High-Density Resource-Restricted Pulse-Based IoT Networks

Ferdinand Peper, Kenji Leibnitz, Chiemi Tanaka, Kentaro Honda, Mikio Hasegawa, *Member, IEEE*, Konstantinos Theofilis, Aohan Li, *Member, IEEE*, and Naoki Wakamiya, *Member, IEEE*

***Abstract*—For the realization of an Internet of Things (IoT) with high densities of devices it is necessary that wireless communication protocols are developed that offer 1) low energy consumption; 2) simplicity of encoding and decoding; 3) an asynchronous mode of communication; and 4) an effective but simple method to deal with interference between transmissions. This paper presents the implementation, experimentation, and analysis of a protocol on the MAC sublayer that encodes information in terms of silent intervals between pulses. Based on the representation of patterns of sparse pulses, this encoding has the potential for extremely low power consumption at the transmitter side. It also results in only few conflicts between messages that are broadcast on the same band overlapped in time, while no synchronization between transmitters and receivers is necessary. The protocol is demonstrated experimentally on the 315 MHz band with 100 senders and one receiver configured in a Star topology. Theoretical analysis confirms that the probability of conflicts between messages is low, even if the number of devices increases to the order of ten thousand. This protocol facilitates the implementation of IoT devices that are restricted in terms of hardware and energy resources.**

***Index Terms*—IoT, high-density networks, resource-restricted devices, pulse-based signals, low duty cycle, communication through Silence.**

I. Introduction

BY 2035 the Internet of Things (IoT) is expected to have one trillion devices, which will be deployed on local scales by the thousands at high densities. Though the connected devices in the IoT can be quite complex and sizeable, the majority of devices in future applications will likely be simple, cheap, and small [1], [2]. Devices in such applications will necessarily be *resource-restricted*, meaning that their hardware complexity and energy consumption will both be low. It is also expected that many devices will be deployed in large numbers in limited spaces. In mass gatherings, which may have thousands to millions of people attending [3], [4], early detection of outbreaks of diseases or medical emergencies is crucial for monitoring and controlling them, as well as for providing timely medical care. Syndromic surveillance has benefited from technological advances to achieve these goals, and it is expected to be of increasing use in drop-in surveillance, in which events have a limited duration and may be organized on short notice [5]. In gatherings of hundreds of thousands of people, for example, one may want to monitor the physical health of people via sensors in free hand-out items, such as pens, in order to timely detect the early stage of a pandemic or treat medical emergencies, like heat stroke and fever. As a consequence of being low-cost, such devices need to have a very simple hardware architecture, and they need to be able to work without a pre-configured and dedicated network infrastructure to allow for flexibility.

Key to achieving these goals is the design of the protocol, especially the part that handles the encoding and decoding of information. Lack of a need to synchronize senders and receivers is an important factor that keeps protocols simple, but this usually necessitates a scheduling mechanism to coordinate message transmissions between devices such that collisions are minimized. When collisions of packets are rare, there is limited need for a mechanism to sense whether a channel is occupied, but this would require a low duty cycle of messages transmitted by a device on a channel [6].

Low Power Wide Area (LPWA) networks have been proposed with similar design goals as outlined above. Among these, LoRaWAN [7], [8] and Sigfox [9] are in theory able to serve large numbers of devices, and, though their physical layers differ from each other, they basically adopt the same strategy of ALOHA media access control to deal with collisions of messages [6]. Though ALOHA is simple to implement, it requires the duty cycle of transmissions from each device to be low, effectively limiting throughput [6], [10].

Duty cycles can also be limited by encoding information in terms of long silent intervals between successive pulses, like in the *Communication through Silence* (CtS) scheme [11]. CtS somewhat resembles *Differential Pulse Position Modulation* (DPPM) [12], but differs in that the intervals between pulses are much longer in the former. Encoding in CtS is straightforward: a value is encoded as a linear function of the length of the silent interval between two successive pulses (Fig. 1(a)).

Manuscript received February 15, 2021; revised May 17, 2021; accepted June 7, 2021. Date of publication June 17, 2021; date of current version November 22, 2021. This work was supported by the Strategic Information and Communications R&D Promotion Programme (SCOPE), Ministry of Internal Affairs and Communications, Japan, under Grant 205007001. *(Corresponding author: Ferdinand Peper.)*

Ferdinand Peper, Kenji Leibnitz, and Konstantinos Theofilis are with the Center for Information and Neural Networks, National Institute of Information and Communications Technology, Osaka 565-0871, Japan (e-mail: peper@nict.go.jp; leibnitz@nict.go.jp; kostas@nict.go.jp).

Chiemi Tanaka, Kentaro Honda, Mikio Hasegawa, and Aohan Li are with the Department of Electrical Engineering, Graduate School of Engineering, Tokyo University of Science, Tokyo 162-8601, Japan (e-mail: c-tanaka@haselab.ee.kagu.tus.ac.jp; k-honda@haselab.ee.kagu.tus.ac.jp; hasegawa@ee.kagu.tus.ac.jp; aohanli@ee.kagu.tus.ac.jp).

Naoki Wakamiya is with the Department of Bioinformatic Engineering, Graduate School of Information Science and Technology, Osaka University, Osaka 565-0871, Japan (e-mail: wakamiya@ist.osaka-u.ac.jp).

Digital Object Identifier 10.1109/TGCN.2021.3090044

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

Two problems with CtS are that 1) the length of a message depends on the encoded value and 2) if multiple transmitters send their messages simultaneously, an unrelated pulse from one transmitter may fall in the interval of a code word sent by another transmitter, thus preventing a receiver from extracting the correct information (Fig. 1(b)).

Asynchronous Pulse Code Multiple Access (*APCMA*) has been introduced to deal with these problems through the introduction of redundancy in its encoding that increases the number of pulses from two to four or more per message [13]. Though this somewhat reduces the data rate compared to CtS, it allows many messages to be transmitted at the same time such that a receiver can decode them correctly. In [14] we test APCMA for four pulses using an expansion board developed for an Arduino Mega 2560 microcontroller. The expansion board employs a 315 MHz transceiver that generates pulses through *On-Off Keying* (OOK). Experiments on ten devices divided in two groups, one with eight devices and one with two devices, all operating on the same 315 MHz band, show that transmissions between devices within a group are correctly carried out, and that interference from devices in the other group is minimal. A theoretical analysis based on code words with four pulses is given in [14], taking into account the length of a code word in APCMA, the number of devices that transmit simultaneously, and the broadcast schedule, showing that the probability of unrecoverable message collisions is low. The throughput is also verified experimentally for a network consisting of a limited number of devices.

This paper builds on the results in [14] by refining and generalizing the theoretical analysis of APCMA's performance to an arbitrary number of pulses per code word, as well as to a more general format of messages that contains both an address frame and a data frame. The encoding of these frames is relatively simple, as the value to be encoded in each frame is translated in a straightforward way into consecutive inter-pulse intervals. A pattern recognizer in the receiver uses the relations between the intervals of the code words to recover the original transmitted messages. When messages collide, there is an overlap of their respective intervals, but the protocol is effectively robust against misinterpretation of messages caused by this, since the probability of misinterpretations only slowly increases with the number of senders. Like in [14] the theoretical analysis shows that the probability of unrecoverable messages due to collisions decreases when the length of code words or the lengths of sleeping periods between transmissions increases. Additionally we show that this probability also decreases when the number of pulses used in code words increases. Collisions of pulses from different messages do not hinder the correct decoding of the messages, since the patterns of pulses characteristic to code words are not disrupted by them, provided that overlapping pulses do not cancel each other out. The code words in APCMA differ from those in CtS in that their first and last pulse are at a fixed time interval from each other; this time interval acts as a marker that is picked up by the pattern recognizer. Though fixed in terms of time slots, these intervals may vary slightly between the devices in terms of absolute time depending on the clock speeds of the devices, so the respective intervals can be used as references for the devices' relative clock speeds. APCMA does not require a shared timing reference between transmitters and receivers, and neither is synchronization between them required.

The theoretical analysis in this paper is validated through experiments with up to 100 senders that implement APCMA based on an FPGA with a simple Tx module on the 315 MHz band, and one receiver with the more sophisticated transceiver in [14] as the Rx module. Like in [14], 4-pulse code words are used, but these are combined into two frames, one for the transmitter address and one for the data, resulting in messages with a total of seven pulses each. Unlike in [14] the devices in the experiments are organized in a network with a Star topology [15], [16], with one receiver at the center listening constantly to the remaining devices, which are broadcasting or sleeping. Scheduling is greatly simplified in this case, since the receiver is never asleep, so the probability that it misses broadcasts is significantly reduced. This scenario applies to many sensor networks used in LPWA networks.

This research is suitable for applications that require wireless communication between devices of extremely low complexity and energy consumption. Since it is difficult to charge a large number of devices individually, devices need to be *energy-autonomous*. One way to accomplish this is by harvesting energy from the environment [17], but this requires a flexible and simple scheme such that communication between devices can be initiated whenever energy is available. We believe that APCMA offers the features that can accomplish these goals.

This paper is organized as follows. Section II places this paper in the context of other schemes that use silences to encode information. Section III describes APCMA in more detail, and Section IV proposes a theoretical model that allows us to analyze the probabilities of misinterpretation of messages. These probabilities are validated experimentally in Section V, using hardware specially developed for this purpose. This paper finishes with a discussion and conclusions in Section VI.

