Feasibility Analysis of Controller Design for Adaptive Channel Hopping*

Branko Kerkez
Department of Civil and Environmental Engineering
University of California
Berkeley, CA 94720-1710
bkerkez@berkeley.edu

Thomas Watteyne
Berkeley Sensor and Actuator Center
University of California
Berkeley, CA 94720-1774
watteyne@eecs.berkeley.edu

Mario Magliocco
Department of Civil and Environmental Engineering
University of California
Berkeley, CA 94720-1710
mag@berkeley.edu

Steven Glaser
Department of Civil and Environmental Engineering
University of California
Berkeley, CA 94720-1710
glaser@ce.berkeley.edu

Kris Pister
Berkeley Sensor and Actuator Center
University of California
Berkeley, CA 94720-1774
pister@eecs.berkeley.edu

ABSTRACT

Communication reliability in Wireless Sensor Networks (WSNs) is challenged by narrow-band interference and persistent multi-channel fading. Frequency-agile communication protocols have been recently designed and standardized to increase reliability. These protocols, however, do not adapt the set of channels they hop on to the environment.

In this paper, we evaluate the efficiency of a controller which continuously samples all available frequency channels in order to operate on a channel which performs reasonably well. We show that the overall average link Packet Delivery Ratio when using this controller reaches 99.4%, and is higher compared to a single channel solution, on any channel.

We evaluate the efficiency of this approach by simulating its behavior on connectivity traces gathered during a real-world deployment. This data set is dense in time and sufficiently large in number of nodes and time to be statistically valid. We believe that the use of connectivity traces for performance evaluation will become commonplace as the number and variety of these traces increases.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

1. INTRODUCTION

Communication reliability in Wireless Sensor Networks (WSNs) is challenged by narrow-band interference and persistent multi-channel fading. As such, motes are equipped with the ability to transmit and receive data on different frequency channels. One approach is to select the channel on which the loss of transmitted packets is minimized; yet, the best channel may vary over time. The option advocated in this paper is to change channel on a link-by-link basis when necessary, a concept known as **adaptive channel hopping**.

In adaptive channel hopping, a mote transmits data on a channel which is predetermined to work acceptably well. Rather than persistently sending information on this stable channel, the pair of communicating motes transmit packets on different channels every now and then. These intermittent transmissions allow for the goodness of other channels to be estimated. When the current channel starts performing unacceptably bad, the mote pair switches to the channel which features the best goodness estimation.

As detailed in Section 2, channel hopping has received increased attention in industrial and standardization bodies, driven by the quest for reliability. To the best of our knowledge, however, no work has quantified the performance of adaptive channel hopping. Does adaptive channel hopping offer an increase in performance over a single-channel solution? Over a blind channel hopping approach? If yes, how often should a pair of nodes evaluate the goodness of the different channels? Is it feasible to design a controller for adaptive channel hopping which is simple enough to be implemented on current hardware?

Choosing the methodology for answering those questions is non-trivial. On one hand, simulated propagation models do not capture well complex phenomena such as persistent

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7934
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7934

multi-path fading, or the impact of a dynamic environment. On the other hand, a purely experimental study does not provide the ability to evaluate different algorithms under the exact same conditions, as connectivity is time-varying in nature. This paper proposes to *replay* communication algorithms on *connectivity traces* gathered from a real-world deployment. We believe that this novel approach elegantly copes with the issues raised above, and that it will become commonplace for the performance evaluation of complex environments such as channel hopping protocols.

The contributions of this paper can be summarized as follows:

- We show that replaying communication algorithms on real-world traces allows for quick and realistic performance evaluations;
- we demonstrate how adaptive channel hopping increases the average link Packet Delivery Ratio $\mathcal{P}$ compared a single channel solution operating on any channel;
- we evaluate the impact of the choice of the channel in a single channel solution, and the impact of the probing period when using adaptive channel hopping.

Evaluating the efficiency of communication algorithms by replaying their behavior on real-world connectivity traces is, we believe, an approach particularly suited for complex systems such as channel hopping Medium Access Control (MAC) protocols.

