Evaluating Multiple-Access Protocols: Asynchronous Pulse Coding vs. Carrier-Sense with Collision Avoidance

Kenji Leibnitz[1(✉)], Ferdinand Peper[1], Konstantinos Theofilis[1], Mikio Hasegawa[2], and Naoki Wakamiya[3]

[1] National Institute of Information and Communications Technology, Osaka, Japan
{leibnitz,peper,kostas}@nict.go.jp
[2] Tokyo University of Science, Tokyo, Japan
hasegawa@ee.kagu.tus.ac.jp
[3] Osaka University, Osaka, Japan
wakamiya@ist.osaka-u.ac.jp

Abstract. While the Internet of Things (IoT) is usually envisioned to support powerful functionality, like in self-driving cars, there is also increasing interest in simpler IoT applications that can be employed on massive scales at high densities, like in data gathering at meetings with large audiences. The latter vision requires low-cost devices consuming little energy, and it tends to come with a relaxed need for high-speed communication. Its realization necessitates the development of wireless protocols that are simple, yet that can effectively arbitrate multi-access to communication channels. *Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA)* is usually deployed in such contexts, but it tends to work less well when large numbers of nodes attempt to simultaneously access a wireless channel. *Asynchronous Pulse Code Multiple Access (APCMA)* has been developed with simultaneous asynchronous access to communication channels in mind by using a sparse representation of pulses to encode messages, but being relatively recent, its performance has never been systematically compared to CSMA/CA. This paper compares APCMA's performance with that of CSMA/CA in terms of the success probability (i.e., absence of errors) of message transmissions through the use of simulations and analytical models, under the assumption of equivalence in throughput. We find that APCMA with four pulses per code word performs worse than CSMA/CA, but the roles are reversed if five or six pulses are used.

Keywords: Internet of Things · Pulse-based encoding · Carrier-Sense Multiple Access

1 Introduction

Wireless sensor networks with cheap, small, and energy-efficient nodes will play a major role in the realization of ubiquitous environments and massive IoT applications. Various standards, like IEEE 802.15.4, have been developed that minimize the complexity

This research and development work is supported by MIC/SCOPE no. JP205007001.

T. Hara and H. Yamaguchi (Eds.): MobiQuitous 2021, LNICST 419, pp. 693–706, 2022.
https://doi.org/10.1007/978-3-030-94822-1_45

of nodes and their energy consumption, while allowing reasonable (but not necessarily high) data rates. Protocols in this standard typically require synchronization between senders and receivers, as well as contention resolution mechanisms to provide access to shared wireless channels in a fair way. Though various multiple access mechanisms have been developed, like *Code-Division Multiple Access (CDMA)*, in which multiple senders are able to transmit simultaneously on the same channel(s), they tend to require quite complex algorithms for encoding and decoding. Alternatively, contention resolution has been implemented based on *ALOHA* [6] and *Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA)* [3,8]. ALOHA was among the first in this context, and it is easy to implement, but it tends to perform worse than CSMA/CA. However, CSMA/CA comes with its own disadvantages: fundamentally, it allows only a single sender to transmit on each band at a time, and it requires scheduling mechanisms to determine back-off times, clear channel assessment, and transmission initiation. Furthermore, it scales poorly: when the number of senders increases, the rate of successful transmissions drops significantly, even if many retrials are allowed.

These issues are addressed by a protocol that is proposed in [9], the so-called *Asynchronous Pulse Code Multiple Access (APCMA)*. In APCMA, information is encoded as the time intervals between successive pulses. It is based on the same principle as *Communication through Silence (CtS)* [17]. In CtS the intervals between pulses need to be silent, i.e., not interrupted by pulses from competing transmissions, and if this condition is not met, corruption of data occurs. In other words, CtS is not robust to collisions of messages from different nodes, which is the reason why it has been combined with multiple access mechanisms like CSMA/CA in [1,2]. APCMA, on the other hand, is designed to cope with message collisions by adding redundant pulses to each code word such that the unique pattern of each code word can be picked up by a decoder at a receiver, even if the pulses in messages are interspersed by pulses from other messages. Collisions are much less of a problem in APCMA due to a number of factors: (1) Each code word contains only a small number of pulses, i.e., code words are sparse; (2) Collisions of pulses are no problem, because they do not disturb the underlying patterns of pulses in code words; (3) Even if a collision occurs, it can be successfully resolved with a high probability through the use of a pattern recognition algorithm that recognizes certain combinations of pulses as valid code words and rejects all others. In other words, APCMA allows multiple access, in which multiple messages can be transmitted by different senders at the same time on the same channel without there being a need to reschedule in case collisions occur. Another strong point of APCMA is that synchronization between senders and receivers is not required: any sender can initiate its transmission at any time without needing to check whether other transmissions are ongoing. In its current state, however, APCMA has a drawback in that it uses unary encoding. This makes code words quite long, thus increasing delay of messages. On the other hand, it also increases the sparsity of code words, which gives it the potential to allow thousands of devices to transmit at the same time [10,11,14,15].

