JAG: Reliable and Predictable Wireless Agreement under External Radio Interference

Carlo Alberto Boano[†], Marco Antonio Zúñiga[§], Kay Römer[†], and Thiemo Voigt[¶‡]

[†]Institute of Computer Engineering, University of Lübeck, Lübeck, Germany
[§]Embedded Software Group, Delft University of Technology, Delft, The Netherlands
[¶]Department of Information Technology, Uppsala University, Uppsala, Sweden
[‡]Swedish Institute of Computer Science, Kista, Sweden

E-Mail: [†]{cboano, roemer}@iti.uni-luebeck.de, [§]m.zuniga@tudelft.nl, [‡]thiemo@sics.se

***Abstract*—Wireless low-power transceivers used in sensor networks typically operate in unlicensed frequency bands that are subject to external radio interference caused by devices transmitting at much higher power. Communication protocols should therefore be designed to be robust against such interference. A critical building block of many protocols at all layers is *agreement* on a piece of information among a set of nodes. At the MAC layer, nodes may need to agree on a new time slot or frequency channel; at the application layer nodes may need to agree on handing over a leader role from one node to another. Message loss caused by interference may break agreement in two different ways: none of the nodes uses the new information (time slot, channel, leader) and sticks with the previous assignment, or – even worse – some nodes use the new information and some do not. This may lead to reduced performance or failures.**

In this paper, we investigate the problem of agreement under external radio interference and point out the limitations of traditional message-based approaches. We propose JAG, a novel protocol that uses jamming instead of message transmissions to make sure that two neighbouring nodes agree, and show that it outperforms message-based approaches in terms of agreement probability, energy consumption, and time-to-completion. We further show that JAG can be used to obtain performance guarantees and meet the requirements of applications with real-time constraints.

***Keywords*-Acknowledgement; Agreement; Handshake; JAG; Jamming; Radio Interference; Two Generals' Problem; Wireless Sensor Networks.**

I. Introduction

Wireless sensor nodes often need to agree on fundamental pieces of information that can drastically affect the performance of the entire network. For example, sensor nodes may need to agree on handing over a leader role from one node to another. An agreement failure would break the leader election, leading to a situation in which either more than one node becomes leader, or no leader is selected, causing reduced performance or failures in the network [1]. Similarly, at the MAC layer, several state-of-the-art protocols use time division multiple access (TDMA) or frequency diversity techniques to optimize their performance, in order to maximize network lifetime and minimize battery depletion. In such protocols, vital information such as the TDMA schedule, the channel-hopping sequence derived by interference-aware protocols, or the seed used to regulate the random channel hopping, need to be agreed upon by two or more sensor nodes in a reliable fashion. Failure to agree on such information correctly (e.g., nodes using inconsistent TDMA schedules) may disrupt network connectivity or substantially degrade performance.

When sharing information using an unreliable medium (such as wireless), no delivery guarantee can be given on the messages that are sent. Akkoyunlu et al. [2] have shown that, in an arbitrary distributed facility, it is impossible to provide the so called *complete status*, i.e., one cannot guarantee that two distributed parties know the ultimate fate of a transaction and whether they are in agreement with each other.

The problem is further exacerbated in the presence of external interference: the low-power transmissions of wireless sensor networks are highly vulnerable to interference caused by radio signals generated by devices operating in the same frequency range. Several studies have highlighted the increasing congestion of the unregulated ISM bands used by wireless sensor networks to communicate, especially the 2.4 GHz band [3]. Sensornets operating on such frequencies must cope with simultaneous communications of WLAN and Bluetooth devices, as well as with the electromagnetic noise generated by domestic appliances such as microwave ovens, video-capture devices, or baby monitors. As a result, wireless sensor nodes often communicate through interfered channels that have low chances of successfully delivering a packet. Hence, it is important to derive reliable techniques to ensure agreement even in the presence of interference, and make sure that they are efficient enough to meet the limited computational capabilities and energy resources of sensor nodes.

In this work, we design, implement, and evaluate JAG, a simple yet efficient agreement protocol for wireless sensor networks exposed to external interference. JAG introduces a jamming sequence as the last step of a packet handshake between two nodes to inform about the correct reception of a message carrying the information to be agreed upon. The key insight behind this approach is that detecting a jamming sequence in the presence of external interference is more reliable than using acknowledgement (ACK) packets to verify whether the information was successfully shared.

In environments that experience high levels of external interference, the probability of successfully transmitting a sequence of packets and completing an handshake is small, even when using short ACK packets. Despite the minimal amount

of information they carry, acknowledgements are embedded into IEEE 802.15.4 frames, and hence can be destroyed if any of the bits in the header, payload, or footer is corrupted by interference. Performance can be improved by means of redundancy (i.e., by sending multiple ACK packets), but this results in a significantly higher energy expenditure and latency, which is undesirable when using resource-constrained wireless sensor nodes.

Using JAG, instead, one can minimize the energy expenditure and provide agreement guarantees under weaker and more realistic assumptions about the underlying interference pattern compared to message-based approaches. By appropriately tuning the length of the jamming sequence, one can parametrize JAG to obtain predictable performance and to guarantee agreement in a finite amount of time, even in the presence of external interference: a perfect fit for applications with timeliness requirements. We focus on the unicast case (agreement between two neighbouring nodes) and show that JAG outperforms traditional packet-based agreement protocols in the presence of interference with respect to agreement probability, energy consumption, and time-to-completion.

JAG is intended as a building block to construct protocols at different layers of the protocol stack. It could be embedded into a MAC protocol to agree on time slots or frequency channels as discussed in Sect. VII, at the transport level to agree on connection establishment or tear-down, or at the application level to agree on handover of a leader role.

Our paper proceeds as follows. Sect. II defines the agreement problem in wireless sensor networks challenged by external radio interference. Sect. III conveys the main idea of the paper: using jamming as a binary signal for acknowledging the reception of packets. Thereafter, in Sect. IV, we illustrate JAG, a protocol for reliable agreement under external radio interference. We describe how JAG can provide the desired quality of service (QoS) in Sect. V, and we experimentally evaluate the performance of JAG under interference in Sect. VI. After discussing the integration of JAG into existing sensornet MAC protocols in Sect. VII, we review related work in Sect. VIII and conclude our paper in Sect. IX.

II. Problem Definition

Agreeing on a given piece of information is a classical coordination problem in distributed computing. The *Two Generals' Agreement Problem*, formulated by Jim Gray to illustrate the two-phase commit protocol in distributed database systems [4], is often used to explain the challenges when attempting to coordinate an action by communicating over a faulty channel, and can be described as follows.

Two battalions are encamped near a city, ready to launch the final attack. Because of the redoubtable fortifications, the attack must be carried out by both battallions at the same time in order to succeed. Hence, the generals of the two armies need to agree on the time of the attack, and their only way to communicate is to send messengers through the valley. The latter is occupied by the city's defenders, and a messenger can be captured and its message lost, i.e., the communication channel is unreliable. Since each general must be aware that the other general has agreed on the attack plan, messengers are used also to exchange acknowledgements. However, because the acknowledgement of a message receipt can be lost as easily as the original message, a potentially infinite series of messages is required to reach an agreement[1].

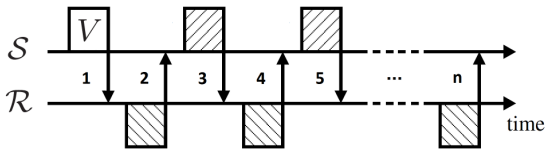


Fig. 1. n-way handshake between nodes $\mathcal{S}$ and $\mathcal{R}$.

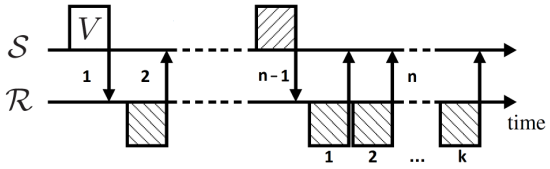


Fig. 2. Enhanced n-way handshake between nodes $\mathcal{S}$ and $\mathcal{R}$ using redundancy: the last ACK message is transmitted k times.