The remainder of this paper is organized as follows. Section 2 present a comprehensive overview of the latest academic and standardization efforts in frequency-agile communication, as well as an overview of the connectivity traces used in this paper. Section 3 describes the simple controller for adaptive channel hopping evaluated in this paper, its complexity and limitations. In Section 4, we present the results obtained by simulating the controller's behavior on real-world connectivity traces; comparison is offered with single channel solutions. Section 5 underlines the research directions stemming from this paper and discusses the choice of parameters. Finally, Section 6 concludes the paper and presents directions for future research.

2. RELATED WORK

2.1 Frequency-Agile Communication

Hardware Support. Frequency-agile communication requires nodes to change the frequency channel they transmit on or listen to regularly. Luckily, radio chips have become very efficient at doing this. As an example, all IEEE802.15.4-compliant [1] radio chips – the *de facto* standard for WSN hardware – switch channels in less than $192\mu s$. Such chips live in most popular platforms, such as TelosB, MICAz, IRIS, SUNspot, IMote or EPIC. Moreover, non-IEEE802.15.4 chips such as Texas Instruments' CC1100, CC1101 and CC2500 [2] feature low turn-around times of $88.4\mu s$. Combined with the fact that typical clocks drift by $10ppm$ or less, fast channel hopping capabilities have made frequency-agile communication efficient.

With nodes changing channels, senders and receivers need to be tightly synchronized for their radio to be on the same channel when communicating. When using one of the protocols presented below, nodes hop from channel to channel regularly (e.g. every few tens of ms), following a pre-determined hopping pattern. To the best of our knowledge, no protocol determines this hopping pattern dynamically. Protocols presented below allow for a relative de-synchronization of up to a few milliseconds. As a result, the vast majority of frequency-agile MAC protocols are Time Division Multiple Access (TDMA)-based: time is cut into slots and nodes maintain synchronization with their neighbors. We present a comprehensive overview of the latest frequency-agile MAC protocols and standardization efforts below.

Frequency-Agile MAC Protocols. Lightweight MAC [3] (**LMAC**) assigns slots to nodes in a distributed way. **Multichannel LMAC** [4] proposes, when all slots are assigned, to pick a slot on another – randomly chosen – frequency. The number of potential slots is roughly multiplied by the number of frequency channels, which allows more nodes to communicate than LMAC. Omnet++ simulations show that the use of multiple channels decreases the number of active nodes while reducing collisions.

Y-MAC [5] is primarily designed to decrease latency. Nodes are synchronized and reception slots are assigned to each node on a common base channel. In case multiple packets need to be sent between neighbor nodes, successive packets are sent on a different frequency. As a result, bursts of messages ripple across channels, which reduces latency. Y-MAC was implemented in the RETOS operating system on the TmoteSky motes and compared to LPL. With $8s$ resynchronization period and 5 frequency channels, the idle duty cycle when sending one packet every $10s$ is around 7%.

Time Synchronized Mesh Protocol (**TSMP**)[6] was designed to improve reliability. TSMP employs channel hopping: different links use different frequency channels and the same link hops during its lifetime across different channels. This reduces the impact of narrow-band interference and persistent multipath fading. [7] presents experimental results in which 44 nodes run TSMP for 28 days in a printing facility. The authors show how channel hopping, combined with a retransmission policy, yields an end-to-end delivery ratio of 99.999%. TSMP uses a central coordinator which retrieves the list of nodes, their neighbors and their traffic requirements. This allows it to construct a schedule which is then communicated back to the network. Note that, in this paper, we reuse the connectivity traces from the experiment presented in [7].

Critical applications require reliable solutions, and channel hopping is one answer to this need. With the success of proprietary solutions such as TSMP, standardization bodies have been working on similar solutions, as detailed in the next section.

Standardization Efforts. **IEEE802.15.1** [8] is the technology used by the Bluetooth consortium. The physical layer features 79 1-MHz channels in the 2.4GHz ISM band.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7934
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7934

Devices wishing to communicate group around a leader and synchronize to that leader's clock. Time is sliced up into $625\mu s$-long slots, and a hashing function translates the leader's address into a channel hopping pattern. All nodes follow that pattern blindly, changing channels roughly 1600 times per second.