II. Related Work

Imagine a lecture hall where a lecturer wishes to identify the student who is the youngest in age. The conventional way to go about this is to ask all students for their ages one after another. Alternatively, the lecturer may instruct the students to encode their ages as a number of seconds, then to count down their respective time intervals upon receiving a start signal from the lecturer, and finally to give a shout once they reach the zero count. Obviously, the student(s) shouting first will be the youngest in the hall. This is an

example of an encoding scheme that represents values by time intervals [18]. Hopfield *et al.* [19] observed that similar coding schemes appear to play a role in neural systems, where the timings of action potentials determine neural responses to cognitive patterns. These schemes have in common that it is not the signals, but the silent intervals between signals that represent information. The scheme in [19] leads to straightforward implementations of pattern matching in neural hardware, whereas alternative algorithms, which require the normalization of pattern vectors, have no known efficient neural implementations.

Simplicity and efficiency are also key to the silence-based communication scheme in [20], which represents zero-values as silences and one-values as pulses. An example is given of an aggregator device that computes an OR function from the binary values of devices in a communication network whereby only the devices with value one transmit one pulse each to the aggregator, while the devices with value zero remain silent. Not only has this kind of minimization of the number of transmitted pulses the potential for reduced energy consumption, it can also lead to faster algorithms, as shown in [20]. A somewhat similar philosophy, in which one or more symbols in an alphabet are represented by a silence, is followed in [21]–[27]. The idea is to design source codes in which one or more symbols occur more frequently than the others, and to represent the most frequent symbol(s) by silences. Energy savings of around 50% can be achieved (more or less depending on the scheme), and there is only a limited penalty in terms of throughput with this approach.

The use of silent intervals in communication originates in early proposals to transmit information covertly [28]–[31], or as a way to "get through" when communication channels are jammed [32], [33]. Also called *Timing Channels*, these schemes use response times, silent intervals between regular bit-encoded packets, and other information related to the timings of transmissions of packets to send additional information that may be difficult to detect. Shifting of packet transmission times is analyzed in [34] with the purpose to increase throughput under various scenarios, including when packets are lost due to collisions. Timing channels are applied to *Bluetooth Low Energy* (BLE) in [35] with the goal of improving energy efficiency by encoding additional information in the duration of silences between successive Bluetooth advertisements. The silences between packets are used to encode control messages in [36], allowing devices with incompatible PHY layers to convey information to each other. The lengths of silences between packets as well as of packets themselves are used in [37] to convey information between devices that are outside each others' reception range but within the carrier-sense range, the latter being typically much larger than the former range.

Variable positioning of pulses in communication has been known as *Pulse Position Modulation* (PPM) [38], [39]. The *Pulse Interval and Width Modulated* (PIWM) code in [40] uses the widths of pulses and the intervals between pulses for modulation, while DPPM [12] and *Digital Pulse Interval Modulation* (DPIM) [41] use only the intervals between pulses. *Multiple Pulse Position Modulation* (MPPM) encodes information as patterns that can be formed when placing a number of pulses in a larger number of time slots [42]. The intervals between pulses in all PPM-based schemes tend to be relatively short, limiting energy savings.

CtS [11] is aimed at efficient implementations of wireless sensor networks. These networks typically require only limited throughput, but their energy budget tends to be very tight. Zhu and Sivakumar [11] list a series of challenges that need to be resolved for CtS to be useful in practice.

Challenge 1 (Framing): How many time slots should there be in a frame, or, equivalently, what is the maximum allowed length of a silence? A practical frame size is 256 to 65536 slots, according to [11], which corresponds to the encoding of 8 to 16 bits, respectively. Longer frames will decrease throughput, while shorter frames may make the silent intervals to be too short to be of practical use.

Challenge 2 (Addressing): Since the address space of a network may be large, embedding an address within one frame will likely make a frame longer than recommended in Challenge 1 above.

Challenge 3 (Sequencing): This refers to the requirement that a receiver needs to reconstruct information in the same order that the transmitter had originally sent it. The traditional solution of using a packet ordering number may make frames longer than recommended in Challenge 1.

Challenge 4 (Error Correction): This is an issue on multiple levels. First, the accuracy of the clocks of the devices is important, because it determines the reliability by which information is transferred between sender and receiver. The clocks of a sender and receiver also need to be synchronized during a transmission for the same reason. Second, implementing error correction in the traditional way may lead to suboptimal solutions, since such schemes are designed for bit streams rather than for pulses. Consequently, error correction based on the admittance of certain lengths or combinations of lengths of silent intervals may be more suitable for CtS.

Challenge 5 (Contention Resolution): How to avoid collisions between transmissions with overlapping frames from different devices? A traditional solution, *Carrier Sense Muliple Access with Collision Avoidance* (CSMA/CA), is not feasible for CtS if silent intervals are long, because it is impossible to distinguish between silences corresponding to transmissions in progress and silences resulting from sleeping devices. In other words, silences can not be sensed.

Throughput is limited in CtS when large values are encoded due to the resulting long silences. The *Variable-Base Tacit Communication* (VarBaTaC) [43] scheme increases throughput by representing the encoded value in a number system with a base that is much smaller than the typical lengths of silences in CtS, yet large enough that the individual digits are limited in number. This results in multiple silent intervals of shorter lengths, placed next to each other, with the total lengths of the shorter silences being much smaller than the corresponding silence in CtS would be. Contention resolution with VarBaTaC based on *Time Division Multiple Access* (TDMA) and CSMA/CA is discussed as well in [43].

Representation in a number system with a certain base is also done in a scheme called *Pulse Position-Coded PDU* (PPCP) [44], [45]. Called *Flexible Base Digit*

Separation (FBDS), this number system uses a base that is selected to minimize the energy consumption as compared to an equivalent traditional bit-encoded scheme, while throughput is maximized. A message in PPCP has a format that includes the number of fields, the sender ID, a type indicator, and data values. These fields are prefixed by a series of four consecutive pulses and postfixed by a series of three consecutive pulses, whereas fields are separated by pairs of consecutive pulses. Each of these fields are represented in terms of FBDS, whereby single pulses separate the digits. An error occurs if a message does not conform to this format, and though not perfect, errors can be detected in many cases by checking the format at the receiver side. Multi access is investigated by simulations based on ALOHA in [44] and CSMA/CA in [45]. This offers limited contention resolution, since an overlap of even two messages causes an error in both schemes, and the probability of an error increases with larger values of the base in FBDS, because this results in longer silences on average, and thus longer messages. The resulting MAC layer is applied in a Star topology in [44], whereby the devices are in transmit-only mode and one aggregator receives all messages, but a more general architecture is proposed in [45], whereby each device can both transmit and listen. Since a receiver consumes energy during silences due to its need to continuously listen, a short length of silences is emphasized in [45]. A hardware implementation is presented in [45], but multi-access appears to be implemented only to a limited extent: while 64 devices are used in simulations, a system of two devices has been deployed in green houses for agricultural parameter sensing. A chaotic version of PPCP is proposed in [46] with the aim of adding security by making the lengths of silences unpredictable.

Yet another CtS scheme using a number system with a certain base is presented in [47], but data is compressed by first sorting the digits, and taking the differences of subsequent digits except the first digit, which is left as is. This information is transmitted together with a permutation number indicating the original order before sorting. Sorted CtS halves the delay of CtS, but it doubles the energy consumption and error probability. Yet, both sorted CtS and raw CtS compare favorably to traditional bit-encoded packet-based communication. The sorted CtS scheme is implemented in Magnetic Induction-based communication (MI). MI is usually applied when no high data rates are expected but the energy budget is very tight, and it is more suitable than RF in environments involving aqueous and animal/plant tissue media, like sensing in food transportation and distribution, underground sensing in agriculture, and underwater communication.

Another scheme employing sorted CtS is proposed in [48] for bacterial communication on a microfluidic chip, together with error correction specific for this application. Bit rates of 10^{-4} bps are achieved, which is ten times the rate traditional techniques can attain.

The *WiChronos* scheme in [49] implements CtS by assigning a unique preamble and postamble to each sender and use these as respectively the start and stop signal of CtS to delimit a silent interval. The preamble and postamble are encoded as regular bit-encoded packets, and this allows overlaps of messages from different senders, since a receiver can uniquely identify which preamble and postamble belong to each other. A preamble and postamble of 2 bytes each is used in [49] to encode an address space of 2^{15} devices, with one bit encoding the type pre/post-amble. For the typical value encoded in a silence equivalent to up to two bytes this gives an overhead of at least four bytes, but the addresses of senders need to be sent anyway in messages, so this overhead appears acceptable. Though this scheme allows multi access in principle, there is still a non-zero probability that the pre/post-ambles of different devices collide with each other, and this is addressed by using ALOHA. This scheme is implemented in hardware, and its performance is verified with 12 devices in heavy message traffic conditions.

