Exploring vs Exploiting: Enhanced Distributed Cognitive Coexistence of 802.15.4 with 802.11

Michael Timmers*†, Sofie Pollin*‡, Antoine Dejonghe*,
Liesbet Van der Perre*† and Francky Catthoor*†
*Interuniversity Micro-Electronics Center (IMEC), Kapeldreef 75, B-3001 Leuven, Belgium
†Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
‡UC Berkeley, 264M Cory Hall, Berkeley, CA 94720, USA
e-mail: michael.timmers@imec.be

***Abstract*— Thanks to recent advances in wireless technology, a broad range of standards are currently emerging. Interoperability and coexistence between these heterogeneous networks are becoming key issues, which require new adaptation strategies to avoid harmful interference. In this paper, we focus on the coexistence of 802.11 WLANs and 802.15.4 sensor networks in the ISM band. These networks have very different transmission characteristics that result in asymmetric interference patterns. We propose distributed adaptation strategies for 802.15.4 nodes to minimize the impact of the 802.11 interference. This interference varies in time, frequency and space and the sensor nodes adapt by changing their frequency channel over time. The proposed strategies use adaptive simulated annealing and find a proper balance between exploration and exploitation.**

I. Introduction

Interest in wireless technology has experienced an explosive growth over the last decades. Recent advances in processor design have enabled the use of wireless transceivers in small portable devices and even sensors. The finalization of a range of standards has eased the development of applications using those wireless functionalities. As a result, the spectrum is getting filled by heterogeneous devices, standards and applications. This is especially the case for the *Industrial, Scientific and Medical* (ISM) bands that are unlicensed and hence host the most diverse range of networks.

In this paper we focus on two standards that operate in the 2.4GHz ISM band: 802.11g Wireless LAN [1] and 801.15.4 Sensor Networks [2] as illustrated in Fig. 1. It is clear that the characteristics of both standards are very different and the problem is thus asymmetric in nature.

Indeed, the output power of 802.15.4 devices is usually as low as 0dBm [3], while the output power of 802.11g devices is typically 15dBm or above [4]. Also, 802.15.4 sensor networks are often designed to monitor the environment or buildings, and are typically very large. 802.11 networks are typically hotspots centered around an Access Point (AP), and hence more local. Finally, sensor network applications are not demanding in terms of throughput, but however require a high reliability and robustness against attacks or unknown events. They should also be self-organizing since it is impossible to maintain such large networks efficiently. 802.11 networks are typically used by a limited number of throughput-intensive applications.

Although recent studies have shown that sensor networks do have an impact on 802.11 networks [5], it is generally accepted that coexistence of both networks will affect the sensor networks most [6]. We will focus on a distributed protocol for 802.15.4 networks to optimize their performance under varying 802.11 interference. The proposed algorithms should be distributed in order to improve scalability, robustness and adaptability. More specifically, we will design distributed channel selection algorithms that allow the sensor nodes to dynamically adapt their channel selection to the current 802.11 interference pattern.

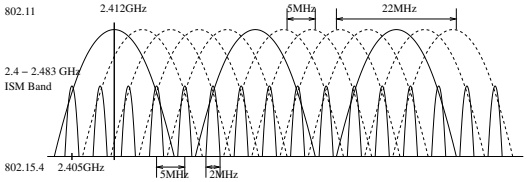


Fig. 1. 802.11 and 802.15.4 both operate in the 2.4GHz ISM band. This leads to coexistence issues, most prominently at the side of the ZigBee network.

Previously, we already concluded that *learning* the 802.11 interference can result in a large reduction both in power and hardware cost. The focus of this paper is to optimize the algorithms presented in [7] so that the performance gain is close to the optimal solution. This is done by a step-by-step analysis of the difficulties.

II. System Model

In this section we give a short overview of the models used for the sensor network, the WLAN interference, and the considered energy and performance metrics that are relevant for the considered scenario. For a more detailed overview of the assumptions made and the implementation details, we refer to [7].

A. 802.15.4 Network model

Sensor networks typically consist of a large number of nodes, e.g., to monitor the environment or for building automation. As a result, we represent the 802.15.4 network by a large number of nodes N that are arranged in a string or a rectangular topology. We assume that all sensors in the range Rd can overhear each other's beacons, where d is the inter-node distance. In Fig. 2 a simple string topology is presented with $R = 2$. Each of the sensor nodes is currently operating in one of the F possible channels. As explained above, $F = 16$ for 802.15.4 networks operating in the 2.4GHz ISM band. Every node transmits its beacon at this frequency, according to the beacon-enabled mode. We assume that the sensors potentially swap frequency every inter-beacon period, which ranges from 15ms to above 4min according to the standard. When and how to swap will be determined by the distributed adaptation algorithms.