The HART Communication Foundation standardizes embedded networking solutions for industrial applications. Their wireless extension, called **WirelessHART** [9], uses a central controller to schedule communication. WirelessHART uses IEEE802.15.4 radios to blindly hop on 15 frequency channels in the 2.4GHz band. Similarly to TSMP, reliability is increased by having each node maintain connectivity to at least two parent nodes in the routing graph, enabling the network to resist link failures.

Another industrial wireless standardization body is the ISA100 Wireless Compliance Institute. Their latest standard **ISA100a** [10] is similar in essence to TSMP or WirelessHART, yet features interesting channel hopping mechanisms. Successive channels in the hopping pattern are separated by at least 15 MHz (three IEEE802.15.4 channels). When retransmissions occur, they will not encounter or cause interference in the same IEEE802.11 – Wi-Fi – channel. The hopping pattern is set manually during network ramp-up, and is not dynamic.

The workgroup **IEEE802.15.4E** focuses on enhancing the MAC protocol proposed in IEEE802.15.4, while keeping the same physical layer. In its current proposal [11], nodes can switch between different hopping sequences. Similarly to TSMP, slots can be added/removed during the lifetime of the network. Note that an open-source implementation of this proposal, called TSCH, is presented in [12].

Summary. In summary, protocols and standards use channel hopping to combat narrow-band interference (e.g. in the presence of IEEE802.11 networks) and persistent multipath fading (mostly present in indoor deployments). All protocols use a TDMA approach to agree upon a schedule, which they follow blindly during network operation. There exists, to the best of our knowledge, no protocol which adapts the hopping pattern dynamically to the environment.

2.2 Gathering Connectivity Traces

Although most WSN deployments collect some sort of connectivity data for network health assessment or debugging, to our knowledge, only [7] presents data which is both sufficiently dense in time, and collected over all IEEE802.15.4 channels. To obtain that dataset, 44 nodes running TSMP [6] are deployed in an indoor printing facility for 28 days. The routing mechanism assigns two parents to each node, creating the routing di-graph depicted in Fig. 1.

We call a **link** two motes that can communicate directly (i.e. an arrow in Fig. 1); links are directional. Health reports indicate how many transmissions where attempted, and how many were successful. A link failure is detected when the transmitter senses an occupied channel before a transmission, or when it does not receive an acknowledgment after transmitting the data. Health reports are generated every $15min$ for each link and for each channel.

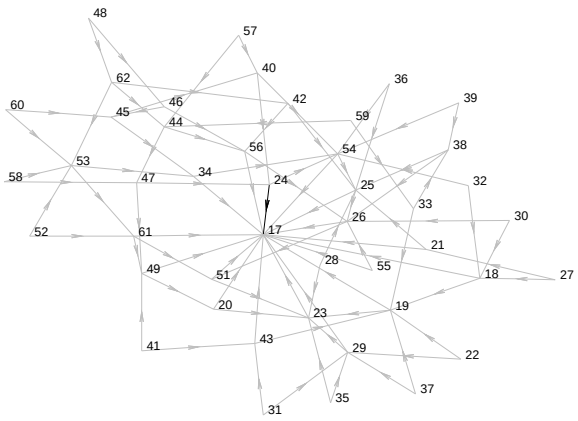

Figure 1: The routing di-graph over which the connectivity data is collected in [7]. Directed arrows represent active routing links (TSMP [6] assigns at least two routing parents to each node). Node 17 is the sink node.

We quantify the goodness of a link through its Packet Delivery Ratio ($\mathcal{P} \in [0..1]$), i.e. the ratio between the number of successfully transmitted packets and the number of transmission attempts. $\mathcal{P} = 0.7$ indicates that 70% of the packets transmitted on a link are received. Fig. 2 depicts the evolution of $\mathcal{P}$ with time for a given link, over all 16 channels. It indicates that activity inside the printing facility induces significant changes in Packet Delivery Ratio; this explains why $\mathcal{P}$ is stable over week-ends.