The APCMA protocol we propose in the current paper realizes contention resolution by defining a code book of pulse combinations that have maximum distances to each other. Since it allows overlaps of messages as well as overlaps of pulses, it is much less sensitive to collisions than other silence-based schemes proposed thus far. As a result, APCMA does not require contention resolution mechanisms like ALOHA and CSMA/CA. This simplifies the scheduling of transmissions by devices significantly, since each device can initiate transmissions whenever they like without considering transmissions from other devices. APCMA's contention resolution mechanism effectively includes error correction against intermittent pulses to a great extent. Since the start and stop pulse in a code word are spaced at a fixed interval, they are easily identified by a pattern recognizer, so no trains of pulses like in [44], [45], [49] are necessary to mark these delimiters. Additionally, the fixed spacing between these delimiters acts as a timing reference for a device's clock, and it takes away the need of devices to synchronize with each other.

Compared to other CtS-inspired schemes, the number of pulses in each code word is low. In order to keep the length of messages limited, we split them in two in this paper, with the first half encoding the address and the second half encoding the data. This resembles the encoding in terms of bases in a number system like in [43]–[45], though it is less explicit. One application in which APCMA has been tested, albeit at a small scale, concerns the distribution of power packets in an electric network [50]. Such networks are expected to be useful in robotics in which power needs to be routed to certain actuators in a flexible way such that the amount of wiring stays limited. This application requires a simple protocol like APCMA that resolves contention in a straightforward way. Similarly, we expect APCMA to be useful in IoT applications that have simple hardware and a tight energy budget.

III. Asynchronous Pulse Code Multiple Access

A. Pulse-Based Codes

Like in CtS, we divide the time axis into time slots of equal and constant width. A time slot is defined to be *occupied* (*have value one*) if and only if it contains a pulse exceeding a certain power level, otherwise it is called *empty* (*have value zero*). In CtS an integer value x in the range $[1, V]$ is encoded as an interval of x empty time slots lying between two occupied time slots.

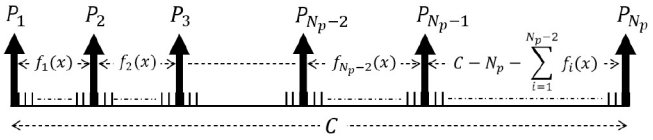


Fig. 2. Format of a code word in APCMA.

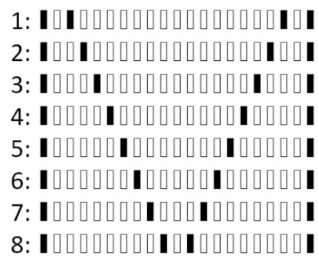


Fig. 3. APCMA code with four pulses and eight code words, with encoding functions $f_1(x) = x$ and $f_2(x) = C - 4 - 2x$.

APCMA extends CtS by encoding a value based on intervals between four or more pulses rather than two pulses. The resulting redundancy of the information improves the robustness to interposing pulses, as we show in Section IV, and as a result it becomes possible to correctly decode a code word with a high probability even if it is overlapped with another code word. The use of four or more pulses allows code words to be designed such that they are always of fixed length, independent of the encoded value. The length of a code word is defined to be *C* time slots, each of unit length, whereby *C* is an integer constant. The first and the last time slot each contain a pulse. These two pulses delineate an interval with a constant length of $C - 2$ time slots, and they act as a marker that can be easily picked up by a decoder. By definition a time slot next to a pulse never contains a pulse.

Assuming code words with N_p pulses each, we encode a value *x* in APCMA as $N_p - 1$ intervals of successive lengths

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

whereby the functions f_i are one-to-one (Fig. 2).

Called the *encoding functions*, the functions f_i are predetermined and shared among the transmitters and the receivers. Functional relations between the functions f_i give the code its robustness to misinterpretation by a decoder in case a mixture of different code words occur. In the experiments in this paper we use $N_p = 4$, $f_1(x) = x$, and $f_2(x) = C - 4 - 2x$. Functions f_1 and f_2 may be chosen differently, depending on the probability distribution of the values that *x* can assume, so that the coding space is optimally utilized. For example, if *x* represents a sensor's output values with a Gaussian distribution, then $f_1(x)$ may be chosen so that its values are uniformly distributed. Due to the choices of the functions f_1 and f_2 in this paper, the silent intervals in a code word encoding the value *x* have the lengths *x*, $C - 4 - 2x$, and *x*, respectively. Fig. 3 gives a code in this format with eight code words.

B. Bounds on Codes

The code in Fig. 3 has three empty columns, which are required because two pulses are not allowed to be in neighboring time slots in a code word. In general, an APCMA code with a pulse in both its first and last time slots always contains at least two empty time slots. Except for the first and last time slots, each column contains at most one pulse, because two code words with pulses common in other time slots would resemble each other too much, making it more difficult for the decoder to distinguish them. The code in Fig. 3 has a third empty column because of the particular format it has. This implies that the code length is $C = 2(N_c + 2) + 1$, where N_c is the number of code words. Given a certain value of the code length *C*, the maximum number of code words in a code of this format then becomes:

$$N_c = \left\lfloor \frac{C - 5}{2} \right\rfloor \quad (2)$$

More in general, for a code of length *C* with N_p pulses per code word in a format of at most one pulse per column except for the first and last column, it holds that:

$$C \geq 2(N_c + 2) + \big(N_p - 4\big)N_c \quad (3)$$

This implies the following upper bound on the number of code words:

$$N_c \leq \left\lfloor \frac{C - 4}{N_p - 2} \right\rfloor. \quad (4)$$

C. Decoding

Decoding of an APCMA code word in a receiver takes into account the possibility that unrelated interposing pulses can occur anywhere in the silent intervals defined by the code word. Decoding algorithms recognize patterns of pulses corresponding to code words of APCMA and ignore other pulses. One such algorithm is based on a so-called *spike automaton*, which is a kind of finite automaton, except that instead of recognizing strings, it recognizes sequences of pulses at certain time stamps [13]. Decoding in this way requires an algorithm that creates and deletes data structures representing spike automata in a dynamic fashion, and it is this algorithm that is implemented on the Arduino microcontroller used in [14].

This paper uses an implementation based on FPGAs in the experiments, which are less suitable for the dynamic nature of spike automata. This is why we use a different decoding method here, i.e., a method based on *shift registers* [51]. According to this method, the data sampled at a receiver is input to the left of a shift register at a frequency corresponding to the width of a time slot, whereas the contents of the shift register shifts to the right each clock tick. The shift register consists of *C* cells, so one code word fits exactly in it at a time. The cells in the shift register are connected to a logic circuit that outputs a logic one corresponding to a code word that is recognized (Fig. 4).

The logic circuit can be kept simple if the structure of the code is regular, which is the case for the code in Fig. 3.

In order for decoding to work optimally, a code is designed such that code words differ as much as possible from each

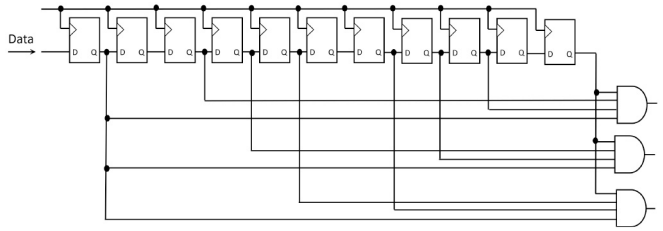


Fig. 4. Shift register with logic circuit that recognizes 4-pulse codes of the form in Fig. 3 that have length $C = 11$. This code length corresponds to a code with three code words.

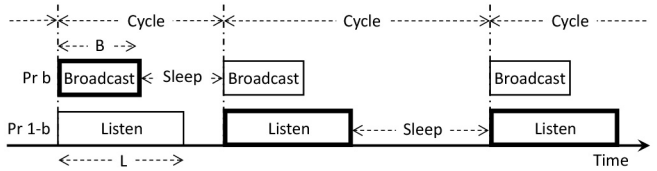


Fig. 5. Scheduling of a device. After each sleep interval the device becomes active, transitioning into Broadcast mode with probability b and Listen mode with probability $1 - b$. The modes the device goes through are indicated by thick-lined boxes. In this example, the device goes successively through the modes Broadcast – Sleep – Listen – Sleep – Listen.

other. For the code in Fig. 3 a code word differs in four time slots from any other code word. This means that if one unrelated pulse occurs during the transmission of a code word of four pulses, any interpretation of the resulting five pulses differs in only one pulse from that code word, but it differs in at least three pulses from any other code word. More in general, given a well-designed code having code words with N_p pulses each, then the occurrence of $N_p - 3$ unrelated pulses in a transmission will not hinder the correct interpretation of code words. If more unrelated pulses occur, then a transmission may be misinterpreted if those pulses occur in time slots that correspond to pulses in another code word.

We emphasize that even if our codes are robust to interposed pulses from unrelated transmitters, they are less well-equipped to deal with missing pulses, caused by for example noise. In its current implementation, decoding of a code word fails when even one pulse is missing. It is possible to alleviate this shortcoming by adjusting the decoding algorithm, but it goes at the code's ability to deal with interposed pulses. However, this problem is less severe in codes with a higher number N_p of pulses per code word. Since we only have four pulses per code word in this paper, we will not address missing pulses, and rather assume that every pulse that is transmitted by a sender will eventually reach the intended receiver(s).