B. 802.11 Interference model

We consider a large 802.15.4 network, that is affected by 802.11 interference. This 802.11 interference is assumed to vary over time, space and frequency.

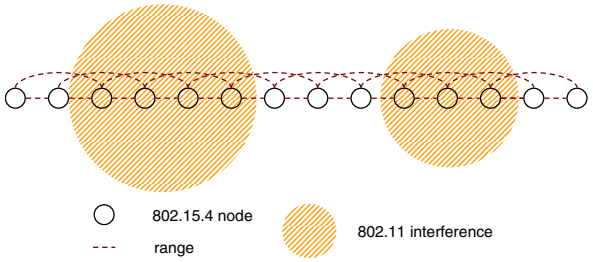


Fig. 2. The considered scenario is a string topology of 802.15.4 nodes, where nodes report to a sink that is placed at one side of the string. Interference is generated by WLAN devices and is dynamic in both time and space.

The variations in frequency result from the fact that different 802.11 networks can operate simultaneously using a different channel. Dynamic Frequency Selection (DFS) is a new functionality currently added to most of those 802.11 networks and Access Points. It is developed to optimize the frequency allocation of 802.11 networks that are subject to interference, which can be self-interference or interference resulting from other networks or devices. As a result of this adaptive and hence dynamic frequency selection, 802.11 networks are not operating at fixed channels over time.

It can be seen in Figure 1, that each 802.11 network always covers four consecutive channels of 802.15.4 networks. The power distribution over the channel is more or less flat, especially for the OFDM-based 802.11g networks [1].

The variation in time results from the fact that the activity of the 802.11 networks is dependent on the time-varying demand of its users [8]. From this study it can also be reasonably assumed that the 802.11 interference stays constant over several (thousands) of 802.15.4 inter-beacon periods.

The spatial variations result from the 802.15.4 deployment, which usually covers a very large area (e.g., for monitoring purposes). As a result, the 802.15.4 networks are expected to be large both in terms of the area they cover and the number of nodes they consist of. The theoretical transmission, and hence also interference, range of 802.11 networks is 100m to even 250m. Although this is a significant range, sensor networks can cover larger areas since they consist of a large number of nodes in a mesh topology. As a result, the 802.11 interference is assumed to affect large geographical subsets of the 802.15.4 network of nodes (Fig. 2).

As a last step, we assume that an active or interfering 802.11 network will always be detected by the 802.15.4 nodes [7].

C. Performance measures

We consider delay as a relevant performance metric (throughput requirements in sensor networks are typically low). More precisely, assuming that sensors monitor a variable that should be communicated to a central sink, we consider the number of periods required to forward a measurement to a fixed central sink. This average is computed over time and over the nodes in the network. The more the network is affected by the interference, the more periods are required on average to reach the sink. We assume that every packet is forwarded during each period, to the node closest to the sink that can be reached in that period. As a result, packets travel each period the largest possible distance.

As all the algorithms presented here are energy-optimal in view of the model presented in [7] (i.e., no parallel out-of-band scanning is assumed), we don't consider energy as a performance measure in this paper.

III. Problem Formulation

In the current paper, our goal is to minimize the average packet delay to the sink. This target function is unfortunately quite difficult to model, so we will approximate it by inspecting the optimal solution. In any learning system, this is a vital step in the solution design (see [9], Chapter I).

As mentioned in Section II, we assumed the sensor network to have a low duty cycle. In this situation a packet will have the least delay if it can travel the furthest distance each hop. Obviously, the optimal solution is then the one where all nodes share a common channel. In that case, we can ensure that each packet, independently of where it was generated, can travel the furthest distance and have an efficient last hop.

We can now come up with a simple approximation of our target function: we no longer look at the delay, but try to get all the nodes on the same channel. It is therefore obvious that any solution presented in this paper can then also be used to find a common control channel, which is of high importance for Cognitive Radio networks [11], [12].

IV. Benchmark Solution: Random Frequency Selection (RFS)

The simplest DFS algorithm is a scheme where nodes randomly (uniform distribution) pick a channel every period. It can of course be expected that the average delay in this scheme will be large. However, since it does not rely on any coordination between the nodes and does not rely on an environment model, it is very robust.