Fig. 2 illustrates the challenges faced by adaptive channel hopping. Overall, every channel undergoes deep fading periods, rendering single channel solutions inadequate[1].

The goal of adaptive channel hopping is to assess the Packet Delivery Ratio $\mathcal{P}$ for all channels, and to use this information to always communicate on a reasonably good channel. Let's assume, for link $24 \rightarrow 17$, that communication starts on channel 15. After a few hours, channel 15 features a low $\mathcal{P}$, and communication can switch for example to channel 26. After day 6, channel 26 experience poor performances, and the controller may switch to a different channel.

3. ADAPTIVE CHANNEL HOPPING

We call **controller** the entity which decides upon the channel to be used. The efficiency of this controller is evaluated in Section 4.

3.1 Controller Operation

Each node implements an instance of the controller for each link it participates in. We refer to a *slot* as a $15min$ time window[2].

[1]Strictly speaking, channel 17 on link $24 \rightarrow 17$ features a constant high $\mathcal{P}$. Nevertheless, a single channel MAC protocol assigns the same channel to all links, and, as shown in Section 4, there is no channel which works well at all times, for all links.

[2]This value is somewhat arbitrarily chosen as it is the time interval at which the dataset provides updates on the chan-

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7934
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7934

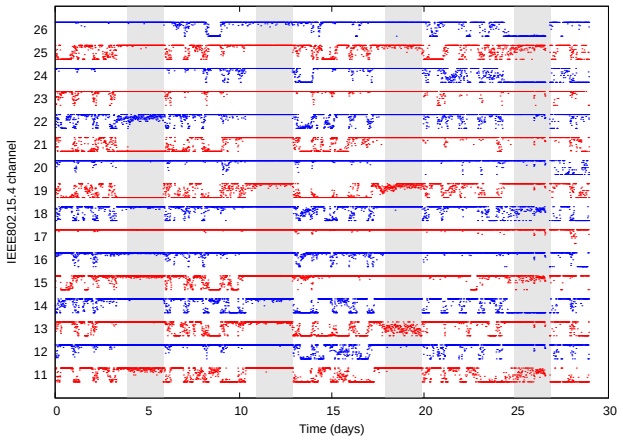

Figure 2: Packet Delivery Ratio evolving over time on the $24 \rightarrow 17$ link (colored black in Fig. 1), for all 16 channels. The grayed out time intervals represent week-ends.

Communication initiates on a randomly chosen channel c_i. Every k slots, the controller switches communication to channel $c_j \neq c_i$ for one slot. By counting the number of attempted and successful transmissions during that slot, the controller assesses the Packet Delivery Ratio $\mathcal{P}$ of channel c_j. We call **probing** the action of switching to a different channel for one slot to assess $\mathcal{P}$. After the probing slot, communication is resumed on channel c_i. k slots later, the controller probes channel $c_k = (c_j + 1)\,\%\mathcal{C}$ where $\%$ is the modulo operator and $\mathcal{C}$ the number of channel ($\mathcal{C} = 16$ for an IEEE802.15.4-compliant radio chip). Each channel is hence probed every $\mathcal{C} \cdot k$ slots.

The controller keeps track of the Packet Delivery Ratio $\mathcal{P}$ of the current channel c_i. Whenever $\mathcal{P}$ falls below a threshold $\mathcal{P}_{thres}$, the controller declares the performance of this channel to be unacceptably bad, and switches to the other channel which features the largest $\mathcal{P}$. We call **switching** the action of the controller to change the current channel resulting from a low $\mathcal{P}$.

3.2 Optimality

We employ a controller because $\mathcal{P}$ is an inherently dynamic and unpredictable metric. Since the nature of the proposed control scheme relies on the sporadic sampling of other channels, the approach tends to be suboptimal in the case for which one specific channel is the best option at all times. Additionally, optimality is treated in an informal fashion, realizing that optimal control of a system such as a WSN depends often on solutions to Hamilton-Jacobi partial differential equations [13]. Since the solutions of such equations in real-time do not appear to be feasible given the computational limitations imposed by the mote hardware, we employ the simplified control scheme presented above.