The robustness to incorrect decodings of APCMA codes depends not only on the number of pulses in code words, but also on the exact placement of those pulses, i.e., on the design of the code. The 4-pulse code in Fig. 3, defined by the functions $f_1(x) = x$ and $f_2(x) = C - 4 - 2x$, is well-designed in the above sense. Though it is possible that a train of pulses is decoded by a receiver as a valid code word while no transmitter has sent it, the probability of such a code word emerging from coincidental overlaps of various code words from different transmitters is low when the code length C is large and the number N_n of transmitting devices is small, because of the resulting low density of pulses. This probability becomes even lower when the number of pulses in a code word is higher, for the same reason that the odds of winning a prize in the powerball lottery decreases with an increase in the number of balls that must match: if there are more pulses in a code word, more of them should match with a random pattern of pulses being transmitted by all senders for there to be a misinterpretation. Section IV explores this relationship in more detail.

D. Scheduling

APCMA can be employed according to various scheduling algorithms of devices. We distinguish three modes of a device: 1) *broadcast*; 2) *listen*; and 3) *sleep*. When a device is not in Sleep mode it is called *Active*. In Broadcast mode a device transmits its value x once, and this mode lasts for B time slots with $B > C$. We call B the *effective code length*. Apart from C time slots to broadcast a code word, a constant overhead of $B - C$ time slots is assumed in Broadcast mode for running the encoder and other peripheral processes on the controller.

In Listen mode, which lasts for L time slots, a device is only able to receive broadcasts. It is recommended that L is several times larger than B to have a reasonable chance that a broadcast is completed before the Listen phase in a receiver ends. After a Broadcast or Listen phase ends, a device goes into Sleep mode. The duration of a Sleep phase is a random variable that is uniformly distributed over the interval $[sC, (s + \sigma)C]$, with the parameters s, the *sleep factor*, and σ, the *sleep spreading factor*, both being non-negative. After emerging from Sleep mode, a device will become active again, and go into Broadcast mode with probability b and into Listen mode with probability $1 - b$. The interval consisting of a Broadcast or Listen phase, followed by a Sleep phase is called a *cycle* (Fig. 5).

The length of a cycle varies, depending on whether an active device is broadcasting or listening, as well as on the (random) length of the sleep interval.

Each device is oblivious to the scheduling of other devices, and no contention resolution schemes like CSMA/CA or ALOHA are required to recover from collisions, because overlaps of transmissions from different devices result at worst in ambiguities in how trains of pulses are interpreted. The underlying information does not get lost, but only becomes more difficult to extract by a receiver.

IV. Analysis of Misinterpretation Probabilities

A. Probabilities for Single-Frame Code Words

Since the decoding in APCMA recognizes particular patterns of pulses as valid code words, the overlap of pulse patterns from various code words at various time offsets may cause a pattern to emerge that corresponds to a valid code word, yet that has not been transmitted by any device. When a code word c has been transmitted in a certain frame F of C consecutive time slots, and there is more than one interpretation possible by the receiver due to interposing

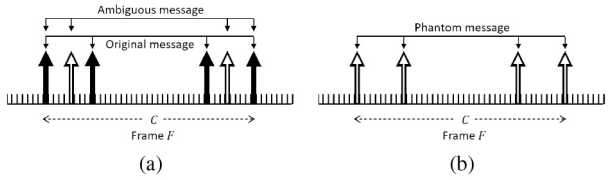


Fig. 6. (a) Ambiguous message arising from unrelated pulses (open arrows) in addition to the original message that has been sent. (b) Phantom message arising from unrelated pulses. The difference with an ambiguous message is that the first pulse and the last pulse of a phantom message do not come from one and the same transmitted code word.

pulses from other transmitters, we have a so-called *ambiguous message* or *ambiguity* [see Fig. 6(a)].

We denote such an event by $\mathcal{A}$. A different case occurs when a frame can be interpreted in more than one way, but its first and last time slots are filled with pulses that do not originate from one and the same transmitted code word. In this case we have a *phantom message* or a *phantom* [see Fig. 6(b)], and an event of the occurrence of a phantom is denoted by $\mathcal{P}$. For a large code length C and a small number of devices the probabilities of ambiguities and phantoms are low, and they tend to decrease if the number N_p of pulses per code word increases. In this section we quantify these probabilities in more detail.

Let p be the probability that a time slot is occupied by a pulse in case there are N_n devices (the subscript in N_n stands for *nodes*). We assume that all time slots are statistically independent. For large values of C and N_n this assumption can be considered reasonable. In order to determine p, we consider a cycle consisting of a Broadcast or Listen phase followed by a Sleep phase (see Fig. 5). Taking into account the lengths of these phases, the broadcast probability b, the sleep factor s, and the sleep spreading factor σ, we arrive at an expected length of this cycle in terms of the number of time slots as:

$$K_{\mathrm{Cy}} = bB + (1-b)L + (2sC + \sigma C)/2 \tag{5}$$

When there is only a single device, the expected number of pulses in this interval is $N_p b$ time slots, so we then arrive at an average pulse density of:

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

When ergodicity is assumed, this represents the probability that a time slot is occupied in case there is one device.

Transmissions from different devices are statistically independent, because the devices operate independently from each other due to there being no contention resolution mechanisms like CSMA/CA or ALOHA. The probability that a time slot is empty then becomes $(1-q)^{N_n}$. We operate under the assumption that pulses falling in the same time slot do not cancel each other out. So, a time slot with more than one pulse is treated as if it contained one pulse. Under this assumption, a time slot is considered occupied if at least one of the N_n devices transmits a pulse in this time slot, so the probability that a time slot is occupied in case of N_n devices is then:

$$p = 1 - (1-q)^{N_n} \tag{7}$$

Given the value of p, we first determine the probability of an ambiguous message. For a random sequence of pulses in F to be recognized as a valid code word c' other than the transmitted code word c, it is necessary that there is at least one combination of N_p pulses in F that matches c'. Since code words c and c' have pulses in common in time slots 1 and C, they can only be distinguished from each other by their remaining $N_p - 2$ pulses. We call these pulses *non-boundary pulses* to distinguish them from the pulses in time slots 1 and C. The probability of the $N_p - 2$ non-boundary pulses of code word c' to occur in $N_p - 2$ specific non-boundary time slots of F is p^{N_p-2}, and the probability that at least one of these non-boundary time slots of F does not contain a pulse is then $1 - p^{N_p-2}$. The non-boundary columns in an APCMA code contain at most one pulse, i.e., no code words share a non-boundary pulse, and the interpretations of c and c' do not influence each other, so we can consider the occurrences of both code words' non-boundary pulses in the frame F to be statistically independent. This implies that the probability that there is no match of the pulse pattern in F with the non-boundary pulses of any of the $N_c - 1$ code words differing from c, equals $(1 - p^{N_p-2})^{N_c-1}$. The complement of this probability is then the probability that the pulse pattern in F can be misinterpreted as one or more of the code words c' differing from c, given that c was transmitted:

$$\Pr[\mathcal{A}] = 1 - \left(1 - p^{N_p-2}\right)^{N_c-1} \tag{8}$$

The probability of a phantom message is different from (8) in the sense that there is one more code word that can match the random pulse pattern in F, so we need to increase the exponent $N_c - 1$ by one in the calculation of probabilities. Also, we need to take into account pulses in the first and last time slots of F, which all code words of the code share. The probability that the first and last time slots both contain a pulse is p^2, and the probability that at least one of these time slots contains no pulse is $1 - p^2$. Then the probability that at least one of the time slots in any of the code words does not contain a matching pulse in F is equal to:

$$1 - p^2 + p^2\left(1 - p^{N_p-2}\right)^{N_c} \tag{9}$$

The probability of a misinterpretation of the pulse pattern in F as a valid code word then equals the complement of this probability, which rewrites to:

$$\Pr[\mathcal{P}] = p^2\left(1 - \left(1 - p^{N_p-2}\right)^{N_c}\right). \tag{10}$$

B. Probabilities for Double-Frame Code Words

Since a code word becomes too long if it encodes a large value, it is necessary to separate it into multiple frames of smaller lengths. We analyze the case in which a pair of frames is used, the first frame encoding the address of the transmitter and the second frame encoding the data. Double frames are also used in the experiment in Section V. The two frames are encoded independently of each other according to the single-frame codes we have discussed so far, and they are combined such that the last pulse of the address frame coincides with

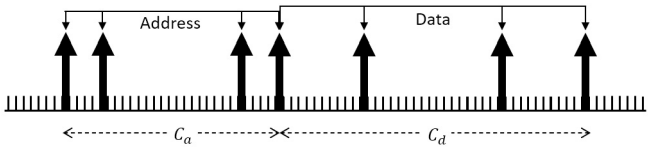


Fig. 7. Format of a message consisting of an address frame of length C_a and a data frame of length C_d time slots based on a code of four pulses. The two frames have one pulse in common, so the total number of pulses of the message is seven.

the first pulse of the data frame. Fig. 7 shows this format in the case the address and data frames are both encoded by 4-pulse codes, with the frames having code lengths C_a and C_d, respectively.