V. Improving over RFS

A. Learning Algorithms

In [7], we opted to improve over the random method using learning. This was shown to reach the same performance as scanning-based approaches. Hence, energy and hardware cost can be saved while keeping performance constant by learning.

Here, we will consider a simplified version of the Q-learning algorithm presented in [7]: the AR-filter. Contrary to Q-learning, the AR-filter maintains a one-dimensional table rather than a two-dimensional one. This has no impact on performance as the reward function used in [7] (eq. (4)) is actually independent of the current state of the node.

Our goal in this section is to learn the reward of selecting a certain channel. To do so, we first have to define our reward function. Like previously stated, we can use the following definition of the reward function as an approximation of the real target function:

$$R_w(f) = \begin{cases} \sum(\text{beacons heard}) + 1 & \text{if no energy detected} \\ 0 & \text{if energy detected} \end{cases}$$

The AR-filter will then try to estimate this reward function by updating its current estimate using the following rule:

$$\widehat{R}_w^{(k+1)}(f) = (1-\alpha)\widehat{R}_w^{(k)}(f) + \alpha R_w(f), \qquad (1)$$

where $\widehat{R}_w^{(k)}(f)$ is the estimate of the reward function at frequency f at interval k, α is the learning parameter and $R_w(f)$ is the current evaluation of the reward function.

It is important to note that the AR-filter updates the estimates, but in fact does not specify what actions should be taken. Instead of greedily optimizing this reward function, arbitrary experimentation is allowed. This is an important property in a time varying environment and allows decoupling the learning phase from the decision policy.

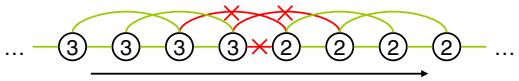


Fig. 3. Local optima can cause very large delays. Cooling down in a local optimum causes a disjunction between two sets of nodes that have converged on different channels. Any packet generated on the left-hand side will never reach the sink.

B. Exploration Algorithms

1) Background: The algorithms presented in [7] suffer from over-exploration when steady-state has reached. In this paper we will try to balance exploitation and exploration through simulated annealing.

Simulated annealing is a very effective heuristic optimization strategy for finding a global optimum. The basic idea of the method is to sample the search space using a Gaussian distribution and to *anneal* this distribution as the optimum is approached. In simulated annealing exploration is embedded in the algorithm to allow the system to jump out of a local optimum. We propose the following algorithm to select the next channel. When experimentation (i.e., learning) is allowed, we select a random frequency according to a uniform distribution. We define the probability to explore, p_e, in the current state as:

$$p_e(R,T) = e^{-\frac{R}{T}}, \tag{2}$$

where R is the reward to be maximized and T the temperature of the algorithm

We see that when T is high, the current solution will be selected randomly. But as we lower T according to some cooling scheme, exploration is reduced and eventually we converge into an optimum. We can see that the reward, R, has the opposite effect on exploration.

2) Reward-based (RB) exploration [7]: For now, we will keep the temperature, T, constant and not define a cooling scheme. Hence, the rate of exploration is only altered through the reward, $\widehat{R}_w^{(k)}$. If the reward of the current state is high, less exploration will be allowed. However, since no cooling scheme is defined, we will never eliminate exploring and nodes can still jump out of the optimal solution once in a while. We have seen that this effect actually can strongly reduce the efficiency (delay) in the steady-state. It is therefore necessary to find a cooling scheme.

3) Finding a cooling scheme: Finding a proper cooling scheme is a difficult task. If we just anneal and gradually drop the temperature to 0K, we risk getting stuck in local optima or are not able to adapt to the time-varying 802.11 interference. In many problems a trade-off is made between the speed of convergence and attaining the global optimum. However, in the current problem the local optima actually represent really dangerous situations. This is due to the fact that we are using a goal approximation (see Section III).

In Fig. 3 we can see that being stuck in a local optimum for the approximated goal destroys every opportunity to get the packet to the sink. Though exploring nodes might be able to bridge the packet over the boundary, a system that has been cooled down cannot. We conclude therefore that straightforward simulated annealing makes the reaching of the global optimum a constraint, rather than a trade-off. This makes the selection of a proper cooling scheme (if there is any at all) very difficult and we have to make for each scenario a very wise choice of all parameters involved. Even then, it can still happen that we get stuck in a local optimum. In the next section, we will discuss how we can devise a more robust algorithm.