Since CSMA/CA is used in many protocols, it brings up the question of how it compares to APCMA. On one hand, packets in CSMA/CA-based protocols tend to be much shorter than packets in APCMA, but on the other hand there is a much higher chance of collisions in CSMA/CA if many senders are active at the same time. Errors in

CSMA/CA are defined differently than in APCMA: while in CSMA/CA an error is considered to have occurred either if a sender has failed to successfully transmit a message to a receiver after a certain limit on the number of retransmissions has been reached or if a transmission has collided with other transmissions rendering it unintelligible, in APCMA an error is considered to have occurred if a receiver has failed to decode a message unambiguously. While the nature of the errors is different for both protocols—in CSMA/CA the message has not been received at all, and in APCMA the message is hidden in a multitude of interpretations—in practice their outcomes are similar, i.e., a message will fail to reach its destination.

Comparing the throughput in both protocols is somewhat complicated. While in CSMA/CA throughput depends on the number of back-offs required to succeed in a transmission, as well as the average sleep interval between transmissions, in APCMA the throughput depends on the used pulse code, especially its sparsity, as well as on the average sleep interval between transmissions. In practice, however, there is little difference, since in the end it is about the number of bits per seconds that are transmitted by each node. A fair comparison between CSMA/CA and APCMA can be obtained when an equivalent throughput is assumed, after which the rates of successfully-received messages are measured.

This paper compares CSMA/CA with APCMA by computer simulations and analytical models. We use a model of CSMA/CA that is inspired by the IEEE 802.15.4 standard [5], but that is adjusted to make the comparison with APCMA easier. Parameters of the models are set such that throughput is similar, and based on these settings the rates at which transmitted messages successfully arrive at a receiver are investigated. The CSMA/CA model is investigated through simulation, as is the APCMA model for code words with four and five pulses each. Results for APCMA based on the analytical model in [10] are also added to the comparison, whereby four, five, and six pulses per code word are used. We investigate scenarios with numbers of nodes varying between one to up to 2500 transmitters, whereas all messages are received at one single receiver in a single-hop fashion. Three different scenarios corresponding to different throughputs are investigated, whereby the throughput in a scenario is reduced by increasing the sleeping periods between transmissions. In general, the success rate decreases with the increase in number of nodes for all models, as can be expected. When the sleep intervals between transmissions are increased in length, the success rate increases in all scenarios, which is also to be expected, since an increase in sleep will reduce the occupancy of a channel. In all scenarios the 4-pulse APCMA model performs worse than CSMA/CA in simulations. The 5-pulse and 6-pulse APCMA models, on the other hand, perform better than CSMA/CA in all scenarios where the success rate is high enough to be of practical value. The analytical APCMA model gives very similar performance as its simulated counterpart for four and five pulses, which suggests that the 6-pulse version of APCMA would follow its analytical model closely if it were to be simulated. All comparisons of performance are done with respect to the data link layer of the OSI model, and we assume perfect fidelity of the physical layer. Our justification is that only simple modulations of signals are used on which the physical layers are equally affected by noise for all models.

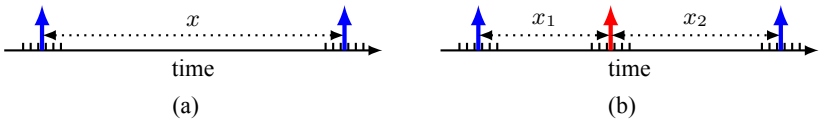

Fig. 1. (a) Length of the silent interval between two successive pulses (indicated by thick blue arrows) encodes the value x in the CtS scheme. (b) When there is an interspersed pulse from an unrelated message (indicated by the red arrow), then the interval is broken in two, giving two erroneous values x_1 and x_2. (Color figure online)

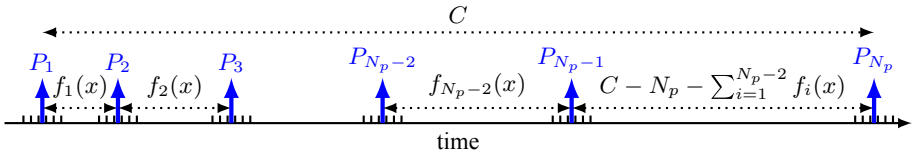

Fig. 2. Format of a code word in APCMA.

Comparison of CSMA/CA with APCMA is important for the realization of an Internet of Things (IoT) in which nodes are extremely simple and have low cost. Such nodes can be expected in ubiquitous environments, as well as in applications like asset tracking or monitoring of anomalous health signs (like high body temperature) at mass events [4,7].

This paper is organized as follows. Section 2 describes APCMA in more detail. Section 3 describes the particular form of CSMA/CA we use, whereas Sect. 4 describes the setup of the simulations in more detail and the simulation results. We finish this paper with a discussion and conclusions in Sect. 5.