A. Agreement in Wireless Networks

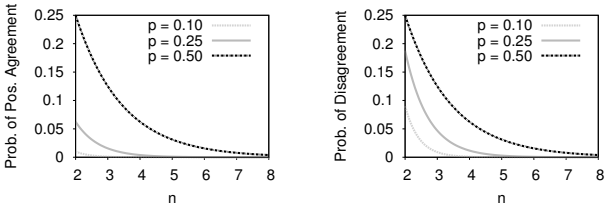
In the context of wireless communications, the problem can be rephrased as follows. When two nodes, $\mathcal{S}$ and $\mathcal{R}$, need to agree on a common value V, they exchange a sequence of n messages in an alternating manner (Fig. 1). Node $\mathcal{S}$ is the initiator of the exchange. After the transmission of V, each subsequent message acknowledges the receipt of the previous message, i.e., a node sends message $i > 1$ only if it correctly received message $i-1$. Each node uses a simple rule to determine the success of the exchange: if all expected messages are received, the exchange is deemed successful, otherwise the exchange is deemed unsuccessful.

The scenario described above corresponds to an *n-way handshake* between nodes $\mathcal{S}$ and $\mathcal{R}$, where n is the number of packets exchanged. The n-way handshake is a widely used mechanism in communication networks. For example, TCP employs a 3-way handshake ($n=3$) to establish connections over the network, whereas IEEE 802.11i (WPA2) uses a 4-way handshake ($n=4$) to carry out the key exchange.

An n-way handshake can have three possible outcomes:

1) **Positive Agreement.** The n messages are all received correctly, and both nodes deem the exchange as successful, accepting V.
2) **Negative Agreement.** A message m with $m < n$, i.e., a message prior to the last message n, is lost. None of the nodes receives all the expected messages, hence both nodes deem the exchange as unsuccessful, discarding V.
3) **Disagreement.** The last message n is lost. One of the two nodes receives all the expected messages, deems the exchange as successful and accepts V; whereas the second node misses the last message and therefore deems the exchange as unsuccessful, rejecting V.

[1]A different problem that we are not addressing in this work is how to guarantee the identity of the sender of the message, as well as how to cope with misbehaving parties.



(a) Positive Agreement (b) Disagreement

Fig. 3. Distribution of the probabilities of positive agreement and disagreement of the n-way handshake shown in Fig. 1 as a function of the probability of successful packet transmission p and length of the handshake n.

In the original two generals' scenario, a *positive agreement* would lead to a simultaneous attack of the city by both battalions and a consequent victory, a *negative agreement* would cause both battalions to stall, while a *disagreement* would trigger the attack of only one battalion and a consequent defeat of the attacking forces.

While *disagreements* are potentially fatal, *negative agreements* are often less severe. For example, if the shared value contains the next channel to be used for communication, two nodes are better off staying in the same lossy channel, rather than having only one of them move to a different frequency. The probability of negative agreements should, however, be minimized, as it may lead to reduced performance. Hence, an agreement protocol should strive to minimize disagreement as a first priority, maximize positive agreements as a second (almost equally high) priority, and minimize negative agreements as a third (substantially lower) priority. A metric to measure the quality of an agreement protocol (whose value should be minimized) is therefore the *DPA ratio* of the probability of disagreements over the probability of positive agreements.

B. The importance of the last message

It is important to emphasize that, in an n-way handshake, disagreements only occur if the last message is lost. Hence, depending on the application, it may be desirable to devote extra-resources to increase the successful delivery of the last packet by means of redundant packet transmissions (i.e., repeating a message several times and assuming successful transmission if at least one copy is received).

A possibility is to employ a n-way handshake in which the last packet is repeated k times, as shown in Fig. 2. Using this approach, the final outcome of the handshake is strongly dependent on the link quality, on the length n of the n-way handshake, and on the redundancy factor k. Letting p represent the probability that a generic message is successfully received (assuming that p remains constant over time and that it is independent for each packet), and $q = 1 - (1-p)^k$ the probability of successfully receiving at least one of the k redundant packets, we obtain:

$$\begin{aligned} \text{Prob}(PositiveAgreement) &= p^{n-1}q \\ \text{Prob}(NegativeAgreement) &= 1 - p^{n-1} \\ \text{Prob}(Disagreement) &= p^{n-1}(1-q) \end{aligned}$$

These equations show that in order to maximize the frequency of positive agreements and, at the same time, minimize the frequency of disagreements, we need to maximize the link quality p and maximize the level of redundancy k. The choice of a suitable n becomes a catch-22 dilemma in the presence of unreliable links, as illustrated in Fig. 3: long n-way handshakes minimize the probability of disagreement, but also the probability of positive agreement, whereas short n-way handshakes maximize the probability of positive agreement, but also the chances of disagreement.

C. Agreement in Wireless Sensor Networks Challenged by External Interference

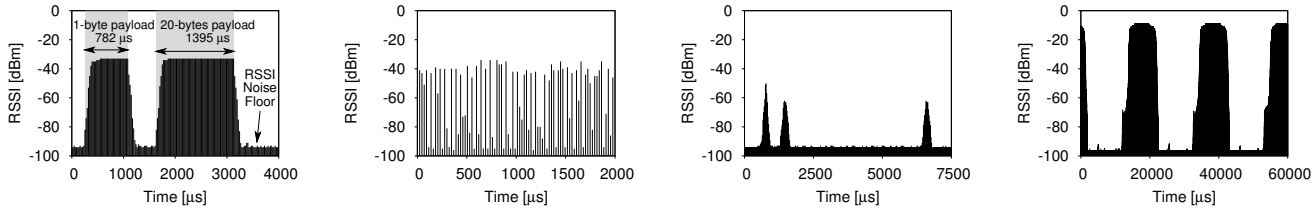
In the context of wireless sensor networks, minimizing the amount of exchanged packets is mandatory because of the limited energy resources available, i.e., sensor nodes need to minimize the time during which the radio is active as much as possible. Therefore, the use of redundant packet transmissions and long handshakes is not advisable, as it would increase the energy consumption.

Another aspect is the channel quality affecting p. Wireless sensor nodes operate in the unlicensed ISM radio bands, and often use a very low transmission power, which makes them vulnerable to external interference. Any wireless appliance operating in the same frequency range of sensornets can potentially interfere with their communications and decrease the probability of a successful packet exchange p. In the 2.4 GHz ISM band, for example, Wi-Fi and Bluetooth networks, as well as domestic appliances such as microwave ovens, can create noise levels that overwhelm the interference resistance capabilities of DSSS radios and radically decrease the packet reception rate [3], [5]. Hence, we need to investigate ways to encode transmissions such that their success probability p is maximized despite interfered channels.

D. Analysis of Common Interference Sources

In order to understand the impact of external interference on the probability of successful transmission p in wireless sensor networks communications, we study the interference patterns produced by common devices operating in the 2.4 GHz ISM band. Using Sentilla Tmote Sky nodes employing a CC2420 radio, we perform a high-speed sampling of the RSSI register (≈ 50 kHz as in [6]). We call this operation *fast RSSI sampling* over a time window t_{samp}. Fig. 4 shows the outcome of fast RSSI sampling in the presence of sensornet communications and external interference.

Absence of external interference. When neither interference nor IEEE 802.15.4 communications are present, the fast RSSI sampling returns the so called RSSI noise floor. The latter has typically values in the proximity of the radio sensitivity threshold (e.g., in the range $[-100, -94]$ dBm for the CC2420 radio). In the presence of IEEE 802.15.4 communications, the fast RSSI sampling returns a stable value corresponding to the strength and the length of the transmitted packet (Fig. 4(a)). As packets have a constrained maximum payload size of 127 bytes according to the 802.15.4 PHY standard, a packet transmission at 250 Kbit/sec would not last more than 4.3 ms.



(a) IEEE 802.15.4 packets (b) Heavy Wi-Fi Interf. (c) Bluetooth interf. (d) Microwave oven interf.

Fig. 4. RSSI values measured using off-the-shelf wireless sensor nodes operating in the 2.4 GHz ISM band. Please notice the different scale of the x-axis.

Presence of external interference. When other devices operating in the same frequency band of wireless sensor networks are active, bursts of interference signals (*busy periods*) alternate with instants in which the channel is clear (*idle periods*). The strength of the interference signals and the duration of idle and busy periods depend on the interfering source and on the specific context. For example, the interference patterns generated by Wi-Fi transmissions depend on the number of active users and their activities, as well as on the traffic conditions in the backbone.

Wi-Fi transmissions are typically much stronger than sensornet transmissions, and can affect several IEEE 802.15.4 channels at the same time. Hauer et al. [7], [8] have shown that with a sufficiently high sampling rate, one can identify the short instants in which the radio medium is idle due to the Inter-Frame Spaces (IFS) between 802.11 b/g packets. Fig. 4(b) shows the outcome of fast RSSI sampling in the presence of heavy Wi-Fi interference (caused by a file transfer): it is indeed possible to identify RSSI values matching the radio sensitivity threshold between consecutive Wi-Fi transmissions.