4) Adaptive simulated annealing (ASA): The high-level idea of adaptive simulated annealing is to change the annealing rate, depending on the current situation. We implemented this as follows: if we seem to be converging, then we decrease our temperature quite rapidly. However, when we see a sudden change in our environment (e.g., an 802.11 network is activated), we reinitialize exploring.

As mentioned in Section V-B.3, we also have to protect the network against local optima. The way we approach this is rather similar to what we described above. Again, we will try to detect if we are in a local optimum and increase exploration rate of the boundary nodes. The boundaries are detected with routing information. Hence, we will see convergence centers created at the boundaries that ripple through the entire network chain. It is only when the different converged regions will agree on the channel that exploration is completely given up.

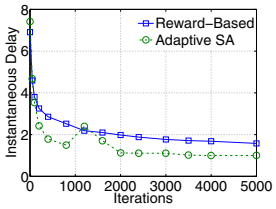
The detection algorithm now is not so easy anymore as in the interference case. As mentioned above, we have to rely on routing information. As it is known to the nodes whether other nodes are further or closer to the sink (in the routing table), we can use this information to detect boundaries. At a boundary a node will see a lot of nodes either closer to the sink or farther from the sink. If a boundary is detected, the boundary algorithm is initialized. It should be noted that the detection algorithm is only executed when the nodes have cooled down (after they have spent more than x iterations on a temperature lower than T_c). Otherwise, this detection algorithm would make a too quick decision. This would interfere with the AR algorithm and can cause non-convergence.

We will now detail the boundary algorithm. The boundary nodes re-initialize the temperature, going for heavy exploration. Furthermore, boundary nodes are not allowed to re-enter the previous channel for some time. The boundary algorithm is hence a complete exploration phase. There are two possibilities:

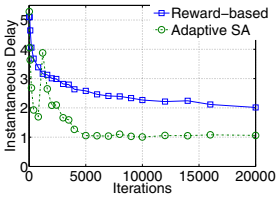
- one side of the boundary has been quicker to converge. They will detect the boundary sooner and begin exploration. Hence, they will go looking for the nodes on the other side of the boundary. If they find the other nodes, new boundary nodes are formed in the quicker converged center. These nodes will then go to the same channel as the previous nodes and, hence, one side of the boundary adopts the channel of the other (1-sided ripple).
- both sides of the boundary detect the boundary at almost the same time. The boundary nodes will converge on a third channel. At both sides new boundary nodes are formed and the new channel is rippled through both convergence centers (2-sided ripple).

VI. Simulation Results

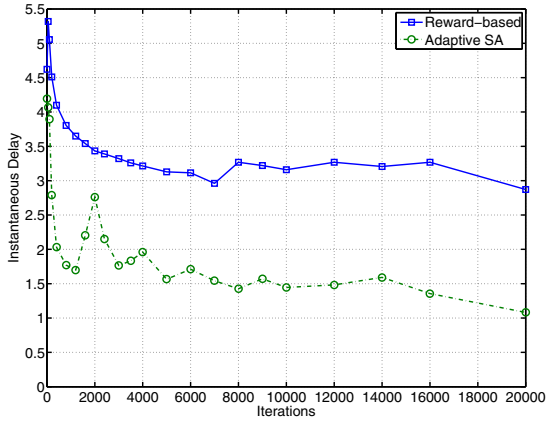
Due to space limitations, we only present a subset of the obtained results. In Fig. 4 the instantaneous delay for different situations is shown. Here, all 16 channels are assumed to be free of 802.11 interference. It can be clearly seen that the ASA algorithm outperforms the RB algorithm. Not only does it have an optimal steady state (no exploration and all nodes on the same channel), but it also converges significantly faster. However, we can see some spikes in Fig. 4. This is due to the local optima. Initially, some neighboring nodes settle quickly on a channel but enough nodes remain exploring to bridge possible boundaries. Later on, these bridging nodes settle down and a the algorithm gets stuck in a local optimum. As described in Section V-B.4, the local optimum is detected and new convergence centers are rippled through the network. This increases the range of the converged areas, while eventually all nodes will settle down on one channel. Obviously, the bigger the network size, the slower the convergence.