2 Asynchronous Pulse Code Multiple Access Model

Our starting point is CtS, which encodes a value as the length of a silent time interval between successive pulses, like in Fig. 1(a). An integer x in the range $[0, V]$ is encoded as an interval of $x + 1$ empty time slots lying between two time slots each occupied by a pulse, whereby there is at least one empty time slot between two pulses transmitted from one and the same transmitter.

As can be seen from Fig. 1(b) this value becomes impossible to decode if the interval is interspersed by a pulse from another message. APCMA addresses this problem by adding redundant pulses such that the resulting code words differ as much as possible under all possible shift operations. The general format of a code word in APCMA contains N_p pulses, as illustrated in Fig. 2.

Hereby, the $N_p - 1$ intervals of successive lengths

$$f_1(x),\ f_2(x),\ \ldots,\ f_{N_p-2}(x),\ C - N_p - \sum_{i=1}^{N_p-2} f_i(x), \tag{1}$$

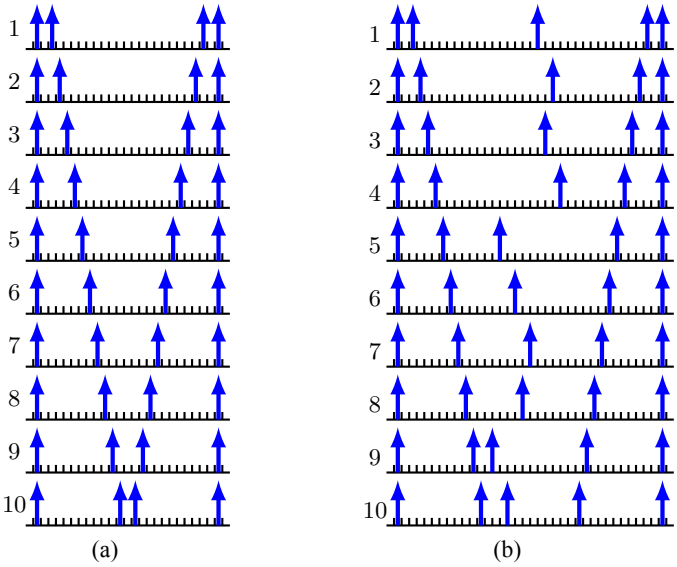

Fig. 3. (a) APCMA code with four pulses and ten code words, with encoding functions $f_1(x) = x + 1$ and $f_2(x) = C - 6 - 2x$ and code length $C = 25$. (b) APCMA code with five pulses and ten code words with code length $C = 36$.

encode the value x. The functions f_i, called the *encoding functions*, are one-to-one, and they are predetermined and shared among the transmitters and receivers, and C is the code length expressed as the number of time slots required for a code word. Examples of 4-pulse and 5-pulse APCMA codes, both with ten code words, are given in Fig. 3.

APCMA codes with five or more pulses tend to be less regular, but the APCMA codes in this paper have in common that their code books contain at most one pulse in each time slot (represented by a column in a code book), except for the first and last pulses, which are always present for all code words. Taking into account these conditions, we assume in this paper that the code length equals $C = (N_p-2)N_c+N_p+1$ time slots, whereby N_p is the number of pulses in each code word and N_c is the number of code words in the code.

We assume a single-hop network in which one node that is continuously listening receives the transmissions from the sender nodes. Sender nodes broadcast their messages according to the schedule in Fig. 4 that consists of cycles, each of which contains one broadcast and one sleep period. A broadcast lasts for C time slots, and it is followed by a sleep interval that is randomly distributed according to a uniform distribution over the interval $\left[S_{\min}^{(a)}, S_{\max}^{(a)}\right]$, whereby $S_{\min}^{(a)}$ is the minimum sleep time and $S_{\max}^{(a)}$ is the maximum sleep time, and the superscript is used to distinguish between APCMA (a) and CSMA/CA (c). $S_{\min}^{(a)}$ is chosen to have value 1 in this paper, and $S_{\max}^{(a)}$ has a value several times C, whereby the multiplier is determined such that a throughput is achieved that is equivalent to that of the CSMA/CA model.

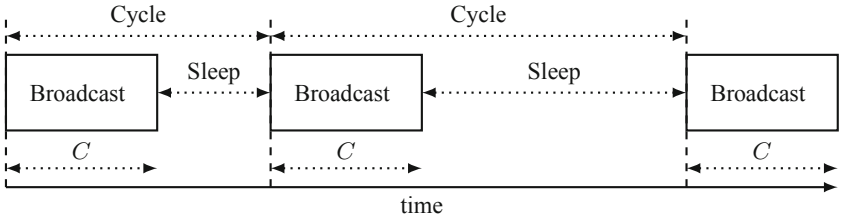

Fig. 4. Scheduling of a node. After each broadcast a node goes to sleep for a random time that is uniformly distributed. Each broadcast is conducted over C time slots, and it starts directly after its preceding sleep period.