The length in time slots of a message then becomes $C = C_a + C_d - 1$. In general, if the address frame is encoded by N_{p_a} pulses and the data frame by N_{p_d} pulses, then a message is encoded by $N_{p_a}+N_{p_d}-1$ pulses. We indicate the number of single-frame code words of the address frame and data frame as N_{c_a} resp. N_{c_d}. Since the address space should be sufficient to address all devices, we then have:

$$N_{c_a} \geq N_n \tag{11}$$

Using (4), (3), and (11), we obtain:

$$\begin{aligned} N_{c_d} &\leq \left\lfloor \frac{C_d - 4}{N_{p_d} - 2} \right\rfloor = \left\lfloor \frac{C - C_a - 4}{N_{p_d} - 2} \right\rfloor \\ &\leq \left\lfloor \frac{C - 2(N_{c_a} + 2) - (N_{p_a} - 4)N_{c_a} - 4}{N_{p_d} - 2} \right\rfloor \\ &= \left\lfloor \frac{C - N_{c_a}(N_{p_a} - 2) - 8}{N_{p_d} - 2} \right\rfloor \\ &\leq \left\lfloor \frac{C - N_n(N_{p_a} - 2) - 8}{N_{p_d} - 2} \right\rfloor \end{aligned} \tag{12}$$

Let $\mathcal{A}_{ad}$ and $\mathcal{P}_{ad}$ denote the events that an ambiguity resp. phantom occurs in a combined message. Following the same reasoning as in the derivation of (8), we then derive the probability of event $\mathcal{A}_{ad}$ to be equal to:

$$\Pr[\mathcal{A}_{ad}] = 1 - \left(1 - p^{N_{p_a}-2}\right)^{N_{c_a}-1} \left(1 - p^{N_{p_d}-2}\right)^{N_{c_d}-1} \tag{13}$$

Similarly, the probability of a phantom event $\mathcal{P}_{ad}$ then equals:

$$\Pr[\mathcal{P}_{ad}] = p^3 \left(1 - \left(1 - p^{N_{p_a}-2}\right)^{N_{c_a}} \left(1 - p^{N_{p_d}-2}\right)^{N_{c_d}}\right) \tag{14}$$

The reason for the exponent of the probability p being 3 in (14), rather than 2 as in (10), is that there are three pulses with fixed positions in the code words, i.e., in the first time slot, the last time slot, and the time slot shared by the address and data frames.

C. Probabilities in Star Topology

In [14] we use a scenario in which all devices are both transmitters and receivers, whereby the probability of broadcasting is around $b \approx 0.6$. In the current paper we assume there is only one receiver that is listening all the time, whereas the remaining devices are all transmitters that are allowed to sleep according to the parameters s and σ. This scenario corresponds to a Star topology, in which the receiver at the center receives the broadcasts from the other devices, all of which transmit with probability $b = 1.0$ in their active phases. Transmit-only devices broadcasting to a single receiver have been analyzed in [15] and demonstrated in [16] to offer lower-cost implementations of sensor networks as compared to networks with bidirectional communication with comparable throughput. In these scenarios energy consumption of the receiver tends to be less important since it is usually in a location where power infrastructure is available.

How does this scenario work out for the probabilities of ambiguities and phantoms? Let's assume that the address space is optimally used, i.e., that $N_{c_a} = N_n$. Furthermore, we assume that the code contains the minimum number of merely two all-zero columns; this implies that equality applies in (12). We can then evaluate the probabilities of ambiguous and phantom messages according to (13) and (14). Fig. 8 shows these probabilities for codes with $N_{p_a} = N_{p_d} = 4$ pulses in the scenarios of $N_n = 10$, 100, 1000, and 10000 devices with the probability of broadcasting $b = 1.0$ and the sleep spreading factor $\sigma = 20$, whereby the code length C ranges from 100 to 10^8 and the sleep factor s ranges from 1 to 10^5.

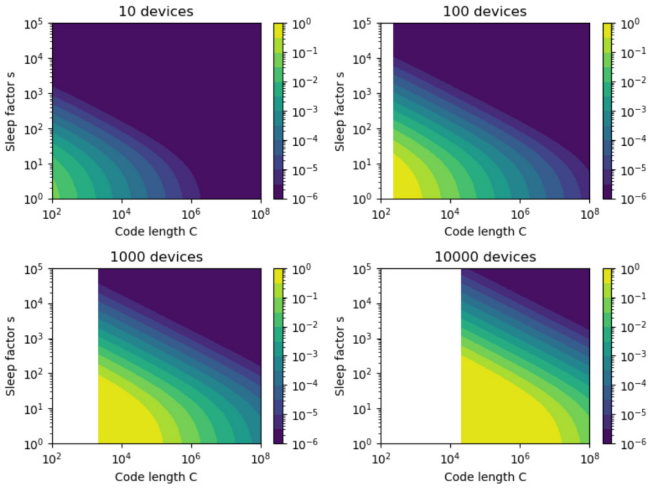
As seen from the figure, the probabilities of ambiguous and phantom messages are low for large values of the code length C and the sleep factor s. Increasing the values of C and s goes at the cost of throughput, though, as the average length of a cycle according to (5) will increase. We also see that the probability of ambiguous messages tends to be higher than the probability of phantom messages. This follows directly from (13) and (14), because they imply that $\Pr[\mathcal{P}_{ad}]/\Pr[\mathcal{A}_{ad}] \approx p^3$ when N_{c_a} and N_{c_d} are large, which is mostly the case. A more informal explanation is that for ambiguous messages there are three time slots that are guaranteed to be filled with a pulse each, so there are less time slots to distinguish ambiguous messages from valid messages. For phantom messages, however, there is no guarantee that these three time slots are occupied, so they differ more from the case in which no message is transmitted at all, making the probability of their occurrence lower.

Another way to reduce the probabilities of ambiguous and phantom messages is by increasing the number of pulses per code word. We focus on ambiguities, since their probabilities are much higher than the probabilities of phantoms. We again assume that the number of pulses in the address and data frames are equal, with the frames having one pulse in common, like in Fig. 7. Fig. 9 shows the probability of ambiguous messages for codes with 4, 5, 6, and 7 pulses in the address and data frames each in the scenarios of $N_n = 100$ devices and $N_n = 10000$ devices with the same parameter settings and parameter ranges as before.

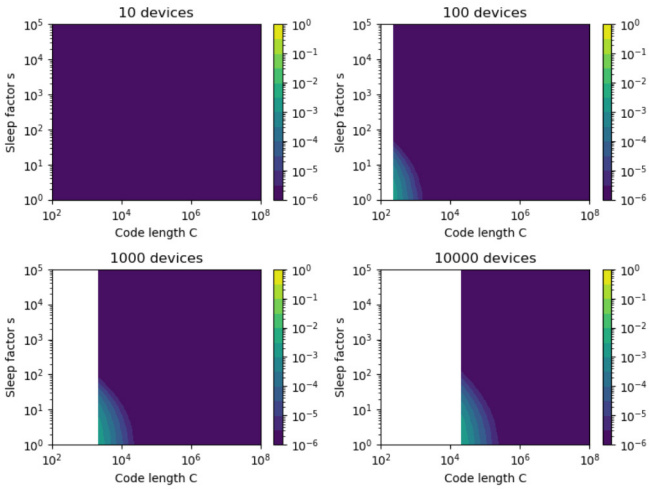
The probability of ambiguous messages indeed decreases with an increase in the number of pulses in a code word.

V. Experiments

To validate the theoretical model in the previous section, we conduct experiments using a Xilinx Spartan-3E FPGA to control the Tx and Rx modules (see Fig. 10).

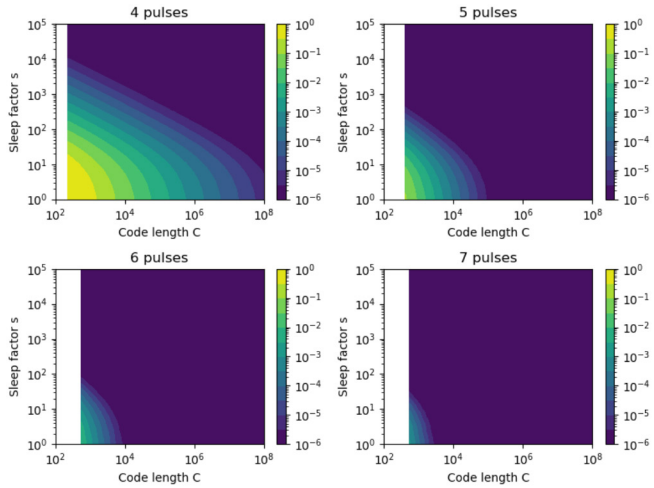


(a) Probability of ambiguities with $N_{Pa} = N_{Pd} = 4$ pulses

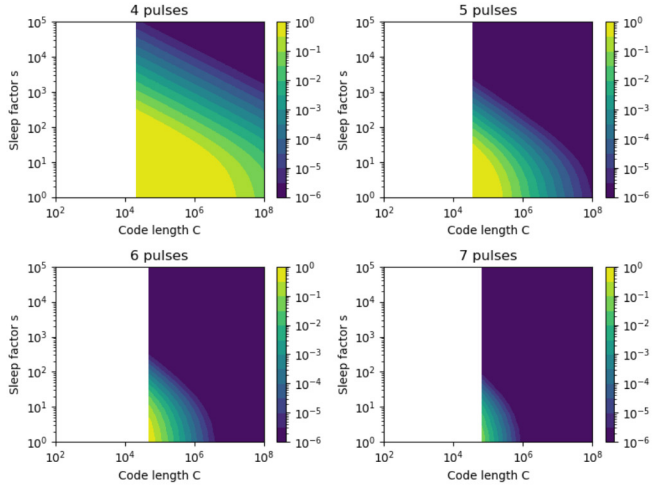


(b) Probability of phantoms with $N_{Pa} = N_{Pd} = 4$ pulses

Fig. 8. (a) Probability of ambiguous messages according to (13) for $N_n =$ 10, 100, 1000, and 10000 devices in a code in which both a 4-pulse address frame and a 4-pulse data frame are combined in 7-pulse code words. Length of Broadcast phase is set to $B = C$, while devices have a broadcast probability of $b = 1.0$ in their active phase, and the sleep spreading factor is $\sigma = 20$. The horizontal axes denote the code length C and vertical axes denote the sleep factor s. The white areas are undefined since a minimal value of C is required due to the fact that the number of code words in the address frame is set to be equal to the number of devices. (b) Probability of phantom messages according to (14) for the same parameter settings.



