. Tsou, and J. H. Reed, "Software defined radio on small form factor systems," Mar 2007.

[12] W. Youssef and M. Younis, "Intelligent gateways placement for reduced data latency in wireless sensor networks," June 2007, pp. 3805–3810.