It is important that there is sufficient spread in the lengths of the sleeping intervals in simulations of the model and in practical implementations, because this will allow the system to quickly reach a stationary state. Without this spread there will be a kind of phase locking in which waves of above-average numbers of nodes attempting to transmit messages are interspersed by less busy periods. This would significantly reduce the rate by which messages are successfully decoded at the receiver.

The average length of a cycle is now estimated as

$$K_{Cy} = C + \frac{S_{\min}^{(a)} + S_{\max}^{(a)}}{2}. \tag{2}$$

Since a unary code is used for APCMA, the number of bits transmitted in a single message is $\log_2 N_c$. In this paper we use $N_c = 1024$, which corresponds to 10 bits per message. Assuming a duration of a time slot in APCMA of $\Delta t^{(a)}$, we then arrive at a throughput of APCMA of approximately

$$\vartheta^{(a)} = \frac{\log_2 N_c}{K_{Cy}\,\Delta t^{(a)}} \quad \text{bits per second.} \tag{3}$$

This throughput is the raw bit rate that is transmitted by each node individually, without consideration of other nodes. It can be manipulated by varying the code length and the lengths of sleep intervals. Higher bit rates result in higher density of pulses, but they tend to cause decreased probabilities by which messages are successfully decoded at a receiver.

We consider a message to be decoded successfully when that message has been transmitted by a node and the decoder can unambiguously decode it. Using Eq. (8) in [10], we then arrive at a theoretical value of the success probability $p_s^{(a)}$ equaling

$$p_s^{(a)} = \left(1 - p^{N_p - 2}\right)^{N_c - 1}, \tag{4}$$

whereby p is the probability that a time slot is occupied by a pulse in case there are N_n transmitters in the network:

$$p = 1 - (1 - q)^{N_n} \tag{5}$$

Hereby the average density q of pulses over time is given by:

$$q = \frac{N_p}{K_{Cy}} \tag{6}$$

The analytical value for $p_s^{(a)}$ will be used in subsequent sections for comparison with the simulations of APCMA and simulations of CSMA/CA.

The success probability of APCMA in simulations is defined by

$$\widetilde{p}_s^{(a)} = \frac{1}{N_n} \sum_{k=1}^{N_n} \frac{U_k}{M_k}, \tag{7}$$

whereby U_k is the number of messages transmitted by node k that can be uniquely identified, i.e., without ambiguities, and M_k is the total number of messages transmitted by node k.

Decoding of messages at the receiver is conducted by pattern recognition algorithms that use either a type of finite automata operating on pulse sequences [9,11] or a shift register that detects pulse sequences that are input at the left side of the register and that shift one cell to the right each time step [14,15].

APCMA has been implemented on various hardware platforms, including the Arduino Mega 2560 microcontroller [11] and the Xilinx Spartan-3E FPGA [10,14,15]. APCMA has also been implemented and tested for the distribution of power packets in a small electric power network [16]. Such networks require routing of power to, for example, actuators in robots in a flexible way such that the amount of wiring stays limited. The simplicity of APCMA is a major advantage in applications like this. Currently under development are miniaturized nodes based on the Teensy 3.2 microcontroller (Fig. 5).

3 Carrier Sense Multiple Access Model

One of the most commonly found ways of providing multiple access over the wireless channel is by using *Carrier Sense Multiple Access/Collision Avoidance* (CSMA/CA) on the medium access control (MAC) sublayer of the data link layer. Slightly different variations of CSMA/CA exist in multi-access protocols, such as IEEE 802.11 [3] for wireless local area networks (WLAN) or IEEE 802.15.4 [5] for sensor networks and wireless personal area networks (WPAN) that take into account the characteristics of the connected devices and their available power limitations.

Several studies have investigated the dynamics of the backoff algorithm in IEEE 802.15.4 by simulation or theoretical analysis, e.g., [8,12,13]. For this paper, we consider a numerical simulation of CSMA/CA that roughly follows the mechanism in IEEE 802.15.4 to compare its performance with APCMA. In particular, we model the nodes in CSMA/CA to behave similarly as in APCMA by periodically transmitting their sensor data, followed by uniformly distributed sleep periods over the interval $\left[S_{\min}^{(c)}, S_{\max}^{(c)}\right]$. Our considered CSMA/CA model operates without an inactive period within the MAC layer superframe and does not use guaranteed time slots. Since the duration of a unit backoff period depends on the modulation type and frequency band, we make the assumption that a unit backoff period matches exactly the length of a single message ($L = 10$ bits) and the time for a unit backoff period is $\Delta t^{(c)} = 200\,\mu$s, where the superscript (c) stands for CSMA/CA. This corresponds to $20\,\mu$s per bit when *Manchester coding* is used. Since the logical signal associated with Manchester coding is high or low each