(a) Probability of ambiguities with $N_n = 100$ devices



(b) Probability of ambiguities with $N_n = 10000$ devices

Fig. 9. (a) Probability of ambiguous messages according to (13) for $N_n =$ 100 devices in a code with 4, 5, 6, and 7 pulses per address and data frame of a code word, with the same parameter settings as in Fig. 8. (b) Same, but for $N_n = 10000$ devices.

We have one receiver, which is equipped with the Linx TRM-315-LT 315 MHz baseband transceiver operating in Rx mode; this is the same transceiver as used in [14]. A Raspberry Pi is connected to the receiver for data collection. The senders are equiped with the Fs1000A315 315 MHz baseband transmitter, which is a low-cost transmitter. The devices are placed in a Micronix MY1530 shielded box at a distance of 20 cm on average. The experiment is conducted with $N_n = 20$, 40, 60, 80, and 100 senders with the parameter settings $b = 1.0$, $N_{p_a} = N_{p_d} = 4$ pulses, and sleep parameters $s = 90$ and $\sigma = 20$. The functions $f_1(x) = x$ and $f_2(x) = C - 4 - 2x$ are used for encoding both the address and the data frames of messages, with the number of code words of the respective frames being $N_{c_a} = N_{c_d} = 127$. This implies that the minimum required value for C is $C = C_a + C_d - 1 = 2 \times (127 + 2) + 1 + 2 \times (127 + 2) + 1 - 1 = 517$ time slots; we choose the slightly larger value of $B = 520$ for the effective code length to account for overhead. The length of a time slot is 1.25 ms and the duration of an experiment is 2000 s, which is equivalent to 1.6 million time slots.

Over the time of one experimental session each device sends 30 messages on average. The address frame of a message contains the sender's address, and the data frame is a pseudo-random number that is known at the receiver side for verification. This allows us to measure the rate of ambiguous messages. Though it is possible to detect multiple messages in the shift register of the receiver at each clock tick, in the current implementation only one of the messages can be transferred to the Raspberry Pi at a time, so only one (randomly determined) message is transferred at each clock tick and the remaining messages are disposed of. Since this affects both

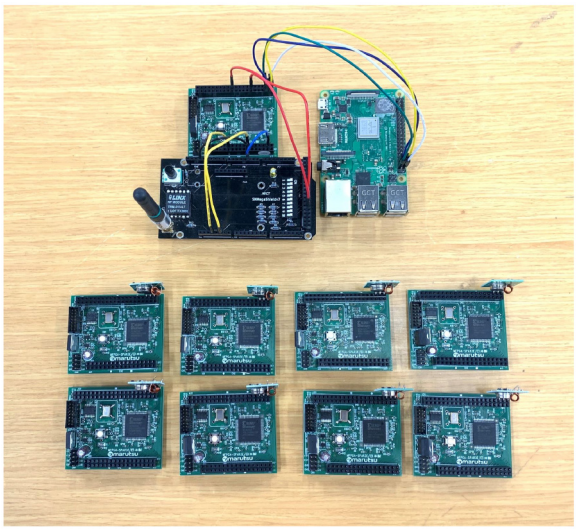


Fig. 10. Devices used in the experiment. (Top) Receiver, consisting of a Xilinx Spartan-3E FPGA, a Linx TRM-315-LT 315 MHz baseband transceiver, and a Raspberry Pi for collecting data. (Bottom) Eight of the senders, each consisting of a Xilinx Spartan-3E FPGA and a Fs1000A315 315 MHz baseband transmitter. The experiment is conducted with a number of 20, 40, 60, 80, and 100 senders.

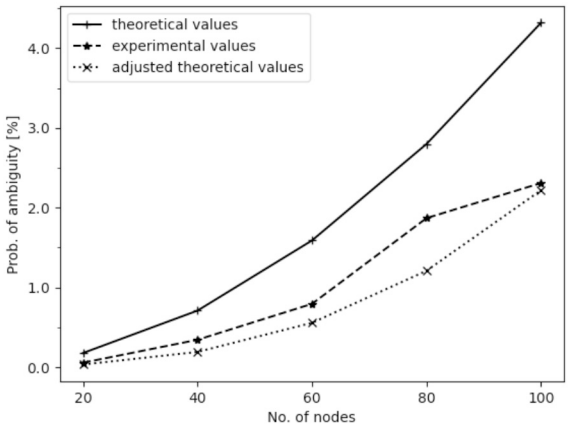


Fig. 11. Probability of ambiguities as predicted by theoretical model and as measured experimentally.

messages that have been transmitted by some device as well as messages that coincidentally arise from the pulses transmitted by multiple devices, this does not affect the evaluation of the probability of ambiguities. Each experiment is repeated three times and the measured values of the probability of ambiguity are averaged. Fig. 11 shows the experimental values as well as the theoretical values for the probabilities of ambiguity.

The experimental results are better than theoretically predicted. The reason for this may be that (13) implicitly assumes that all code words are equally likely to be used, but this does not happen in the experiment since the address space of $N_{c_a} = 127$ code words is only filled with a number of code words equaling the number N_n of senders. Similarly, each device sends only 30 messages on average, so $N_{c_d} = 127$ overestimates the number of code words used for the data frame. To compensate for this, we set $N_{c_a} = N_n$ and $N_{c_d} = 30$ and plot the results in the graph of Fig. 11 as *adjusted theoretical values*. These values lie much closer to the ones measured in our experiment, though they may under-estimate them somewhat, since the data frames of the messages transmitted by the different devices do not come all from the same pool of 30 code words.

The theoretical model for the probability of ambiguities of 4-pulse APCMA codes corresponds well with the measurements obtained from experiments. Experimental validation of the probabilities of phantom messages is more difficult, because it requires that the exact timing of the broadcast of a message is verified, which would necessitate a more complex experimental setup. Moreover, the probability of phantom messages is much lower than the probability of ambiguous messages, making it even more difficult to obtain an accurate estimate, given the low number (on average 30) of messages that each device broadcasts in one session of the experiment. Since our model shows that phantom messages are much less prevalent than ambiguous messages, we have focused on measuring the probabilities of only the latter by experiment.

We give a rough estimate of the energy consumption of the transmitters. According to the data available for the Fs1000A315 Tx module [52] the supply voltage is 3 to 12 V and the supply current is 28 mA when transmitting a logic high signal (no data on supply current is listed for logic low). We measured a supply current of 30.0 mA at a supply voltage of 5.13 V at logic high for the Tx module and 0.0 mA for logic low. Since the FPGA supplies power to the Tx module, we also investigated the FPGA including the Tx module, and measured 65.0 mA for logic high and 37.0 mA for logic low at 5.0 V. The difference of 28.0 mA in supply current to the FPGA appears to account for the current used by the Tx module. For the calculation of the energy consumption we use the values directly measured at the Tx module.

A pulse is represented by a logic high and its length is 1.25 ms, so the energy consumed by the Tx module to transmit a single pulse is estimated as 30.0 mA $\times$ 5.13 V $\times$ 1.25 ms $=$ 0.192 mJ. A message consists of 7 pulses, which represent 14 bits in this paper, so per transmitted bit the energy consumption is 0.096 mJ. Note that the number of encoded bits is independent of the number pulses, given a certain (fixed) number of frames per message. In other words, the per-bit energy consumption can be made arbitrarily small by increasing the number of bits encoded in a code word. However, this goes at the cost of throughput, since the average size of silent intervals will increase exponentially with the number of encoded bits.

Another way to reduce energy consumption is by reducing the widths of pulses. However, the Fs1000A315 being a low-cost Tx module, it does not allow pulses with widths of less than 1 ms. If the Linx TRM-315-LT transceiver, which we used in the receiver, is used as a Tx module in a transmitter, pulse widths in the μs range are possible, and its supply current for logic high is typically 12.0 mA according to its data sheet [53], but for logic low it is 4.0 mA. This means that it consumes energy during silent intervals, unlike the Fs1000A315 Tx module. To achieve low energy consumption in APCMA and other CtS-like schemes, a Tx module needs to be able to transmit short pulses, like the Linx TRM-315-LT,

while consuming virtually zero energy when transmitting a silence, like the Fs1000a315.

VI. Discussion and Conclusion

The APCMA-based protocol described in this paper employs messages with low sparsity of pulses, which allows not only a low energy consumption, but also low duty cycles. While duty cycles are usually defined in terms of the degree as to which a channel is occupied by broadcasts of messages, in the case of APCMA this may be redefined as the degree as to which a channel is occupied by transmitted pulses, because, unlike in many other schemes, messages may overlap in APCMA, so it is only the individual pulses that count towards occupation of a channel. This in itself makes the duty cycle of APCMA-encoded messages significantly lower than in other schemes, because the duty cycle then becomes the ratio of a low number of pulses used to encode a message on one hand and a large number of time slots in one cycle (5) of the scheduling on the other hand.