(a) 50 nodes in the network ($R = 10$)



(b) 100 nodes in the network ($R = 10$)



(c) 200 nodes in the network ($R = 10$)

Fig. 4. These are instantaneous delays for different size networks. All 16 ZigBee channels are assumed to be available. Simulation procedure is the following: we allow the algorithms to reach a certain point. At that point, a packet is generated in the network. The instantaneous delay is then the delay of this packet. In these figures, the delays are averaged over 400 packets per run and 100 runs are executed.

For reference, we also present the results from a scenario presented in [7]. We can clearly see that the ASA algorithm reaches an almost optimal solution, even in the presence of a dynamic 802.11 interference. The results are even better than might be expected from Fig. 4. This is due to the 802.11 interference. As the interference is easily detected and degrades all communication, the 802.11 interference reduces the available channels from 16 to 8. The presence of less available channels increases convergence speed. In the case that only one channel would be free for all nodes, any interference-aware algorithm will convergence almost directly on that channel.

VII. Conclusions

In this paper we have proposed some fully distributed robust channel selection algorithms for 802.15.4 networks in presence of dynamic 802.11 interference. The algorithms improve over the ones presented in [7] by letting the nodes settle down and freeze in a certain state. The algorithms can also be used to find a common control channel in a CR network. A next step is to fully implement the proposed algorithms in real protocols to investigate the impact of noise on the estimates.

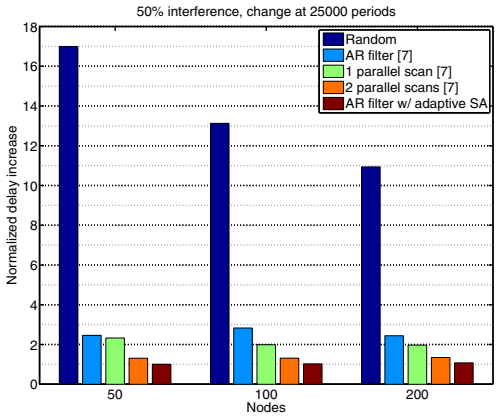


Fig. 5. Normalized delay increase compared to ideal channel allocation.

References

[1] TG 802.11g, *IEEE standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements Part II: wireless LAN medium access control (MAC) and physical layer (PHY) specifications*. IEEE-Std, 2003.

[2] S. Ergen. (2004) ZigBee/IEEE 802.15.4 summary. [Online]. Available: http://pages.cs.wisc.edu/ suman/courses/838/papers/zigbee.pdf

[3] K. Le. (2004, Nov.) Designing a zigbee-ready ieee 802.15.4-compliant radio transceiver. [Online]. Available: http://rfdesign.com/mag/411rfdf4.pdf

[4] A. Doefexi, S. Armour, L. Beng-Sin, A. Nix, and D. Bull, "An evaluation of the performance of ieee 802.11a and 802.11g wireless local area networks in a corporate office environment," in *IEEE International Conference on Communications*, vol. 2, May 2003, pp. 1196–1200.

[5] S. Pollin, I. Tan, B. Hodge, C. Chun, and A. Bahai, "Harmful coexistence between 802.15.4 and 802.11: A measurement-based study," in *Proc. of the International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, vol. 1, May 2008, pp. 1–6.

[6] S.-T. Center. Compatibility of IEEE 802.15.4 with IEEE 802.11, bluetooth and microwave ovens in 2.4ghz ism-band. [Online]. Available: http://www.ba-loerrach.de

[7] S. Pollin, M. Ergen, M. Timmers, A. Dejonghe, L. V. der Perre, F. Catthoor, I. Moerman, and A. Bahai, "Distributed cognitive coexistence of 802.15.4 with 802.11," in *Proc. of the International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, vol. 1, Mar. 2006, pp. 1–5.

[8] D. Tang and M. Baker, "Analysis of a local-area wireless network," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2000, pp. 1–10.

[9] T. M. Mitchell, *Machine Learning*. McGraw-Hill International Editions, 1997.

[10] M. Timmers, A. Dejonghe, L. V. der Perre, and F. Catthoor, "A distributed multichannel mac protocol for cognitive radios with primary user recognition," in *Proc. of the International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, 2007.

[11] T. Chen, H. Zhang, M. Katz, and Z. Zhou, "Swarm intelligence based dynamic control channel assignment in cogmesh," in *ICC Workshop on Cognitive and Cooperative Wireless Networks (COCONET)*, 2008, pp. 123–128.

[12] C. Doerr, D. Grunwald, and D. Sicker, "Local indepent control of cognitive radio networks," in *Proc. of the International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, 2008.