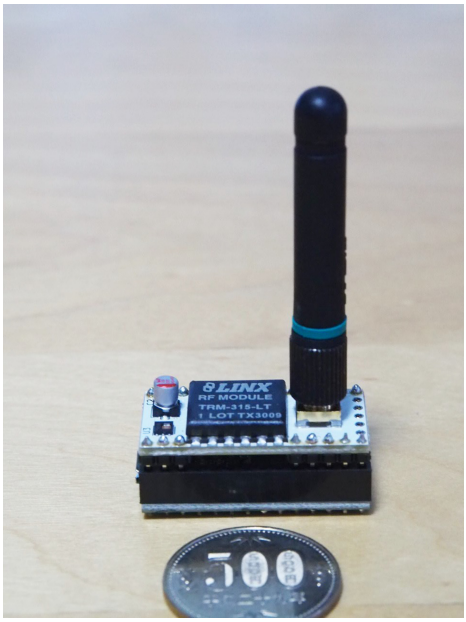

Fig. 5. APCMA node under development, with a Japanese 500 Yen coin for size comparison.

half of the time, we use the time during which the bit has on average a logic high value as the pulse width to be used in the APCMA model, i.e., $\Delta t^{(a)} = 10\,\mu$s. This appears to be a fair way to match both models with each other, since the time slot next to a pulse in APCMA has always a logic low value.

In IEEE 802.15.4, each node first performs a random backoff within the interval of $[0, 2^{BE} - 1]$ before attempting to access the channel. The variable *BE* stands for the *backoff exponent* that is initialized as *minBE* $= 3$ and is doubled every time the channel is sensed to be busy during all transmission attempts of this packet. After the backoff timer expires, the node performs two *clear channel assessments* (CCA) in successive time slots to confirm that the channel is free. The attempting node will only begin its transmission if both CCAs are successful. When either one of the CCAs fails due to an ongoing transmission by another node, the attempting node will perform a new random backoff at an increased backoff window size and will try to access the channel again when this new backoff timer expires. This entire process continues until either the transmission attempt is successful or the maximum backoff limit *maxBO* is reached. We use for both values, i.e., the maximum backoff exponent *maxBE* and the maximum backoff limit *maxBO*, the default value of 5. A flowchart of the simulated CSMA/CA algorithm is shown in Fig. 6.

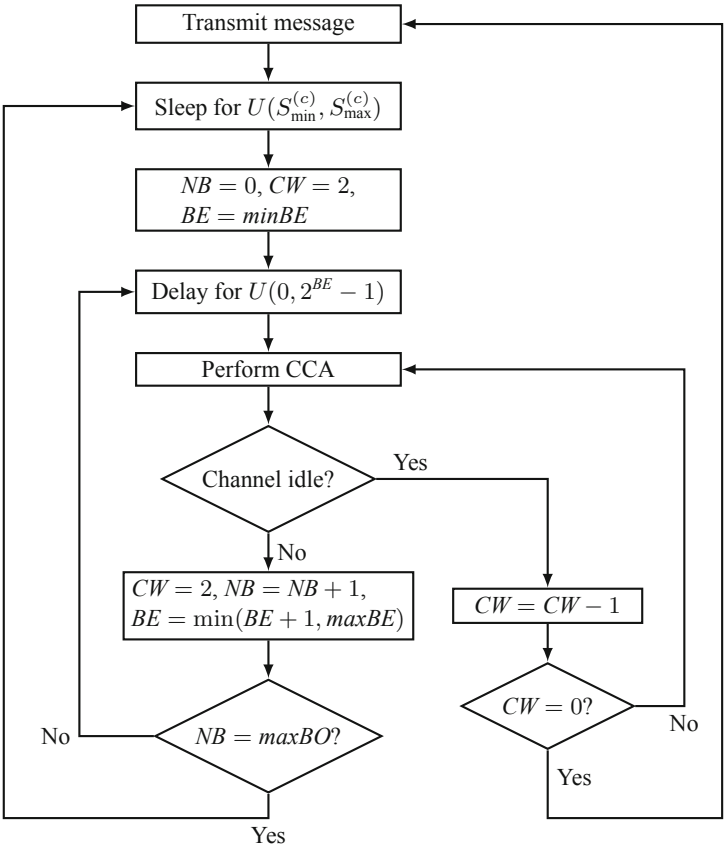

Fig. 6. Flowchart of CSMA/CA algorithm used in simulation

A node that is in a sleep state, will remain in that state for a uniformly distributed random number of unit backoff periods expressed as time slots of length $\Delta t^{(c)}$ in the interval $\left[S^{(c)}_{\min}, S^{(c)}_{\max}\right]$. In our simulations, we use $S^{(c)}_{\min} = 1$ and $S^{(c)}_{\max} = 118\ sf$, with the *sleep factor sf* having values $sf \in \{5, 10, 100\}$, and where the 118 reflects the worst possible number of unit backoff periods a node could be waiting before finally being able to transmit.