For the parameter settings in our experiment this translates into a duty cycle of approximately $7/(520 \times 100) = 0.014\%$, where 7 pulses per message are used. In this calculation an effective code length of $B = 520$ time slots is assumed, whereas the factor 100 corresponds to the average length of one Transmit-Sleep cycle in terms of the effective code length. Many platforms in Low Power Wide-Area (LPWA) networks are bound by maximum duty cycles of 1% or 0.1%, as a result of which they can transmit only a limited number of packets in 24 hours. Sigfox, for example, has a maximum packet payload of 12 bytes under a duty cycle of 1%, whereby the number of packets per device cannot exceed 140 packets per day [9]. This gives an average bit rate measured over 24 hours of less than 0.16 bps. LoRaWAN does better in this respect, with typical values ranging between 0.1 and tens of bps [7], [8], [10], [54], but it strongly depends on spreading factor, packet load, the class of the end devices, and the number of devices. The bit rate of each device in our experiment is 30 messages $\times$ 14 bits/2000 s $= 0.21$ bps. Our hardware is not optimized, however, and we expect to be able to achieve at least a factor 100 higher if Tx modules with pulse widths in the μs scale are used.

The theoretical model in this paper shows that misinterpretation probabilities are significantly reduced with every single pulse that is added to the encoding according to APCMA. The question whether codes with more than four pulses per code word actually exist can be answered affirmatively, but their design is much less trivial than designs of 4-pulse codes like the ones in Fig. 3. Similarly, decoding algorithms are more complex for codes with more than four pulses. We will report in more detail on such codes in subsequent papers.

The proposed protocol allows a simple organization into different channels, as the preliminary version of this paper [14] shows. While all devices operate on a single 315 MHz band, they can be easily divided into separate groups by adjusting the code length. So, rather than relying on differences of frequencies to create different channels, the protocol relies on a different parameter setting of the encoding scheme, thus allowing for a great degree of flexibility. This mechanism has not been tested in the current paper, since we prioritize the validation of the APCMA performance model through experiments. In the current paper all devices in the experiment use the same code length, and thus operate on the same channel, but it is expected that the theoretical model accurately predicts the misinterpretation probabilities of devices operating on different channels, since there is no fundamental difference if a channel differentiation is made through the code length.

How sensitive is the APCMA-based protocol to interference from external wireless sources? When pulses with similar time slot widths are transmitted on the 315 MHz band by an external wireless source, we expect that the analysis in Section IV applies, so interference will depend on the density of pulses. Interference from external sources on the 315 MHz band employing different protocols has not been tested in our experiments, but we expect it to be significant if the duty cycles of such sources are high, because a single baseband frequency is used in this paper, which is less resilient than the spread spectrum chirping signals used by LoRa and the triple frequencies used by Sigfox. The performance of the pattern recognizer in the receiver, however, plays a significant role: if it can be designed to recognize pulses reliably, interference from other wireless signals will be limited to a great extent.

Since the goal of this paper is to study the feasibility of APCMA, the hardware used in this paper is designed for prototyping. It is not yet suitable for implementation on off-the-shelf commercial hardware, since such hardware tends to accept only protocols abiding by certain standards. However, ultimately we aim to commercialize it into low-cost IoT devices that are employable on massive scales. One important condition is that the use of CSMA is not prescribed, because that would defy the reason for using APCMA in the first place. We are currently investigating strategies to increase robustness to errors through improved coding so that higher pulse densities and throughput can be achieved. We are also considering implementations on sub-GHz bands, used by LoRa and Sigfox, as well as implementations of impulse radio on ultra-wide bands (UWB). The latter will be useful for indoor IoT applications requiring higher data rates.

References

[1] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of Things: Vision, applications and research challenges," *Ad Hoc Netw.*, vol. 10, no. 7, pp. 1497–1516, 2012.

[2] F. Peper, K. Leibnitz, J. Teramae, T. Shimokawa, and N. Wakamiya, "Low-complexity nanosensor networking through spike-encoded signaling," *IEEE Internet Things J.*, vol. 3, no. 1, pp. 49–58, Feb. 2016.

[3] Z. Memish *et al.*, "Mass gatherings medicine: Public health issues arising from mass gathering religious and sporting events," *Lancet*, vol. 393, no. 10185, pp. 2073–2084, 2019.

[4] N. Morimura *et al.*, "Medicine at mass gatherings: Current progress of preparedness of emergency medical services and disaster medical response during 2020 Tokyo Olympic and Paralympic Games from the perspective of the Academic Consortium (AC2020)," *Acute Med. Surgery*, vol. 8, no. 1, 2021, Art. no. e626.

[5] K. Henning, "Overview of syndromic surveillance—What is syndromic surveillance?" *Morbidity Mortality Weekly Rep.*, vol. 53, pp. 5–11, Sep. 2004.

[6] A. Laya, C. Kalalas, F. Vazquez-Gallego, L. Alonso, and J. Alonso-Zarate, "Goodbye, ALOHA!" *IEEE Access*, vol. 4, pp. 2029–2044, 2016.

[7] A. Augustin, J. Yi, T. Clausen, and W. Townsley, "A study of LoRa: Long range & low power networks for the Internet of Things," *Sensors*, vol. 16, no. 9, p. 1466, Sep. 2016.

[8] A. Lavric, "LoRa (long-range) high-density sensors for Internet of Things," *J. Sensors*, vol. 2019, Feb. 2019, Art. no. 3502987.

[9] A. Lavric, A. I. Petrariu, and V. Popa, "Long range SigFox communication protocol scalability analysis under large-scale, high-density conditions," *IEEE Access*, vol. 7, pp. 35816–35825, 2019.

[10] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of LoRaWAN," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 34–40, Sep. 2017.

[11] Y. Zhu and R. Sivakumar, "Challenges: Communication through silence in wireless sensor networks," in *Proc. Annu. Int. Conf. Mobile Comput. Netw. (MOBICOM)*, Aug. 2005, pp. 140–147.

[12] D.-S. Shiu and J. M. Kahn, "Differential pulse-position modulation for power-efficient optical communication," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1201–1210, Aug. 1999.

[13] F. Peper, K. Leibnitz, M. Hasegawa, and N. Wakamiya, "Spike-based communication networks with error correcting capability," *Brain Neural Netw.*, vol. 25, no. 4, pp. 157–164, 2018.

[14] F. Peper *et al.*, "On high-density resource-restricted pulse-based IoT networks," in *Proc. IEEE Global Commun. Conf.*, 2020, pp. 1–6.

[15] B. Blaszczyszyn and B. Radunovic, "Using transmit-only sensors to reduce deployment cost of wireless sensor networks," in *Proc. 27th Conf. Comput. Commun.*, 2008, pp. 1202–1210.

[16] C. Huebner, R. Cardell-Oliver, S. Hanelt, T. Wagenknecht, and A. Monsalve, "Long-range wireless sensor networks with transmit-only nodes and software-defined receivers," *Wireless Commun. Mobile Comput.*, vol. 13, no. 17, pp. 1499–1510, 2013.

[17] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 3, pp. 443–461, 3rd Quart., 2011.

[18] B. K. Szymanski and G. G. Chen, "Computing with time: From neural networks to sensor networks," *Comput. J.*, vol. 51, no. 4, pp. 511–522, 2008.

[19] J. Hopfield, C. Brody, and S. Roweis, "Computing with action potentials," in *Advances in Neural Information Processing Systems*, vol. 10. Cambridge, MA, USA: MIT Press, 1998, pp. 166–172.

[20] A. Dhulipala, C. Fragouli, and A. Orlitsky, "Silence-based communication," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 350–366, Jan. 2010.

[21] K. Sinha, B. Sinha, and D. Datta, "CNS: A new energy efficient transmission scheme for wireless sensor networks," *Wireless Netw.*, vol. 16, no. 8, pp. 2087–2104, 2010.

[22] K. Sinha, B. P. Sinha, and D. Datta, "An energy-efficient communication scheme for wireless networks: A redundant radix-based approach," *IEEE Trans. Wireless Commun.*, vol. 10, no. 2, pp. 550–559, Feb. 2011.

[23] K. Sinha, "A new energy efficient MAC protocol based on redundant radix for wireless networks," 2016. [Online]. Available: arXiv:1606.04935.

[24] A. Bhattacharya, P. Majumder, K. Sinha, B. Sinha, and K. Kavitha, "An energy-efficient wireless communication scheme using Quint Fibonacci number system," *Int. J. Commun. Netw. Distrib. Syst.*, vol. 16, no. 2, pp. 140–161, 2016.

[25] A. Bhattacharya, K. Sinha, D. Datta, and B. Sinha, "MRBNS: A new energy-efficient communication scheme in low power wireless networks," *Int. J. Sens. Netw.*, vol. 23, no. 3, pp. 155–169, 2017.

[26] P. Majumder, L. Dash, K. Sinha, and B. Sinha, "CMNS: An energy-efficient communication scheme for wireless sensor networks," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, 2018, pp. 1–6.

[27] P. Majumder, K. Sinha, and B. Sinha, "DCV NS: A new energy efficient transmission scheme for wireless sensor networks," in *Proc. IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, 2018, pp. 1–5.