Fig. 4(c) shows an example of interference generated by Bluetooth. The latter uses an Adaptive Frequency Hopping mechanism to combat interference, and hops among 1-MHz channels around 1600 times/sec., hence it remains in a channel for at most 625 μs. Since Bluetooth channels are more narrow than the ones defined by the 802.15.4 standard, it may happen that communication in multiple adjacent Bluetooth channels affects a single 802.15.4 channel.

Fig. 4(d) shows an example of the interference pattern caused by microwave ovens: high-power noise ($\approx$ 60 dBm) is emitted in the 2.4 GHz frequency band in a very periodic fashion. The period mostly depends on the power grid frequency, but can also slightly vary depending on the oven model. Works in the literature report a power cycle of roughly 20 ms (at 50 Hz) or 16 ms (at 60 Hz) with an active period of at most 50% of the power cycle [6], [9].

E. The Role of Idle Periods

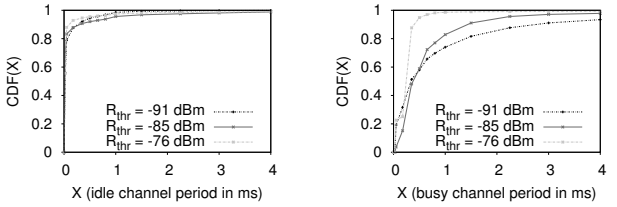
In the presence of external interference, n-way handshakes need to take advantage of idle periods. In principle, the longer the idle period and the shorter the handshake, the higher the likelihood of obtaining positive agreements. However, the interplay between idle periods and n-way handshakes is complex because of the particular patterns of each interfering source. Some devices, such as microwave ovens, generate periodic interference patterns with relatively long idle periods (Fig. 4(d)), while others, such as Wi-Fi stations, generate interference patterns with short idle periods of a highly variable length (Fig. 4(b)).

Having short idle periods reduces the probability of successfully completing a handshake, and this is especially critical in the presence of heavy Wi-Fi interference. Fig. 5 shows the cumulative distribution function (CDF) of idle and busy periods measured by a Maxfor MTM-CM5000MSP node in the presence of a laptop continuously downloading a file from a nearby access point. A channel is defined as busy if the RSSI is higher or equal than a configurable threshold R_{thr} and idle otherwise. In such a scenario, the probability of having an idle period longer than 2 ms is smaller than 5%. Therefore, there is only a little chance that a message-based handshake successfully completes within an idle period. In order to escape interference, one would need to use short messages and send them as close as possible to each other, in order to increase the chances of fitting into an idle period.

Off-the-shelf IEEE 802.15.4-compliant radios such as the CC2420 offer the ability to automatically generate and send ACKs for data frames in hardware. The advantage of hardware acknowledgements is a significant reduction of latency compared to solutions in which the ACK is generated via software [10]. However, hardware ACKs cannot be used to carry out a complete n-way handshake (with $n > 2$), since they cannot be used in reply to another hardware ACK. Imagine a node $\mathcal{S}$ starting a handshake by sending a message to $\mathcal{R}$. The latter can reply with a hardware ACK, but $\mathcal{S}$ will have to receive and extract the packet, analyse its validity, as well as to prepare a new ACK frame, load it into the buffer, and send it over-the-air[2]. This may cause long latencies that break the agreement in the presence of short idle periods.

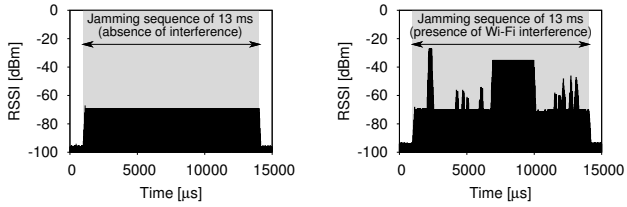
Furthermore, it is also highly inefficient to encode the binary information carried by an ACK message inside an IEEE 802.15.4 frame, especially in the presence of interference. Despite the payload contains only a single ACK bit, the whole packet consists of synchronization preamble and a physical header (4-bytes preamble, 1-byte Start of Frame Delimiter (SFD), 1-byte length field), as well as a MAC header and footer (2-bytes frame control, 1-byte sequence number, 4-20-bytes address, 2-bytes Frame Check Sequence (FCS)). If any of the bits in the headers and preamble is corrupted by interference, the packet may become undecodable [11], [12].

[2]In case a train of k redundant software ACKs is sent, the packet can be loaded into the buffer once and sent repeatedly.



(a) Idle periods (b) Busy periods

Fig. 5. Cumulative distribution function (CDF) of idle and busy periods measured by a Maxfor MTM-CM5000MSP node in the presence of a laptop continuously downloading a file from a nearby access point.



(a) Absence of Interference (b) Presence of Wi-Fi Interference

Fig. 6. RSSI values measured by a Maxfor MTM-CM5000MSP node during the transmission of a jamming sequence in absence of interference (a), and in the presence of external Wi-Fi interference (b).

Therefore, instead of encoding the last ACK as packet transmission, we propose to encode it by means of **jamming**, where the presence of a jamming sequence signals the receipt of the previous message. The key advantage of this approach is that jamming, as generated by off-the-shelf wireless sensor nodes, can be reliably detected even under interference.

III. Jamming as Binary ACK Signal

We propose to encode the last acknowledgement of a n-way handshake by means of **jamming** (i.e., transmission of a carrier signal), where the presence of a jamming sequence signals the receipt of the previous message. The key advantage of this approach is that precisely timed jamming signals can be generated using off-the-shelf wireless sensor nodes and can be reliably detected even under heavy interference.

A. Generating a Jamming Sequence

In a recent study, we showed that off-the-shelf radios can be used to generate controllable and repeatable jamming signals in specific IEEE 802.15.4 channels by transmitting a modulated or unmodulated carrier signal that is stable over time [6], [13]. This approach is superior to packet-based jamming, as the generated signal is independent of both packet sizes and inter-packet times. We hence generate precisely timed jamming signals by configuring the MDMCTRL1 register, so that the CC2420 radio outputs a continuous modulated carrier signal. The detection of the latter is based on high-frequency RSSI sampling, as discussed next.

B. Detecting a Jamming Sequence

Common radio chips offer the possibility to read the RSSI in absence of packet transmissions. Several researchers have shown that it is a useful way to assess the noise and the level of interference in the environment [5], [8], [14]. RSSI readings close to the sensitivity threshold of the radio indicate absence of interference, whereas values above this threshold identify a packet transmission, or a busy/congested medium (see Fig. 4).

Hence, we use the fast RSSI sampling mechanism mentioned in Sect. II-D to detect the presence or absence of a jamming signal generated by a sensor node. A jamming sequence generated using the method described in Sect. III-A results in a stable RSSI value above the sensitivity threshold of the radio, as shown in Fig. 6(a). Therefore, one can detect if a jamming signal was transmitted by making sure that no RSSI sample falls down to the sensitivity threshold of the radio.

In the presence of additional external interference, the RSSI register will return the maximum of the jamming signal and the interference signal due to the co-channel rejection properties of the radio [6]. Fig. 6(b) illustrates this for a jamming signal sent in the presence of Wi-Fi interference. As we have shown in Sect. II-D, typical interference sources – in contrast to our jamming signal – do not produce continuous interference for long periods of time, rather they alternate between short idle and busy periods. That is, *if the jamming signal lasts longer than the longest busy period of the interference signal, we are unequivocally able to detect the absence of the jamming signal* by checking if any of the RSSI samples equals the sensitivity threshold of the radio. We exploit this property to design JAG, a protocol for reliable agreement under external interference.

C. Identification of the Interfering Source

While a jamming signal can encode the binary acknowledgement information, it cannot encode the identities of sender and receiver as a regular packet would. When carrying out a handshake, however, these identities are already included in the message V to be acknowledged, and therefore are implicitly known to the two nodes, as long as the communication channel remains allocated exclusively for the whole duration of an exchange. In this way, intra-network interference is avoided, and a jamming sequence acknowledging the reception of V can be identified reliably by means of an RSSI threshold, as we discuss in Sect. IV. Any protocol that embeds JAG as a building block for agreement needs to meet this requirement. At the MAC layer, RTS/CTS can be used to allocate the channel in CSMA protocols, whereas in TDMA protocols the timeslots must be long enough to complete an exchange.