Let us assume that there are N_n nodes in the network, where each node k is operating with the same $S^{(c)}_{\min}$ and $S^{(c)}_{\max}$ values. At every time instant t when node k attempts to begin its transmission after two successful CCAs, we distinguish the two cases if node k is the only node beginning to transmit at t, or if there are possibly also other (multiple) nodes beside node k doing the same. We log both cases as A_{k1} and A_{km}, respectively. While from node k's viewpoint both cases would be regarded as successful transmissions, the A_{km} case could in fact lead to a garbled transmission that cannot be correctly

decoded at the receiver. We define the success probability of CSMA/CA from a single simulation as

$$\widetilde{p}_s^{(c)} = \frac{1}{N_n} \sum_{k=1}^{N_n} \frac{A_{k1}}{A_{km}}. \tag{8}$$

In each simulation run we generate the same total number of $M = 100N_n$ messages over all nodes during T simulated backoff periods. We further know that each message has the same length of L bits, so we then obtain the throughput as follows in bits per seconds.

$$\vartheta^{(c)} = \frac{L\,M}{T\,\Delta t^{(c)}} \tag{9}$$

4 Comparative Evaluation of CSMA/CA and APCMA

4.1 Mapping the Parameters Between CSMA/CA and APCMA

Since APCMA and CSMA/CA are different protocols, we need to clarify how we set the parameters in each system to ensure a fair comparison. As our performance metric, we use the success probabilities as defined in Eqs. (4), (7), and (8) in both models. The basis for the comparison is a matching of the throughput of all models. Note that for CSMA/CA backoffs in themselves do not reduce its success probability according to Eq. (8), but they do reduce throughput, since the time intervals between successive transmissions will increase. In the comparison with APCMA when throughput of both models are matched, the success probability of CSMA/CA will then look relatively less favorable, so indirectly backoffs affect the relative performance of CSMA/CA.

For given values of N_n nodes and sleep parameters $S_{\text{min}}^{(c)}$ and $S_{\text{max}}^{(c)}$, we first perform 100 replications of the CSMA/CA simulations as described in the previous section to determine average values $\langle \widetilde{p}_s^{(c)} \rangle$ and $\langle \vartheta^{(c)} \rangle$ over all replications. Given that the APCMA code word length for N_p pulses is $C = (N_p - 2)N_c + N_p + 1$, we compute the maximum sleep limit $S_{\text{max}}^{(a)}$ of the corresponding APCMA model that has the same average throughput as $\langle \vartheta^{(c)} \rangle$ from Eq. (10), where $\Delta t^{(a)}$ is the time for transmitting a single pulse in APCMA.

$$S_{\text{max}}^{(a)} = \frac{2\,L}{\langle \vartheta^{(c)} \rangle \Delta t^{(a)}} - 2\,C - 1 \tag{10}$$

Using this value of $S_{\text{max}}^{(a)}$, we can then determine the success probability of APCMA through simulation and analysis corresponding to the equivalent throughput achieved by CSMA/CA.

4.2 Numerical Results

We now describe the numerical results we obtained from the simulations and analysis described in the previous sections. We consider three different levels of system load having high, medium, and low values that are controlled by the sleep factor *sf* of the maximum sleep time $S_{\text{max}}^{(c)}$ in the CSMA/CA simulation. Depending on the achieved throughput in the CSMA/CA simulation, we adjust the maximum sleep value $S_{\text{max}}^{(a)}$

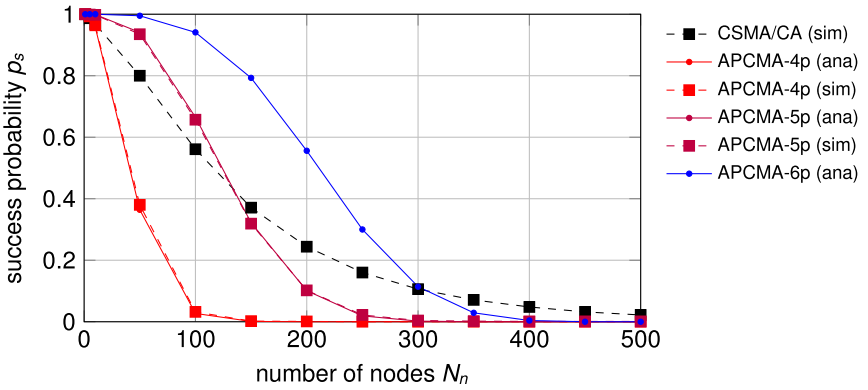

Fig. 7. Success probabilities of CSMA/CA with sleep factor $sf = 5$ and APCMA with corresponding throughput. Dashed lines show results from simulations and solid lines show analytical results. The results from simulation and analysis of APCMA show a near perfect match. The throughput varies between 164 bps for lower numbers of nodes to 129 bps for higher numbers of nodes.

of the APCMA simulation to reach the same throughput. Since the sleep time is randomly selected with a relatively large variance, stationarity of the simulations is almost instantly reached and we therefore don't need to consider an initial transient phase after which the simulation results must be extracted.