3.3 Packet Delivery Ratio $\mathcal{P}$ Estimation

The Packet Delivery Ratio $\mathcal{P}$ estimate of a channel is updated using (1), with $\mathcal{P}_{new}$ the updated estimate for a particular channel, $\mathcal{P}_{old}$ is the old estimate, and $\mathcal{P}_{measured}$ the estimate resulting from the probing.

$$\mathcal{P}_{new} = \alpha\mathcal{P}_{old} + (1-\alpha)\mathcal{P}_{measured} \quad (1)$$

The constant $\alpha \in [0, 1]$ is introduced as a weighting factor. A large value of alpha implies that the average Packet Delivery Ratio over time is more important than the one inferred from the last reading.

4. RESULTS

We evaluate the efficiency of the controller described in Section 3 onto the dataset presented in [7][3]. The controller initially picks a channel at random, and is allowed to evolve according to the dynamics described in Section 3.

4.1 Parameters Used

Results are obtained for $\alpha = 0.2$. This puts more weight on the new Packet Delivery Ratio estimate, as Fig. 2 indicates that $\mathcal{P}$ tends to change abruptly. The choice of α depends on how fast a link transitions from high to low $\mathcal{P}$.

We set $\mathcal{P}_{thresh} = 0.9$. $\mathcal{P}_{thresh}$ influences the link hysteresis: a lower $\mathcal{P}_{thresh}$ value leads to less less frequent switching, at the cost of communicating longer on a sub-optimal channel.

k indicates the probing period in slots. The larger the value of k, the less frequently probing occurs, and the less up-to-date the estimate of $\mathcal{P}$ for each channel are. Section 4.3 evaluates the impact of k.

4.2 Witnessing Adaptive Channel Hopping

Fig. 3 depicts the channel chosen by the controller on link $24 \rightarrow 17$. This plot features two components, distinguished by line thickness. The thin line shows the polling, i.e. the periodic evaluation of the different channels' Packet Delivery Ratio $\mathcal{P}$. Polling happens every $k = 20$ slots, or 5 hours. Polling is used to maintain statistics on the goodness of each channel. When the current channel becomes unacceptably bad ($\mathcal{P} < \mathcal{P}_{thresh}$), the information gathered during polling is used to select the best channel among all other channels. Switching is then used to communicate on a different channel. Fig. 3 shows that 9 switches occured on link $24 \rightarrow 17$.

4.3 Equivalent Packet Delivery Ratio $\bar{\mathcal{P}}$

The di-graph depicted in Fig. 1 contains 62 links. Over the course of the 28 days of the deployment, a link switches channels multiple times as the Packet Delivery Ratio of the current channel drops below $\mathcal{P}_{thres}$. We refer to the Equivalent Packet Delivery Ratio $\bar{\mathcal{P}}$ the Packet Delivery Ratio averaged over all links and over the 28 days of the deployment. $\bar{\mathcal{P}}$ indicates what fraction of packets one may expect to the sent successfully when choosing a random time and a random link. We determine $\bar{\mathcal{P}}$ for single channel and Adaptive Channel Hopping. A higher value of $\bar{\mathcal{P}}$ indicates a "better" channel.

nels' Packet Deliver Ratio $\mathcal{P}$.

[3]As an online addition to this paper, the dataset is made available by the authors at `http://wsn.eecs.berkeley.edu/replay/`.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7934
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7934

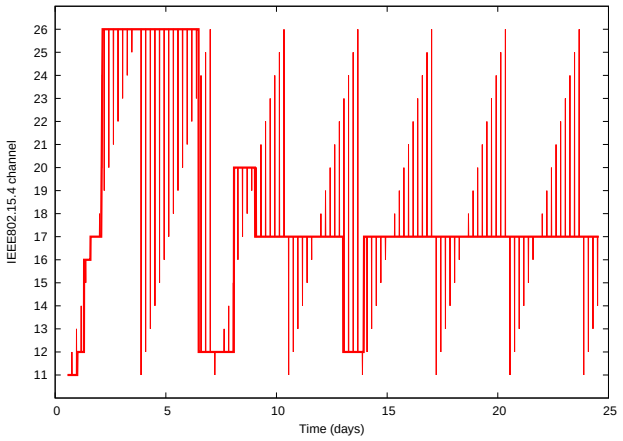

Figure 3: Polling – thin line – and switching – thick line – on link $24 \rightarrow 17$ (colored black in Fig. 1).