IV. JAG: Reliable Agreement under Interference

We call *JAG (Jamming-based AGreement)* the three-way handshake in which the last ACK is sent in the form of a jamming signal as shown in Fig. 7. The choice of three-way handshakes (as opposed to two-way) is motivated by two facts. First, a three-way handshake increases the reliability of identifying the jamming signal because it provides a reference RSSI value (this will be explained in more detail in Sect. IV-B). Second, three-way handshakes avoid disagreements due to asymmetric links: for instance, if $\mathcal{S}$ has a link with $\mathcal{R}$ but the reverse link is not present, a two-way handshake would always lead to disagreements, since $\mathcal{R}$ is not able to confirm the reception of V.

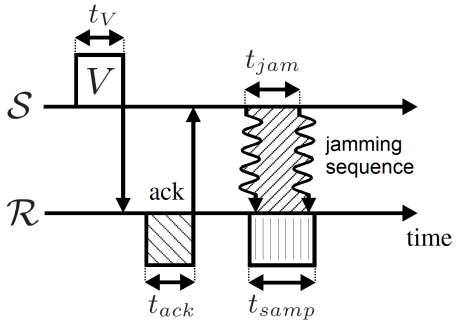


Fig. 7. Illustration of JAG: the last acknowledgement of the 3-way handshake between nodes $\mathcal{S}$ and $\mathcal{R}$ is sent in the form of a jamming signal.

A. Protocol Design

The protocol proceeds as follows. $\mathcal{S}$ initiates the exchange and sends the information V towards a receiver $\mathcal{R}$. If V is successfully received, $\mathcal{R}$ saves the signal strength r_s of the received packet and sends an ACK message back to $\mathcal{S}$. We can send either hardware or software acknowledgements: in the remainder of this paper we assume that hardware ACKs are available. If $\mathcal{S}$ receives the acknowledgement, it transmits a jamming signal for a period t_{jam}. Meanwhile, $\mathcal{R}$ carries out a fast RSSI sampling for a period $t_{samp} \leq t_{jam}$ that is synchronized in such a way that the fast RSSI sampling is carried out while the jamming signal is on the air. The message V is used as the synchronization signal: given that clock drift is not too high at timescales of a few milliseconds, it is sufficient to include a short safety margin to compensate for drift (more details in Sect. IV-D). For simplicity, in the rest of the paper, we assume $t_{jam} = t_{samp}$.

If $\mathcal{R}$ detects the presence of the jamming signal, it deems the exchange as successful; otherwise, V is discarded. $\mathcal{S}$ deems the exchange as successful if the ACK is received within a short timeout period, otherwise the jamming sequence is not generated and the handshake immediately terminated.

After the reception of V, node $\mathcal{R}$ carries out a fast RSSI sampling as described in Sect. III to detect the absence or the presence of the jamming sequence transmitted by $\mathcal{S}$. The method to detect the jamming signal is simple: if a jamming sequence is sent, *all* RSSI samples should be above r_{noise}, with the latter being the RSSI noise floor threshold of the radio. Hence, if during t_{samp} we observe *at least one RSSI sample* with a value comparable to r_{noise}, we conclude that the jamming sequence was not transmitted.

This process can be described as follows. Denoting $\{x_1, x_2, \ldots, x_n\}$ as the sequence of RSSI values sampled during t_{samp}, we define the binary sequence $\{X_1, X_2, \ldots, X_n\}$ as follows: if $x_i \leq r_{noise}$, then $X_i = 1$, else $X_i = 0$. $\mathcal{R}$ makes a decision about the presence of the jamming sequence as follows: if $\sum_{i=1}^{n} X_i = 0$, then $\mathcal{S}$ was transmitting a jamming signal and hence V is accepted; otherwise, V is discarded.

Using this algorithm, JAG would operate correctly and would be able to recognize the presence or absence of a jamming signal reliably. However, we can enhance its performance significantly by exploiting the knowledge of the received signal strength r_s of the packet containing V.

B. The Role of r_s

Under the hypothesis that the jamming signal has a reasonably similar signal strength to r_s (RSSI does not change significantly between consecutive transmissions spaced by only a few milliseconds), $\mathcal{R}$ can filter out any interference source weaker (i.e., resulting in an RSSI range smaller) than $(r_s - \Delta_r)$, with Δ_r being a tolerance margin to compensate for the inaccuracy of low-power radios and the instability of the RSSI readings. This allows to shorten t_{jam} and achieve a higher energy-efficiency: as we can see in Fig. 5(b), the higher R_{thr}, the shorter the duration of busy periods.

Hence, if $(r_s - \Delta_r) > r_{noise}$, JAG's algorithm is executed as follows: if $x_i < (r_s - \Delta_r)$, then $X_i = 1$, else $X_i = 0$. $\mathcal{R}$ still makes a decision about the presence of the jamming sequence in the following way: if $\sum_{i=1}^{n} X_i = 0$, then $\mathcal{S}$ was jamming and hence V is accepted; otherwise, V is discarded.

Furthermore, r_s also increases the reliability of fast RSSI sampling. The maximum distance over which a packet can be successfully received and decoded is shorter than the distance over which a jamming signal can be captured. This may lead to confusion in a scenario in which two nodes that cannot communicate with each other are allocated the same time slot in a TDMA protocol and transmit a message concurrently. By using a threshold r_s, we make sure that a receiver $\mathcal{R}$ is in the communication range of $\mathcal{S}$, and therefore r_s cannot be achieved by any other node transmitting simultaneously.

C. The Role of t_{jam}

The length of the jamming sequence t_{jam} can be tuned in order to provide probabilistic guarantees on the fraction of disagreements. Denoting t_{busy}^{max} as the maximum busy period that can be encountered in the presence of interference, we can guarantee that $\mathcal{S}$ and $\mathcal{R}$ will agree on V by setting $t_{jam} > t_{busy}^{max}$. In such a case, an idle period will surely be encountered during t_{samp}, and the absence of a jamming sequence unequivocally detected, as discussed in Sect. III-B. Hence, the most pernicious outcomes (disagreements) are eliminated, and only positive or negative agreements can occur.

In some scenarios, however, one may need to know the outcome of the agreement process before t_{busy}^{max}. In these cases, where $t_{jam} \leq t_{busy}^{max}$, disagreements may occur. For these type of scenarios, given t_{jam}, we derive an upper bound for the probability of obtaining disagreements. In this way, a user with stringent real-time constraints can assess if the fraction of disagreements is within the limits permitted by the QoS requirements of the application. The probabilistic model bounding the fraction of disagreements is presented in Sect. V.

D. JAG Implementation

We implement JAG on Maxfor MTM-CM5000MSP and Sentilla Tmote Sky nodes. Our implementation, based on Contiki [15], uses two main building blocks: the generation of a jamming sequence and the high-frequency RSSI sampling. The former uses the CC2420 transmit test modes as described in Sect. III-A. The latter is implemented as in our previous work [6], so that we roughly obtain one RSSI sample every

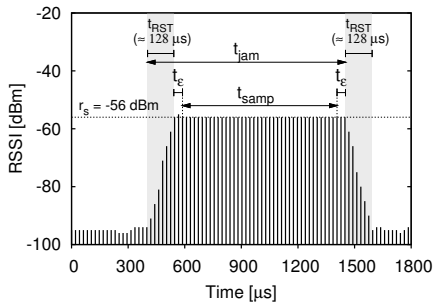


Fig. 8. Alignment between t_{samp} and t_{jam}: RSSI readings obtained during t_{RST} and t_ϵ are discarded to compensate for synchronization inaccuracies.

20 μs. Although a sampling rate of 50 kHz does not capture the transmissions from all wireless devices operating in the same frequency band of sensor networks (e.g., IEEE 802.11n devices), it is still enough to identify most of the idle periods that occur between Wi-Fi transmissions and hence to distinguish the jamming sequence from external interference.

For all our experiments we use NULLMAC, a MAC layer that just forwards packets to the upper or lower protocol layer and does not perform any duty cycling, but reports the presence of hardware acknowledgements. We chose NULLMAC in order to obtain results that are independent of specific MAC features and parameters. To ensure that the execution time of the entire handshake is bounded and independent of clear channel assessment (CCA) back-off times, we do not postpone transmissions until the channel becomes clear. Instead, we carry out a single clear channel assessment before sending V: if the channel is found busy, the transmission is cancelled. This is an optimization, as sending V despite the busy channel would result in a negative agreement (V would be lost).