[28] R. Gallager, "Basic limits on protocol information in data communication networks," *IEEE Trans. Inf. Theory*, vol. 22, no. 4, pp. 385–398, Jul. 1976.

[29] V. Anantharam and S. Verdu, "Bits through queues," *IEEE Trans. Inf. Theory*, vol. 42, no. 1, pp. 4–18, Jan. 1996.

[30] A. S. Bedekar and M. Azizoglu, "The information-theoretic capacity of discrete-time queues," *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 446–461, Mar. 1998.

[31] W.-M. Hu, "Reducing timing channels with fuzzy time," *J. Comput. Security*, vol. 1, nos. 3–4, pp. 233–254, 1992.

[32] R. Pietro and G. Oligeri, "Silence is golden: Exploiting jamming and radio silence to communicate," *ACM Trans. Inf. Syst. Security*, vol. 17, no. 3, pp. 1–24, Mar. 2015.

[33] J. Giles and B. Hajek, "An information-theoretic and game-theoretic study of timing channels," *IEEE Trans. Inf. Theory*, vol. 48, no. 9, pp. 2455–2477, Sep. 2002.

[34] D. Feng, S. Das, and S. Biswas, "Packet position modulation: A technique for enhancing information transfer capacity in ultra-low duty cycle networks," in *Proc. 21st Int. Conf. Distrib. Comput. Netw.*, 2020, Art. no. 27.

[35] X. Fafoutis, E. Tsimbalo, and R. Piechocki, "Timing channels in Bluetooth low energy," *IEEE Commun. Lett.*, vol. 20, no. 8, pp. 1587–1590, Aug. 2016.

[36] X. Zhang and K. Shin, "Gap Sense: Lightweight coordination of heterogeneous wireless devices," in *Proc. IEEE INFOCOM*, 2013, pp. 3094–3101.

[37] K. Sanzgiri, I. Chakeres, and E. Belding-Royer, "The utility of perceptive communication between distant wireless nodes," in *Proc. 2nd Int. Conf. Testbeds Res. Infrastruct. Dev. Netw. Commun.(TRIDENTCOM).*, 2006, pp. 102–111.

[38] S. Karp and R. Gagliardi, "The design of a pulse-position modulated optical communication system," *IEEE Trans. Commun. Technol.*, vol. 17, no. 6, pp. 670–676, Dec. 1969.

[39] I. Garrett, "Pulse-position modulation for transmission over optical fibers with direct or heterodyne detection," *IEEE Trans. Commun.*, vol. 31, no. 4, pp. 518–527, Apr. 1983.

[40] M. Sato, M. Murata, and T. Namekawa, "A new optical communication system using the pulse interval and width modulated code," *IEEE Trans. Cable Television*, vol. CATV-4, no. 1, pp. 1–9, Jan. 1979.

[41] E. D. Kaluarachi, Z. Ghassemlooy, and B. Wilson, "Digital pulse interval modulation for optical free space communication links," in *Proc. IET Conf.*, Jan. 1996.

[42] H. Sugiyama and K. Nosu, "MPPM: A method for improving the band-utilization efficiency in optical PPM," *J. Lightw. Technol.*, vol. 7, no. 3, pp. 465–472, Mar. 1989.

[43] Y. Chen, D. Wang, and J. Zhang, "Variable-base tacit communication: A new energy efficient communication scheme for sensor networks," in *Proc. 1st Int. Conf. Integr. Internet Ad Hoc Sens. Netw.*, 2006, Art. no. 27.

[44] D. Feng, F. Hajiaghajani, S. Das, and S. Biswas, "Pulse position coded PDUs: A new approach to networking energy economy," in *Proc. 14th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, 2017, pp. 498–503.

[45] D. Feng, S. Das, F. Hajiaghajani, Y. Shi, and S. Biswas, "Pulse position coded medium access in energy-starved networks," *Comput. Commun.*, vol. 148, pp. 62–73, Dec. 2019.

[46] D. Feng, Y. Shi, S. Das, and S. Biswas, "Energy-efficient and secure data networking using chaotic pulse position coded PDUS," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 2, pp. 375–386, Jun. 2020.

[47] A. Pal, R. Gulati, and K. Kant, "Towards building low power magnetic communication protocols for challenging environments," in *Proc. 28th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2019, pp. 1–9.

[48] B. Krishnaswamy *et al.* "Time-elapse communication: Bacterial communication on a microfluidic chip," *IEEE Trans. Commun.*, vol. 61, no. 12, pp. 5139–5151, Dec. 2013.

[49] Y. Sangar and B. Krishnaswamy, "WiChronos: Energy-efficient modulation for long-range, large-scale wireless networks," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, Art. no. 21.

[50] C. Tanaka, F. Peper, and M. Hasegawa, "Application of APCMA protocol to power packet networks for multiplexing power packet transmissions," *IEICE Nonlinear Theory Appl.*, vol. 11, no. 4, pp. 433–445, 2020.

[51] C. Tanaka *et al.*, "Implementation of pulse-based multiplexing protocol for massive IoT," in *Proc. Int. Symp. Nonlinear Theory Appl. (NOLTA)*, 2020, pp. 346–349.

[52] (May 15, 2021). *Data of Fs1000A315 Transmitter (in Japanese)*. [Online]. Available: https://www.aitendo.com/product/16613

[53] (May 15, 2021). *Linx LT Series Transceiver Module Data Guide*. [Online]. Available: https://www.linxtechnologies.com/wp/wp-content/uploads/trm-fff-lt.pdf

[54] D. Bankov, E. Khorov, and A. Lyakhov, "On the limits of LoRaWAN channel access," in *Proc. Int. Conf. Eng. Telecommun. (EnT)*, 2016, pp. 10–14.

Ferdinand Peper received the M.Sc. degree in mathematics and the Ph.D. degree in Computer Science from Delft University of Technology, the Netherlands, in 1985 and 1989, respectively. In 1990, he joined the National Institute of Information and Communications Technology (NICT), Japan, (then named Communications Research Laboratory) as a Researcher and has been with NICT since then, currently as an Associate Director of the Neural Information Engineering Laboratory. His research interests include artificial neural networks, cellular automata, nanocomputer architectures, asynchronous systems, distributed computing, noise and fluctuations, wireless sensor networks, and error-correcting codes.

Kenji Leibnitz received the M.Sc. and Ph.D. degrees in information science from the University of Würzburg, Germany, in 2003. In 2004, he joined the research group of Prof. Masayuki Murata with Osaka University, where he became a Specially Appointed Associate Professor in 2006. In 2010, he moved to the National Institute of Information and Communications Technology (NICT) as a Senior Researcher and he has been a Principal Investigator with the Center for Information and Neural Networks, NICT and Osaka University since 2013. His research interests include modeling and performance analysis of communication networks, especially biologically and brain-inspired mechanisms for self-organization in future networks.

Chiemi Tanaka received the B.Eng. and M.Eng. degrees from the Department of Electrical Engineering, Tokyo University of Science, Japan, in 2019 and 2021, respectively. She has recently moved to KDDI Ltd., Japan, Japan. Her research interest includes implementation of asynchronous pulse-code multiple access protocol to power packet networks and wireless communication systems.

Kentaro Honda received the B.Eng. degree from the Department of Electrical Engineering, Tokyo University of Science in 2021. His research focuses on the implementation of asynchronous pulse-code multiple access protocol for massive IoT and its performance evaluation.

Mikio Hasegawa (Member, IEEE) received the B.Eng., M.Eng., and Dr.Eng. degrees from Tokyo University of Science, Tokyo, Japan, in 1995, 1997, and 2000, respectively. From 1997 to 2000, he was a Research Fellow with the Japan Society for the Promotion of Science. From 2000 to 2007, he was with the Communications Research Laboratory, Ministry of Posts and Telecommunications, which was reorganized as the National Institute of Information and Communications Technology in 2004. He is currently a Professor with the Department of Electrical Engineering, Faculty of Engineering, Tokyo University of Science. His research interests include mobile networks, cognitive radio networks, chaos, neural networks, machine learning, and optimization techniques.

Konstantinos Theofilis received the B.Sc. and M.Sc. degrees from Aston University, U.K., and the Ph.D. degree from the University of Hertfordshire, U.K. He is currently a Researcher with the National Institute of Information and Communications Technology, Japan. His research interests include cognitive robotics and AI.

Aohan Li (Member, IEEE) received the Ph.D. degree from Keio University, Yokohama, Japan, in March 2020. Since April 2020, she has been an Assistant Professor with Tokyo University of Science, Tokyo, Japan. Her current research interests include machine learning, resource management, and Internet of Things. She was recipient of the 9th International Conference on Communications and Networking in China 2014 Best Paper Award, and the 3rd International Conference on Artificial Intelligence in Information and Communication Excellent Paper Award in 2021. She is a member of IEICE.

Naoki Wakamiya (Member, IEEE) received the M.E. and Ph.D. degrees from Osaka University, Osaka, Japan, in 1994 and 1996, respectively. He has been a Research Associate with the Graduate School of Engineering Science and the Educational Center for Information Processing, an Assistant Professor with the Graduate School of Engineering Science, and an Associate Professor with the Graduate School of Information Science and Technology, Osaka University, where he became a Professor in 2011. His research interests include biologically and brain inspired information and communication technology and self-organizing network control.