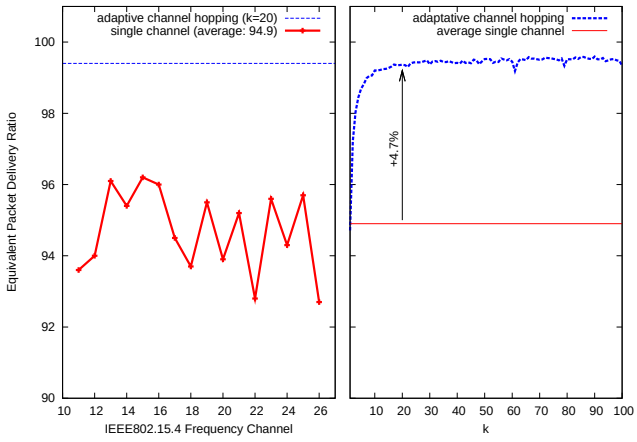

Figure 4: Equivalent Packet Delivery Ratio $\bar{\mathcal{P}}$ for a single channel operation (left) and adaptive channel hopping (right). Adaptive channel hopping performs better than single channel operation, on any channel.

Fig. 4 shows $\bar{\mathcal{P}}$ for single channel operation (left) and Adaptive Channel Hopping (right). As an example, when $k = 20$, Adaptive Channel Hopping features a $\bar{\mathcal{P}} = 99.4\%$. Note that Adaptive Channel Hopping performs better than a single channel solution, on any channel.

5. DISCUSSION

The controller designed in this paper is simple enough to be implemented on current hardware. It requires each node to maintain statistics for each channel and for each link it participates in. Considering a node which participates in 5 links with 16 available channels, if each statistic is an integer between 0 and 255, the controller requires 320 bytes of RAM memory. This is an acceptable overhead for a microcontroller which typically features 10kB of RAM [14]. We are currently working on implementing Adaptive Channel Hopping in an open-source version of TSCH [12].

The results presented in Section 4 indicate that Adaptive Channel Hopping performs better than using a single channel. These results are obtained over a set of data sufficiently large in number of nodes (44 nodes) and time (28 days) to be statistically valid. However, connectivity traces were gathered in a printing facility in which there is no external interference from technologies operating on the same frequency band (WiFi, Bluetooth). As a result, channels are stable for hours or days.

The results presented in this paper may not hold for very dynamic channels. This is the case when there is external interference (e.g. cause by a nearby 802.11 – WiFi – network) which causes *microscopic* dynamics and channels which are stable for only $500ms$ [15]. [16] shows that, in this case, blind channel hopping can be used. Blind Channel Hopping is implemented in TSMP [6], ISA100a [10], IEEE802.15.4E [11] and WirelessHART [9] and causes a link to switch channels every tens of ms, hopping evenly on all available channels.

A real-world deployment most likely features both macroscopic and microscopic dynamics. An issue worthwhile investigating is whether blind channel hopping can be coupled with adaptive channel hopping. This could be done through the use of whitelists. In such a system, the controller described in this paper could be used to identify the n best channels (the whitelist) over which a blind channel hopping protocol operates.

6. CONLUSIONS

This paper advocates the use of Adaptive Channel Hopping, in which nodes continuously evaluate the goodness of all available frequency channels in order to operate on a channel which performs reasonably well. We show how the use of a simple controller yields performances which are higher than a single channel solution operating on any channel. This controller is simple enough to be implemented on current hardware, and deals efficiently with slow-varying channels.