To ensure alignment between jamming t_{jam} and sampling t_{samp}, we implement a simple synchronization mechanism. $\mathcal{S}$ and $\mathcal{R}$ synchronize their operations based on the reception of V: the transmission or reception of the Start of Frame Delimiter (SFD) is used as the synchronization signal. Although at timescales of a few milliseconds clock drift is minimal, the beginning of t_{samp} may not be aligned with the beginning of the jamming sequence because of the time required for RSSI to settle. The RSSI of the CC2420 radio is indeed an average of the last 8 bit symbols [6] and hence one needs to wait for the RSSI to stabilize (this takes $\approx t_{RST} = 128\mu$s) before being able to measure r_s (see Fig. 8). Since RSSI readings are not instantaneous and their duration may slightly differ among different nodes, we introduce a safety margin t_ϵ during which the RSSI readings are discarded: this allows us to compensate for possible synchronization inaccuracies. The actual length of t_{jam} must therefore be increased by $2 \cdot (t_{RST} + t_\epsilon)$ to make sure that t_{samp} is correctly aligned.

V. Predictable Performance under Interference

We mentioned in Sect. IV-C that one can use t_{jam} to provide probabilistic guarantees on the fraction of disagreements. When setting $t_{jam} > t_{busy}^{max}$ is not possible, it is important to precisely calibrate t_{jam} so that a user with stringent real-time constraints can know in advance the fraction of disagreements to be expected. Hence, we now derive a probabilistic model that bounds the probabilities of positive agreements and disagreements for JAG, given a certain value of t_{jam}.

Variable	Description
t_{pkt}	Transmission delay of PKT containing V
t_{ack}	Transmission delay of ACK
t_{jam}	Duration of jamming signal in JAG
X	Random variable denoting the length of the idle period
$p(x)$	Probability density function (*pdf*) of X

TABLE I
Notation used in our probabilistic model.

The parametrization of the probabilistic model requires the user to run a wireless sniffer in order to capture the characteristics of the surrounding interference. We use continuous RF noise measurements to measure the duration of idle and busy periods and compute their probability density function (*pdf*): a channel is defined as busy if the RSSI is higher or equal than a configurable threshold R_{thr} and idle otherwise.

Preferably, this operation should be carried out before the actual deployment, but it would also be possible to characterize interference at runtime, for example in case the RF environment has changed significantly from the prior observation.

The user can then follow three simple steps: (i) compute the *pdf* of the idle periods $p(i)$, where i represents the length of the idle period, (ii) compute the conditional *pdf* of the busy periods following the idle periods $p(b > x|i)$, and (iii) use the model to obtain the value of t_{jam} that provides the desired QoS.

Table I shows the notation used in our analysis. Our goal is to derive the probabilities of positive agreements and disagreements for JAG given a certain value of t_{jam}. First, we obtain the probability of selecting an idle period of length i, then, we derive the probabilities of obtaining positive agreements and disagreements over all possible idle periods.

Denoting $p(i)$ as the probability density function of the idle periods formed by the interference pattern, the probability of selecting an idle period of length i is given by:

$$s(i) = \frac{ip(i)}{\sum_{i=1}^{\infty} ip(i)} \tag{1}$$

i.e., the more frequent and the longer the idle period, the higher the likelihood of selecting it.

In order to derive the required probabilities, we need to understand the interplay between the length of an idle period i and the 3-way handshake method used by JAG (i.e., the transmission of the PKT embedding V, the ACK, and the JAM signal). In principle, based on the definitions presented in Sect. II, losing an ACK should lead to negative agreements. The practical implementation of JAG, however, takes an optimistic approach that increases the likelihood of positive agreements at the cost of turning some negative agreements into disagreements. In JAG, if $\mathcal{R}$ sends the ACK, four outcomes can occur: (i) a positive agreement, if the ACK is successfully delivered to $\mathcal{S}$ and the JAM signal is correctly decoded by $\mathcal{R}$; (ii) a negative agreement, if the ACK is lost and $\mathcal{R}$ detects the *lack of* JAM; (iii) *another positive agreement*, independently of the fact that the ACK is received or not if, after sending

the ACK, $\mathcal{R}$ detects an interference signal with a strength higher than the expected JAM signal and hence assumes a successful transaction (this is the optimistic approach, which assumes the JAM was buried within the stronger signal); and (iv) a disagreement, if the ACK is lost, but, by chance, a high interference signal lasts longer than t_{samp}. In this case, $\mathcal{R}$ assumes, mistakenly, a successful exchange, i.e., a negative agreement turns into a disagreement.

Based on the above description, in JAG, positive agreements are given by the following equation:

$$P_{\text{jam}}\{\text{Pos. Agr.}\} = \sum_{i>t_{\text{pkt}}+t_{\text{ack}}}^{\infty} s(i)(1 - \frac{t_{\text{pkt}} + t_{\text{ack}}}{i}) \quad (2)$$

whereby the first term of the product states the probability of obtaining an idle slot of length i, and the second term states the probability that the selected idle slot can "contain" the transmission of the packet followed by the ACK ($t_{\text{pkt}} + t_{\text{ack}}$).

In order to obtain the fraction of disagreements, we use a bounding probability. There are three necessary but not sufficient conditions to obtain disagreements: (i) PKT is transmitted successfully, (ii) the ACK is corrupted and (iii) the interference signal after the ACK is longer than t_{jam} (to shadow the JAM signal). Hence, we define the probability of obtaining disagreements with JAG as follows:

$$\begin{aligned} P_{\text{jam}}\{\text{Disagreement}\} \leq & \sum_{i=1}^{t_{ack}} s(i)p(b > t_{jam}|i) + \\ & \sum_{i>t_{ack}}^{t_{pkt}+t_{ack}} s(i)p(b > t_{jam}|i)(1 - \frac{\min(t_{pkt}, i)}{i}) + \\ & \sum_{i>t_{pkt}+t_{ack}}^{\infty} s(i)p(b > t_{jam}|i)(\frac{t_{ack}}{i}) \end{aligned} \quad (3)$$

Each of the sums on the right side of the equation has three terms. The first term $s(i)$ denotes the probability of obtaining an idle slot of length i. The second term $p(b > t_{jam}|i)$ denotes the probability of obtaining a busy period b longer than t_{jam} after an idle period of length i (the minimum requirement to shadow the jamming signal). The third term differs for each sum, and denotes the probability that the ACK will be corrupted: in the first summation this probability is 1, because the idle time is less than t_{ack}, i.e., the ACK will always be corrupted; in the second and third summations, this probability describes the chances that the agreement starts early enough to allow a successful delivery of PKT, but late enough to corrupt the ACK. Please note that, in Eq. 3, the term $p(b > t_{jam}|i)$ assumes that the corrupted ACK ends exactly before the next busy period starts. In practice, the ACK will likely have a Δ overlap with the beginning of the busy period b, and hence, b will need to be longer than $(t_{jam} + \Delta)$ to lead to a disagreement. Given that $p(b > t_{jam}|i) > p(b > (t_{jam} + \Delta)|i)$, in practice, we can expect a lower fraction of disagreements.

For the case of disagreements, JAG allows the user to fine-tune the duration of t_{jam} according to the requirements of the application (Eq. 3). In Sect. VI-E, we will observe that this fine-tuning capability is central to provide QoS guarantees.

VI. Experimental Evaluation

A. Experimental Setup

We carry out our experiments in two small-scale sensornet testbeds with USB-powered sensor nodes. The first testbed consists of 15 MTM-CM5000MSP nodes deployed in an office environment, whereas the second testbed uses the same type of sensor nodes deployed in a residential building. We use our first testbed to evaluate the performance of several agreement protocols under different types of interference. To this end, we use JamLab [6], a tool for controlled and realistic interference generation in specific IEEE 802.15.4 channels. We configure JamLab to emulate a continuous file transfer produced by either Bluetooth or Wi-Fi devices in specific IEEE 802.15.4 channels. We further carry out experiments in the presence of a Wi-Fi interference generated by a laptop continuously downloading a file from a nearby access point. We validate our first set of results using a second testbed deployed in residential buildings surrounded by Wi-Fi stations: we run different agreement protocols for several days and compare their performance over time.