In Fig. 7, we show the results for the highly loaded case where we set $sf = 5$. Analytical values are shown for APCMA with solid lines, and simulated values for both models are shown as dashed lines. Each simulation is repeated 100 times with the same settings and since the confidence intervals are very small, we decided to leave them out from the figures for clarity.

We see in Fig. 7 that the success probability in CSMA/CA lies somewhere between that of the three variations of APCMA. The more pulses are used in APCMA, the better its success probability becomes. However, it can also be seen that already for 300 to 400 nodes all models show rather poor performance. Figures 7, 8 and 9 show that the analytical model of APCMA with four and five pulses matches very closely with the results of the simulations.

Figure 8 shows that when the CSMA/CA sleep factor is increased to $sf = 10$, all curves shift to the right and more nodes can be accommodated. The relative performance among all three APCMA models and the CSMA/CA model remains roughly the same.

Finally, Fig. 9 shows a significantly less loaded system with $sf = 100$. While the 4-pulse APCMA variant reaches 50% success probability for about 750 nodes, CSMA/CA and APCMA with five pulses can accommodate about 2500 nodes. At 2500 nodes, the APCMA variant with six pulses even remains above 85% success probability.

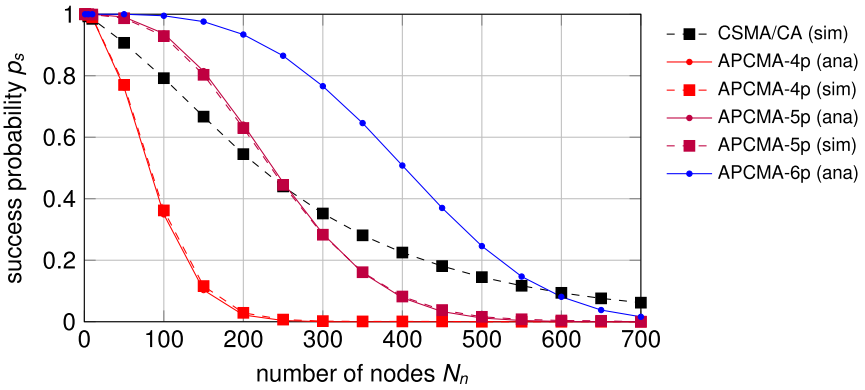

Fig. 8. Success probabilities of CSMA/CA with sleep factor $sf = 10$ and APCMA with corresponding throughput. Dashed lines show results from simulations and solid lines show analytical results. Again, the results from simulation and analysis of APCMA show a near perfect match. The throughput varies between 83 bps for lower numbers of nodes to 69 bps for higher numbers of nodes.

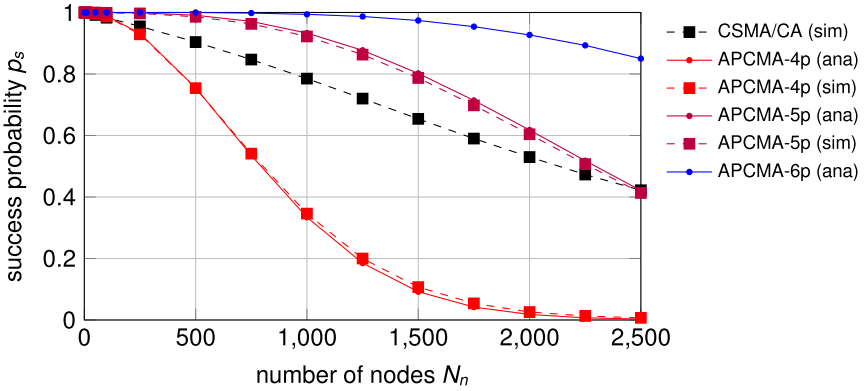

Fig. 9. Success probabilities of CSMA/CA with sleep factor $sf = 100$ and APCMA with corresponding throughput. Dashed lines show results from simulations and solid lines show analytical results. Again, the results from simulation of APCMA match the analysis very well. The throughput is around 8 bps for all numbers of nodes.

5 Discussion and Conclusions

We compared a CSMA/CA-like protocol with APCMA for four, five, and six pulses in terms of the probability that a transmission is successfully received/decoded at a receiver, whereby the parameters of all models are set such that they have similar throughput. Though no simulations were conducted for 6-pulse APCMA, they would likely show very similar performance as the analytical model, given that is the case for 4-pulse and 5-pulse APCMA. APCMA compares favorably to CSMA/CA when five or six pulses per APCMA code word are used, but not so in case of only four pulses,

as our simulations and analytical models show. APCMA compares even more favorable to CSMA/CA when more than six pulses per code word are used according to the analytical model in [10].