Results are obtained by replaying the behavior of the controller on traces gathered from a real-world deployment. The data set is sufficiently large in number of nodes (44 nodes) and time (28 days) to be statistically valid, and is dense in time. For such a *replay* approach to be widely adopted, a larger set of traces is needed, with traces collected in different environments.

Our current work includes designing a TinyOS application which does just that. It will be able to run on the different university testbeds scattered on different continents to eventually create a comprehensive database of traces. These can then be used by the community as a benchmark to evaluate the efficiency of protocols.

Acknowledgments

Branko Kerkez' work is supported by the NSF Graduate Fellowship program, and the NSF grant EAR 0725097. Thomas Watteyne's work is supported by the California Energy Commission. Mario Maglioccо's work is funded by NSF grant CMMI-0727726.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7934
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7934

7. REFERENCES

[1] *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, IEEE Std., Rev. 2006, 8 September 2006.

[2] CC2500, *CC2500, Low-Cost Low-Power 2.4 GHz RF Transceiver (Rev. B)*, Texas Instruments, Inc., 13 September 2007, data Sheet SWRS040B [available online].

[3] L. van Hoesel and P. Havinga, "A lightweight medium access protocol (LMAC) for wireless sensor networks: Reducing Preamble Transmissions and Transceiver State Switches," in *International Conference on Networked Sensing Systems*, 2004.

[4] O. D. Incel, S. Dulman, and P. Jansen, "Multi-channel Support for Dense Wireless Sensor Networking," in *First European Conference on Smart Sensing and Context (EuroSSC) Lecture Notes in Computer Science 4272. Springer.*, Enschede, the Netherlands, 25-27 October 2006, pp. 1–14.

[5] Y. Kim, H. Shin, and H. Cha, "Y-MAC: An Energy-efficient Multi-channel MAC Protocol for Dense Wireless Sensor Networks," in *International Conference on Information Processing in Sensor Networks (IPSN).* St. Louis, Missouri, USA: IEEE, April 22-24 2008, pp. 53–63.

[6] K. Pister and L. Doherty, "TSMP: Time Synchronized Mesh Protocol," in *Parallel and Distributed Computing and Systems (PDCS)*, Orlando, Florida, USA, 16-18 November 2008.

[7] L. Doherty, W. Lindsay, and J. Simon, "Channel-Specific Wireless Sensor Network Path Data," in *16th International Conference on Computer Communications and Networks (ICCCN).* Turtle Bay Resort, Honolulu, Hawaii, USA: IEEE, August 13-16 2007, pp. 89–94.

[8] *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements. Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)*, IEEE Std., Rev. 2005, 14 June 2005.

[9] *HART Field Communication Protocol Specifications, Revision 7.1, DDL Specifications*, HART Communication Foundation Std., 2008.

[10] ISA, *ISA100.11a:2009 Wireless systems for industrial automation: Process control and related applications*, International Society of Automation Std., 2009, [Draft standard, in preparation].

[11] C. S. Kang, K. H. Chang, R. Enns, K. Pister, L. Winkel, C. Powell, and J. Gutierrez, "MAC Enhancements for Time Slotted Channel Hopping," 14 May 2009, iEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs) [work in progress].

[12] T. Watteyne, D. Zats, and K. Pister, "TSCH: Time Synchronized Channel Hopping," under review.

[13] F. L. Lewis and V. L. Syrmos, *Optimal Control, 2nd Edition*, F. L. Lewis and V. L. Syrmos, Eds. Wiley-Interscience, 1995.

[14] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-Low Power Wireless Research," in *International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS).* Los Angeles, CA, USA: IEEE, April 2005.

[15] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis, "The beta-factor: Measuring Wireless Link Burstiness," in *6th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Raleigh, NC, USA, 5-7 November, 2008.

[16] T. Watteyne, A. Mehta, and K. Pister, "Reliability through frequency diversity: Why channel hopping makes sense," in *6th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*, Tenerife, Canary Islands, Spain, 26-30 October 2009.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7934
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7934