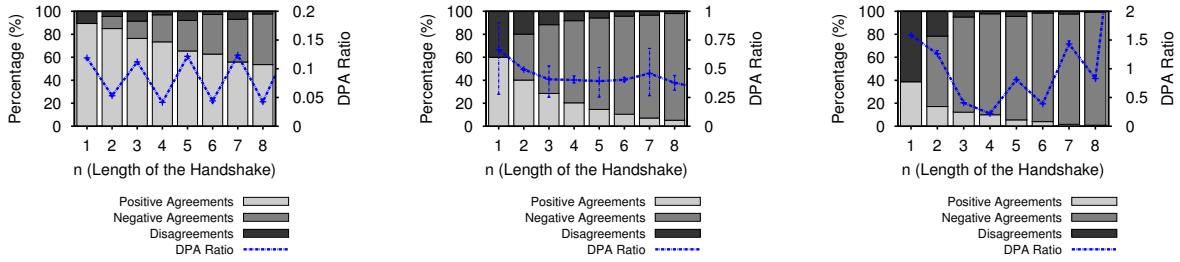
In our experiments, we use several pairs of nodes $\mathcal{S}$ and $\mathcal{R}$. Node $\mathcal{S}$ always initiates the handshake, and transmits a data packet composed of a 6-byte payload containing the information to be agreed upon V and the transmission power used T_P. For each handshake (which is initiated after a random interval in the order of hundreds of milliseconds), we select a random transmission power between -25 dBm and 0 dBm in order to create different types of links. $\mathcal{R}$ replies to the packet using T_P, i.e., the same transmission power used by $\mathcal{S}$. Hardware ACKs are enabled by default, and nodes remain on the same channel during the whole duration of the experiment, in which we perform several hundred thousand handshakes.

B. Packet-based n-way handshake

We firstly analyse the performance of the packet-based n-way handshake shown in Fig. 1 (redundancy factor $k = 1$) under different interference patterns. In our implementation, every packet from $\mathcal{R}$ to $\mathcal{S}$ is sent using the hardware ACK support, so to minimize the latency between the reception of the previous packet and the dispatch of the following one.

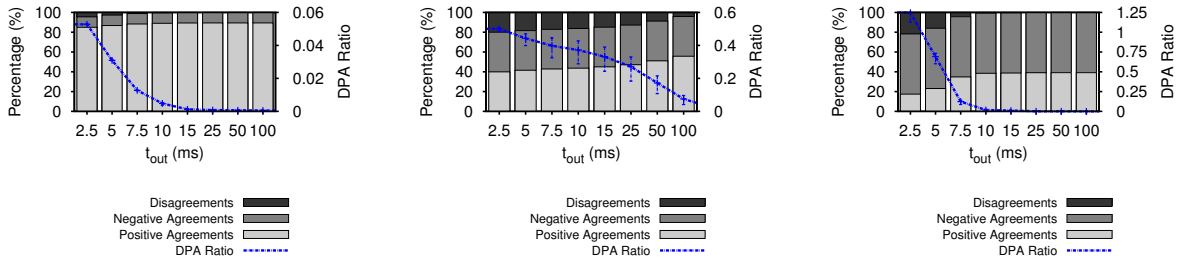
Fig. 9 shows the percentage of positive/negative agreements and disagreements obtained under different interference patterns. The values are computed as an average over all transmission power values T_P used in our experiments, excluding the ones leading to asymmetric links.

Fig. 9(a) depicts the performance of the protocol under JamLab's emulated Bluetooth file transfer. As discussed in Sect. II, the longer the handshake, the smaller the amount of disagreements and positive agreements. Hence, the DPA ratio does not decrease when increasing the length of the handshake n. The alternating performance of the DPA ratio is caused by the interchange between software and hardware ACKs: the former require a higher latency to be transmitted, and hence offer a worse performance with respect to the latter. Fig. 9(b) and 9(c) show the performance of the n-way handshake protocol under JamLab's emulated Wi-Fi transfer



(a) Bluetooth (b) Real Wi-Fi (c) Emulated Wi-Fi

Fig. 9. Performance of a packet-based n-way handshake under different types of interference.



(a) Bluetooth (b) Real Wi-Fi (c) Emulated Wi-Fi

Fig. 10. Performance of 2-MAG (2-way handshake in which the last acknowledgement packet is sent k times) under different types of interference. The longer t_{out}, the lower the amount of disagreements in favour of positive agreements, at a price of an increased energy consumption.

and under Wi-Fi interference generated by a continuously active laptop, respectively. As the interference becomes heavier, the amount of positive agreements and the amount of disagreements drastically decrease after few iterations, hence the DPA ratio does not improve significantly. Our experiments therefore confirm our observations in Sect. II: packet-based n-way handshakes are not optimal under external interference.

C. 2-MAG: 2-way handshake enhanced with redundancy

To minimize the DPA ratio, we introduce redundancy of the last ACK packet as discussed in Sect. II-B, and we analyse the performance of a 2-way handshake in which the last ACK packet is sent k times, as illustrated in Fig. 2. For simplicity, in the remainder of this paper we will refer to this protocol as 2-MAG (2-way handshake Message-based AGreement).

Given the structure of JAG, a more fair comparison would involve a 3-way handshake message-based agreement protocol in which the last packet is sent k times. The choice of a 2-way handshake is driven by the results obtained in Fig. 9: a low n minimizes the probability of negative agreements, and therefore there are higher chances that 2-MAG sustains more positive agreements and outperforms JAG thanks to its redundant transmissions. We make sure to carry out a fair comparison by eliminating asymmetric links that would always lead to disagreements when using a two-way handshake.

In our implementation, hardware ACKs are enabled, i.e., the first ACK packet sent from $\mathcal{R}$ to $\mathcal{S}$ has a short and fixed-delay latency. Every other ACK packet will be generated via software by pre-loading the ACK into the radio buffer and by repeatedly sending its content k times. Please note that the preparation of the software ACK is time-critical, as one need to extract and analyse V before creating the ACK and loading it into the radio buffer.

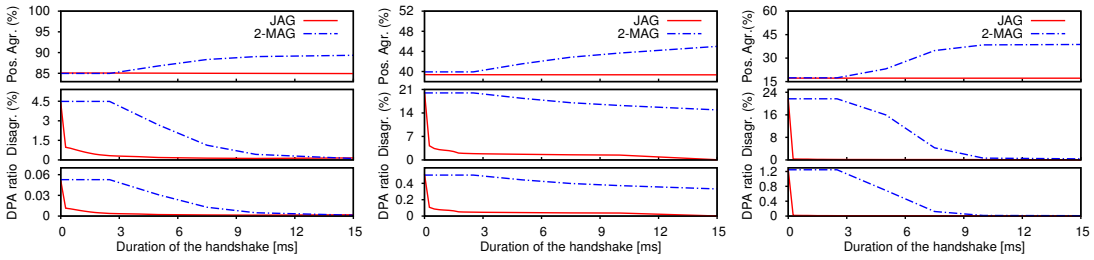
In order for $\mathcal{S}$ to consider V as successfully exchanged, it is sufficient to receive one ACK packet within a maximum waiting time t_{out}. Clearly, the longer t_{out}, the higher the likelihood that at least one ACK packet will be correctly decoded and the better 2-MAG will perform (at the price of an increased energy consumption). Hence, we compute t_{out} as the maximum time in which node $\mathcal{S}$ waits for a valid ACK packet from $\mathcal{R}$.

Fig. 10 shows the percentage of positive and negative agreements as well as disagreements obtained in the presence of interference using 2-MAG as a function of t_{out}. As expected, the longer t_{out}, the lower the amount of disagreements in favour of positive agreements. As this minimizes the DPA ratio, 2-MAG outperforms a generic n-way handshake without redundancy in the presence of external interference.

D. JAG: Jamming-based AGreement

We now evaluate the performance of JAG and compare it against 2-MAG. In particular, we are interested in comparing how the percentage of positive/negative agreements and disagreement change when we increase the duration of the handshake. Intuitively, the longer t_{out} for 2-MAG and the longer t_{jam} for JAG, the better the performance. However, it is important to see their distribution to study the protocols' energy-efficiency and their DPA ratio under interference.

Fig. 11 shows the results: JAG sustains a significantly lower amount of disagreements compared to 2-MAG already for small values of t_{jam}. For example, 2-MAG requires more than 7.5 ms to obtain less than 1% disagreement under Bluetooth interference, whereas JAG achieves this amount with a $t_{jam} \leq 250\mu$s.



(a) Bluetooth (b) Real Wi-Fi (c) Emulated Wi-Fi

Fig. 11. Compared to the 2-way handshake in which the last acknowledgment packet is sent k times, JAG performs better independent of the interfering source, as it reduces the duration of the handshake required to minimize the amount of disagreements.