CSMA/CA and APCMA are very different protocols, so there are points in which no match in operating conditions are possible. Notably, the employed CSMA/CA protocol is slotted, which means that all nodes are synchronized by a beacon signal. This requires that transmitter nodes need to have the functionality to receive beacon signals, even if they are only intended to be used as transmitters. Nodes working according to APCMA lack this requirement, and so all transmitters do not need receiver functionality. This significantly reduces hardware complexity and power consumption. Depending on the situation, this may also allow scenarios in which 4-pulse APCMA is preferable over CSMA/CA for small numbers (say tens) of nodes, even though 4-pulse APCMA shows a lower success probability.

The coding of CSMA/CA and APCMA also differ radically. Whereas CSMA/CA encodes its packets by binary coding, APCMA uses unary coding. This causes packets of APCMA to be much longer than those of CSMA/CA, but this does not pose problems, since APCMA is designed to be robust to overlaps in time of its transmissions. In fact, this robustness increases with the length of packet size, though it goes at the expense of throughput. The throughput for the fastest scenario investigated in this paper ranges from 129 to 164 bps, given a pulse width of 10 μs in APCMA and an equivalent bit width of 20 μs in CSMA/CA, but it decreases to around 8 bps for the slowest scenario. These bit rates are in line with the type of applications that are supported by the IEEE 802.15.4 standard, like wireless sensor networks.

In both CSMA/CA and APCMA no error correcting coding schemes were used, because we aimed to compare the raw performance of the models. We believe that especially APCMA will benefit from adding error correction, and we will report on this in more detail in future papers.

References

1. Chen, Y., Wang, D., Zhang, J.: Variable-base tacit communication: a new energy efficient communication scheme for sensor networks. In: Proceedings of the First International Conference on Integrated Internet Ad Hoc and Sensor Networks, InterSense'06. Association for Computing Machinery, New York (2006)
2. Feng, D., Das, S., Hajiaghajani, F., Shi, Y., Biswas, S.: Pulse Position Coded medium access in energy-starved networks. Comput. Commun. **148**, 62–73 (2019)
3. Bianchi, G.: Performance analysis of the IEEE 802.11 distributed coordination function. IEEE J. Sel. Areas Commun. **18**(3), 535–547 (2000)
4. Henning, K.: Overview of syndromic surveillance-what is syndromic surveillance? MMWR Morb. Mortal. Wkly. Rep. **53**(Suppl), 5–11 (2004)
5. IEEE STD 802.15.4-2020 (Revision of IEEE STD 802.15.4-2015): IEEE standard for low-rate wireless networks (2020)
6. Laya, A., Kalalas, C., Vazquez-Gallego, F., Alonso, L., Alonso-Zarate, J.: Goodbye, ALOHA! IEEE Access **4**, 2029–2044 (2016)
7. Memish, Z., et al.: Mass gatherings medicine: public health issues arising from mass gathering religious and sporting events. Lancet **393**(10185), 2073–2084 (2019)

8. Misic, J., Shafi, S., Misic, V.: The impact of MAC parameters on the performance of 802.15.4 PAN. Ad Hoc Netw. **3**(5), 509–528 (2005)
9. Peper, F., Leibnitz, K., Hasegawa, M., Wakamiya, N.: Spike-based communication networks with error correcting capability. Brain Neural Netw. **25**(4), 157–164 (2018)
10. Peper, F., et al.: High-density resource-restricted pulse-based IoT networks. IEEE Trans. Green Commun. Netw. **5**, 1856–1868 (2021)
11. Peper, F., et al.: On high-density resource-restricted pulse-based IoT networks. In: GLOBECOM 2020–2020 IEEE Global Communications Conference, pp. 1–6 (2020)
12. Pollin, S., et al.: Performance analysis of slotted carrier sense IEEE 802.15.4 medium access layer. IEEE Trans. Wirel. Commun. **7**(9), 3359–3371 (2008)
13. Ramachandran, I., Das, A., Roy, S.: Analysis of the contention access period of IEEE 802.15.4 MAC. ACM Trans. Sensor Netw. **3**(1), 4-es (2007)
14. Tanaka, C., et al.: Performance evaluation of pulse-based multiplexing protocol implemented on massive IoT devices. Nonlinear Theory Appl. IEICE **12**(3), 1–12 (2021)
15. Tanaka, C., et al.: Implementation of pulse-based multiplexing protocol for massive IoT. In: Proceedings of the 2020 International Symposium on Nonlinear Theory and Its Applications (NOLTA), pp. 346–349 (2020)
16. Tanaka, C., Peper, F., Hasegawa, M.: Application of APCMA protocol to power packet networks for multiplexing power packet transmissions. Nonlinear Theory Appl. IEICE **11**(4), 433–445 (2020)
17. Zhu, Y., Sivakumar, R.: Challenges: communication through Silence in wireless sensor networks. In: Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM, pp. 140–147, August 2005