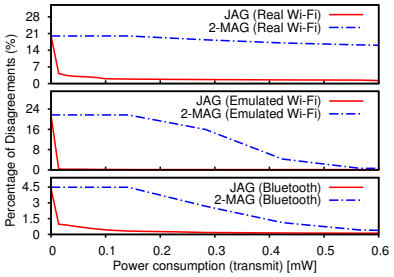
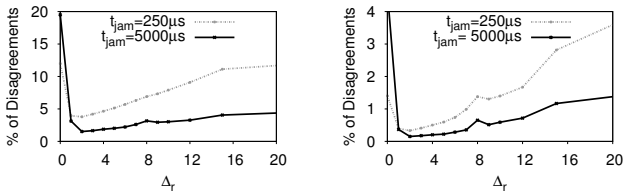


Fig. 12. Disagreements as function of energy for JAG and 2-MAG.



(a) Real Wi-Fi (b) Heavy Wi-Fi

Fig. 13. Role of Δ_r on the probability of disagreement.

Even though 2-MAG has a high number of positive agreements, it requires significantly higher values of t_{out} to reduce the amount of disagreements and the DPA ratio. JAG, instead, has a very low rate of disagreements under every type of interference even with small t_{jam}, which enables significant energy savings, as shown in Fig. 12. Furthermore, when t_{jam} is longer than the longest interference burst, we do not have any disagreements as discussed in Section IV-C. Obtaining this behaviour using packet-based approaches would require a significantly higher cost: Fig. 10(b) shows that even when sending bursts of ACKs for 100 ms, one cannot still guarantee the absence of disagreements. Hence, compared to packet-based approaches, JAG performs better and guarantees agreement with less costs and with weaker and more realistic assumptions about the underlying interference pattern.

Fig. 11(c) shows that the rate of disagreements obtained in the presence of emulated Wi-Fi interference tends to zero faster than the one obtained in the presence of real Wi-Fi interference. This is because the interference generated by JamLab contains fast transmissions with short idle and busy periods. Therefore, JAG has high chances to detect an idle period already when using a short t_{jam}.

In addition to t_{jam}, another parameter to be configured in JAG is Δ_r, which helps in compensating changes between r_s and the strength of the received jamming signal. Δ_r should be selected not too small (so to account for the inaccuracy of the RSSI readings), but at the same time not too large, as this would neutralize the benefits of having knowledge of r_s. Fig. 13 depicts the percentage of disagreements as a function of Δ_r: a value of 3 dBm offers a good trade-off.

Finally, we validate the goodness of JAG by running a long-term experiment in our second testbed deployed in a residential environment. In particular, we compare the performance of JAG and 2-MAG over time when using $t_{jam} = 500\mu$s for JAG and $t_{out} = 5ms$ for 2-MAG (Fig. 14). We do not change the

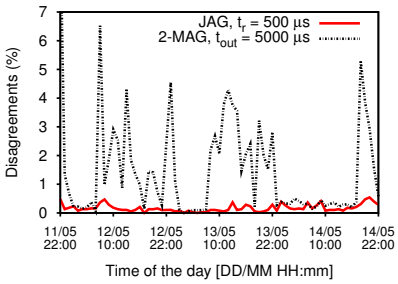
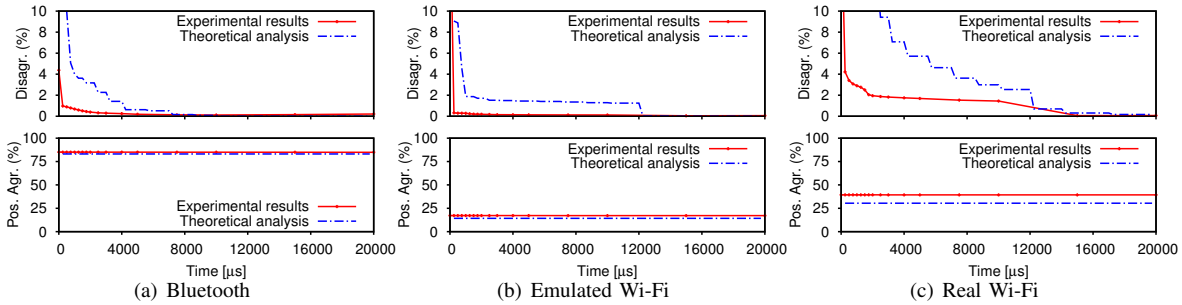


Fig. 14. Long-term experiment in a residential environment.

configuration of the two protocols throughout the duration of the experiment. The interference in the environment changes significantly over the day: a lot of Wi-Fi activity was present during daytime in the weekend (May, 12-13), but it was quiet during night and on Monday (May, 14) during the day, as most people were not in their homes. Despite selecting a t_{out} 10 times higher than t_{jam}, JAG sustains a significantly lower amount of disagreements and outperforms 2-MAG during the whole duration of the experiment.

E. Predictability of JAG

We now evaluate the goodness of the probabilistic model presented in Sect. V with respect to the predictability of the performance of JAG. In order to do this, we firstly obtain the *pdf* of idle and busy periods using sensor nodes in wireless sniffer mode in the scenarios described in the previous sections, i.e., in the presence of JamLab's emulated interference and real Wi-Fi interference generated by a laptop (the *pdf*s in the presence of real Wi-Fi interference are shown in Fig. 5). Then, based on equation (2) and (3), we obtain an upper bound for the probability of obtaining disagreement and a lower bound for the probability of obtaining positive agreements as a function of t_{jam} using t_{pkt} = 1 ms, t_{ack} = 750 μs, and t = -90 dBm.



(a) Bluetooth (b) Emulated Wi-Fi (c) Real Wi-Fi

Fig. 15. Comparison of the rate of positive agreement and disagreement obtained running JAG on real wireless sensor nodes, and deriving the probabilities using the analytical model shown in Sect. V. The model actually returns a lower bound for positive agreements and an upper bound for disagreements.

By running JAG on real wireless sensor nodes, we verify experimentally whether the probabilistic model is able to predict the performance of JAG. The results illustrated in Fig. 15 show that our probabilistic model parametrizes correctly t_{jam} by giving an upper bound on the amount of disagreements and a lower bound on the amount of positive agreements, hence predicting the performance of the protocol correctly. Note that the probabilistic model was computed for every possible t_{jam}, whereas due to memory limitations of real nodes only a finite amount of t_{jam} were computed experimentally. Please note that Fig. 15 shows a different performance between emulated and real interference: whilst JamLab is designed to attain repeatability and test algorithms under the same conditions, real-world settings have several variables affecting their dynamics.

Based on our results, we can conclude that our theoretical model is indeed able to parametrize JAG and predict correctly the maximum amount of disagreements occurring for a given t_{jam}. This can be useful when the latter is shorter than the longest busy period created by interference (t_{busy}^{max}).

VII. Integration of JAG into MAC Protocols

As previously discussed, JAG is intended as a building block to construct protocols at different layers of the protocol stack. For example, it could be embedded into a MAC protocol to agree on the TDMA schedule or the next frequency channel. We now discuss how JAG can be integrated in existing MAC protocols to enhance their performance.

As many deployments gather environmental data and send them to a number of sinks, several convergecast MAC protocols have been proposed in sensor networks, such as Chrysso [16] and CoReDac [17]. In these protocols, nodes are logically organized into parent-children groups that may operate on different channels. In Chrysso [16], individual parent-children pairs collaboratively switch their communication channel as soon as performance degrades. In particular, a parent node monitors the average back-off time, and as soon as it exceeds a given threshold, it instructs all its children to carry out a channel switch by piggybacking the "switch-channel command" onto ACK messages, and then switches to the next channel. This operation is carried out for each parent-child pair individually, and can be considered a two-way handshake between child and parent (2-MAG) in which the information V to be agreed upon is contained in the second message. Please note that, on a high-level basis, V does not have to be necessarily included in the first message of the exchange: in a n-way handshake, V is in any case only used once the last message has been received, so it can be embedded in any of the messages exchanged in the handshake. The only difference with respect to 2-MAG is that, when piggybacking an information V into an ACK message, the latter cannot be sent as a hardware ACK as it contains extra-information.

JAG can be embedded into Chrysso by replacing the 2-way handshake between child and parent with a 3-way handshake in which the child sends an initial packet P, the parent answers with a software ACK containing the new channel to be used (V), and the child confirms the reception of V by jamming for a predefined amount of time t_{jam}. The parent node deems the exchange as successful (jamming sequence detected) or unsuccessful (jamming sequence not detected) depending on the results of a fast RSSI sampling, as described in Sect. IV-A.

The same principle can be used to enhance the performance of CoReDac [17], a TDMA-based convergecast protocol in which parent nodes split their reception slots into subslots, and assign one slot to each child in order to build a collection tree that guarantees collision-free radio traffic. As in Chrysso, also in CoReDac the assignment information used for synchronizing the TDMA-schedules is piggybacked onto ACK messages, and one can introduce a three-way handshake using JAG in the same way as described above. However, in the current version of CoReDac, there is a single aggregated ACK message containing the identifier of all children: this can be easily changed to individual ACKs to each child without affecting the overall protocol architecture.

The use of a 3-way handshake requires additional energy compared to the traditional message-based 2-way handshake implemented by Chrysso and CoReDac. However, this may pay off in the presence of interference, as it would increase the chances of agreement. As we have shown in our previous work [18], CoReDac performs poorly in the presence of interference, since when an ACK is lost, a sensor node needs to keep its radio on until it hears a new one, and integrating JAG may lead to substantial performance improvements.

VIII. Related Work

Agreement is a well-known problem in distributed systems. Pioneering work in the late 1970s highlighted the design challenges when attempting to coordinate an action by communicating over a faulty channel [2], [4].

In the context of wireless sensor networks, the agreement problem has not been widely addressed. The main focus has been on security for the exchange of cryptographic keys [19], and on average consensus for nodes to agree on a common global value after some iterations [20]. Similarly to these studies, our work aims at protocols that allow a set of nodes to agree on a piece of information. In addition, we also tackle agreement under interference and provide a lightweight energy-efficient solution that fits applications with strict performance requirements.

Our work is motivated by studies reporting the degrading QoS caused by the overcrowding of the RF spectrum in unlicensed bands [3]. Several solutions have been proposed: Chowdhury and Akyildiz identify the type of interferer and schedule transmissions accordingly [21]. Liang et al. increase the resilience of packets challenged by Wi-Fi interference using multi-headers and FEC techniques [11]. Other protocols, such as Chrysso and ARCH, dynamically switch the communication frequency as soon as interference is detected [22], [16]. As these protocols rely on packet exchanges to coordinate the channel switching, one can use JAG to improve their performance, as discussed in Sect. VII.

Another set of studies propose to cope with interference by exploiting its idle or busy periods. Noda et al. have proposed a channel quality metric based on the availability of the channel over time, which quantifies spectrum usage [23]. Hauer et al. report the interference observed by a mobile body area network in public spaces, and the study shows the intermittent interference caused by Wi-Fi AP in all IEEE 802.15.4 channels [7]. Similarly, Huang et al. have shown that Wi-Fi traffic inherently leaves "a significant amount of white spaces" between 802.11 frames [24]. BurstProbe uses a probing mechanism to periodically measure burst error patterns of all links used in the deployment and, whenever the interference patterns leave predicted bounds, a warning is issued so that one can reconfigure the deployed network [25]. Similarly to these studies, JAG exploits idle times for data packets, but also leverages the bursty nature of interfering sources to achieve reliable agreements through the use of jamming signals.

IX. Conclusions

In this paper, we propose JAG, a simple and efficient agreement protocol for wireless sensor networks exposed to external interference. JAG introduces a novel technique that utilizes jamming signals to acknowledge the reception of a packet. Our results show that JAG outperforms traditional methods using packet-based acknowledgements. Further, JAG provides predictable performance in that it keeps, within a specified energy budget and delay time, the probability of disagreements below a pre-defined threshold even in the presence of external interference, and in that it can be configured to always reach agreement (positive or negative) in a finite amount of time.

A limitation of the current version of JAG is that jamming sequences do not provide identity information, and hence may be generated by a malicious device. JAG partially solves the problem by using a mechanism to verify that the strength of the jamming signal equals the one that would be produced by the device of interest. However, security is an important concern nowadays, and it would be important to unequivocally guarantee the identity of the jamming node by means of authentication. We will address this issue in future work.

Acknowledgments

This work has been partially supported by the European Commission with contracts FP7-2007-2-224053 (CONET, the Cooperating Objects Network of Excellence), and INFSO-ICT-317826 (RELYonIT). This research has been also partially financed by VINNOVA, the Swedish Agency for Innovation Systems, by the DFG-funded Cluster of Excellence 306/1 "Inflammation at Interfaces", and by the Mercatur Research Center Ruhr Grant number Pr-2011-0014 (SEVERE).

References

[1] T. Abdelzaher et al., "EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks," in *24th ICDCS*, 2004.
[2] E. Akkoyunlu et al., "Some Constraints and Tradeoffs in the Design of Network Communications," in *Symp. on Op. Syst. Princ. (SOSP)*, 1975.
[3] G. Zhou, J. Stankovic, and S. Son, "Crowded Spectrum in Wireless Sensor Networks," in *3rd Worksh. on Emb. Net. Sens. EmNetS*, 2006.
[4] J. Gray, "Notes on Data Base Operating Systems," in *Operating Systems, an Advanced Course, pp. 393–481*, 1978.
[5] R. Musaloiu-E. and A. Terzis, "Minimising the Effect of WiFi Interference in 802.15.4 WSN," *IJSNet*, vol. 3, no. 1, pp. 43–54, Dec. 2007.
[6] C.A. Boano, T. Voigt, C. Noda, K. Römer, and M. Zúñiga, "JamLab: Augmenting Sensornet Testbeds with Realistic and Controlled Interference Generation," in *10th ACM/IEEE IPSN*, 2011.
[7] J. Hauer, V. Handziski, and A. Wolisz, "Experimental Study of the Impact of WLAN Interf. on IEEE 802.15.4 BANs," in *6th EWSN*, 2009.
[8] J. Hauer, A. Willig, and A. Wolisz, "Mitigating the Effects of RF Interference through RSSI-Based Error Recovery," in *7th EWSN*, 2010.
[9] A. Kamerman and N. Erkocevic, "Microwave Oven Interf. on Wireless LANs Operating in the 2.4 GHz ISM Band," in *IEEE PIRMC'97*.
[10] W.-B. Pöttner, S. Schildt, D. Meyer, and L. Wolf, "Piggy-Backing Link Quality Measurements to IEEE 802.15.4 ACKs," in *8th MASS*, 2011.
[11] C. Liang, N. B. Priyantha, J. Liu, and A. Terzis, "Surviving Wi-Fi Interf. in Low Power ZigBee Networks," in *8th ACM SenSys*, 2010.
[12] K. Jamieson and H. Balakrishnan, "PPR: Partial Packet Recovery for Wireless Networks," in *ACM SIGCOMM*, 2007.
[13] C.A. Boano, Z. He, Y. Li, T. Voigt, M. Zuniga, and A. Willig, "Controllable Radio Interference for Experimental and Testing Purposes in Wireless Sensor Networks," in *4th IEEE SenseApp*, 2009.
[14] J. Polastre, J. Hill, and D. Culler, "Versatile Low-Power Media Access for Wireless Sensor Networks," in *2nd ACM SenSys*, 2004.
[15] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki: a Lightweight and Flexible Op. System for Tiny Networked Sensors," in *1st EmNetS*, 2004.
[16] V. Iyer, M. Woehrle, and K. Langendoen, "Chrysso: A Multi-channel Approach to Mitigate External Interference," in *8th IEEE SECON*, 2011.
[17] T. Voigt and F. Österlind, "CoReDac: Collision-Free Command-Response Data Collection," in *13th IEEE ETFA*, 2008.
[18] C.A. Boano, T. Voigt, N. Tsiftes, L. Mottola, K. Römer, and M.A. Zúñiga, "Making Sensornet MAC Protocols Robust Against Interference," in *7th EWSN*, 2010.
[19] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney, "A Key Management Scheme for WSN using Depl. Knowledge," in *23rd INFOCOM*, 2004.
[20] L. Xiao, S. Boyd, and S. Lall, "A Scheme for Robust Distributed Sensor Fusion based on Average Consensus," in *4th ACM/IEEE IPSN*, 2005.
[21] K. Chowdhury and I. Akyildiz, "Interferer Classification, Channel Selection and Transmission Adaptation for WSN," in *IEEE ICC*, 2009.
[22] M. Sha, G. Hackmann, and C. Lu, "ARCH: Practical Channel Hopping for Reliable Home-Area Sensor Networks," in *17th IEEE RTAS*, 2011.
[23] C. Noda et al., "Quantifying the Channel Quality for Interference-Aware Wireless Sensor Networks," *ACM SIGBED Review*, vol. 8, no. 4, 2011.
[24] G. Huang et al., "Beyond Co-Existence: Exploiting WiFi White Space for Zigbee Performance Assurance," in *18th IEEE ICNP*, 2010.
[25] J. Brown et al., "BurstProbe: Debugging Time-Critical Data Delivery in Wireless Sensor Networks," in *8th EWSN*, 2011.