Performance evaluation of pulse-based multiplexing protocol implemented on massive IoT devices

Chiemi Tanaka¹, Kentaro Honda¹, Aohan Li¹, Ferdinand Peper², Kenji Leibnitz², Konstantinos Theofilis², Naoki Wakamiya²,³, and Mikio Hasegawa¹a)

¹ Department of Electrical Engineering, Tokyo University of Science
6-3-1 Niiyuku, Katsushika-ku, Tokyo 125-8585, Japan
² Center for Information and Neural Networks (CiNet), NICT
1-4 Yamadaoka, Suita, Osaka 565-0871, Japan
³ Graduate School of Information Science and Technology, Osaka University
Yamadaoka, Suita, Osaka 565-0871, Japan
a) hasegawa@ee.kagu.tus.ac.jp

Received March 26, 2021; Revised June 22, 2021; Published October 1, 2021

Abstract: An Internet of Things (IoT) employing resource-restricted (e.g., battery-limited) IoT devices at high densities will require an effective multiplexing protocol that can be implemented with low energy consumption, while being able to effectively avoid collisions between the transmissions from multiple IoT devices. Asynchronous Pulse-Code Multiple Access (APCMA) has been proposed as a communication protocol based on pulse-encoded signals that allows multiple senders to simultaneously transmit their messages. Even if multiple messages on the same band overlap in time during transmission, they are disentangled at the receiver by a decoding algorithm that is based on pattern matching of pulse trains. In this paper, we implement the APCMA protocol on an FPGA and evaluate its performance in wireless communication. We also implement a decoding algorithm that matches pulses by logical operations on a shift register. Our experiments show that the APCMA protocol can realize multiplexing with low overhead and that the error rate caused by misdetection during decoding is reduced with longer pulse trains.

Key Words: communication protocols, information networks, multiplexing, Asynchronous pulse-code multiple access, massive IoT

1. Introduction
In recent years, wireless sensor networks have been used in many applications [1] and the number of Internet of Things (IoT) devices has increased dramatically. International Data Corporation (IDC)
expects to have more than 41.6 billion IoT devices deployed by 2025, generating 79.4 zettabytes (ZB) of data [2]. This is likely to give high densities of devices, even on local scales, which will increase the likelihood of packet collisions on the used wireless channel. To avoid such collisions, different protocols have been proposed on the link layer, for instance, centralized channel control type algorithms [3] or distributed random access protocols like carrier-sense multiple access with collision avoidance (CSMA/CA) [4]. However, although these technologies are well-established, their overheads are high and they tend to increase power consumption. This makes it difficult to implement such protocols on resource-restricted IoT devices. Therefore, it is necessary to design new IoT protocols for resource-restricted systems with low overhead and without synchronization requirements.

The pulse-based APCMA communication protocol [5] is suitable for such resource-restricted scenarios (Fig. 1). APCMA encodes data as a train of pulses, in which the (relatively long) silent intervals between those pulses [6] are based on predetermined functions. Though other schemes also use pulses for communication, the APCMA protocol makes it possible to multiplex pulse-based frames. Since multiple senders can transmit pulse trains at arbitrary times in APCMA, a receiver typically receives a superposition of pulse trains. A decoding algorithm then disentangles the mixed signals with a high probability of success, which makes APCMA suitable for data multiplexing in communication systems without using carrier sense, centralized channel control-type algorithms, or CSMA/CA. Hence, APCMA can be implemented with low energy consumption, while being able to effectively avoid collisions between the transmissions from IoT devices.

![Fig. 1. Encoding and Decoding scheme of APCMA.](image)

A first implementation of APCMA already exists in wired communication [5, 8] and in wireless communication [7]. Decoding in [5, 7, 8] is based on the recognition of pulse patterns by automata that are created and deleted during the decoding process. APCMA is implemented in [7] on the MAC sublayer and experiments are conducted with an expansion board developed for an Arduino Mega 2560 Rev 3 microcontroller that includes a 315 MHz baseband transceiver (Linx TRM-315-LT). The experiments show few conflicts between messages that are broadcast on the same band overlapped in time. Theoretical analysis in [7] confirms that the probability of conflicts is low, even if the number of nodes increases to ten thousand.

In [8], APCMA is implemented in Power Packet Networks (PPN) [9–11]. In PPN, multiple power sources and destinations are connected to each other via power packet routers. A power packet is a packetized power unit whereby the information (e.g., destination address) is added to the voltage waveform. In [10, 11], the PPN Protocol was implemented using semiconductor power devices, such as GaN. In PPN a collision may occur if multiple power packets are transmitted simultaneously from multiple power sources. In general, a power packet router is a simple device in which input/output is one-way while synchronization is difficult. So it is difficult to implement conventional multiplexing protocols on PPN devices. Furthermore, for real-time system control, such as in Electric Vehicles (EVs) and robots, power transmission delay will impact system performance. However, a conventional CSMA-based protocol has a large delay in a low bit-rate power packet system like PPN. The APCMA-based power packet system makes it possible to correctly and simultaneously transmit power packets from multiple power sources to different destinations with low latency [8].

In a massive IoT environment (Fig. 2), it is necessary to have a simple multiplexing protocol that can
deal with multiple simultaneous transmissions, and APCMA is suitable for this purpose. Typically, the gateways in a massive IoT environment are much more limited in number than the IoT devices, so even if the gateways are always on to decode the packets received from the IoT devices, overhead of the system as a whole is limited.

In this paper, we evaluate the performance of the APCMA protocol in wireless communications by real device experiments using an APCMA-based IoT system. We implement the APCMA protocol based on a shift-register on the FPGA. The use of a shift register allows for more efficient implementations, as compared to the automaton-based pattern matching employed in [5, 7, 8].

This paper is organized as follows. Section 2 describes APCMA and its coding and decoding in more detail. This is followed by Section 3, which describes the hardware and demonstrates the decoding process. Two experiments are carried out in Section 4. One uses different code lengths for various small numbers of transmitters and one receiver. The other experiment fixes the code length, while the number of transmitters also varies, but it is much larger, going up to 100. This paper finishes with conclusions in Section 5.

2. Asynchronous pulse-code multiple access (APCMA)

2.1 APCMA encoding scheme on pulse trains

APCMA encodes information as a pulse train that has (relatively long) silent intervals between pulses. In [5] information $x$ is represented by four pulses and three intervals of length $x$, $x$, and $C - 2x$, respectively, where the total length of each code word is a constant $C$.

A more general formulation of the APCMA pulse code is shown in Fig. 3, whereby the number of pulses in a code word is $m$, the total length of the pulse train is $C$, and $P_1, P_2, \ldots, P_m$ are pulses, and $m$ and $C$ are integer constants. The time intervals in this general code are $f_1(x), f_2(x), \ldots, f_{m-2}(x)$ and $C - \sum_{n=1}^{m-2} f_n(x)$, whereby the functions $f_1, \ldots, f_{m-2}$ are one-to-one, and are predetermined and shared among the transmitters and receivers. The functional relationships between these functions are used to give the code its robustness to misdetection by a decoder in case mixtures of different code words occur. Since large numbers of IoT devices can transmit at the same time, the combined
communication capacity of APCMA becomes high compared to other schemes, especially in dense networks.

The intervals of the 4-pulse code used in this paper are \( x, C - 2x \) and \( x \) in sequence, and the code is referred to by the notation \( x/C - 2x/x \). This is a simple case of an APCMA code in which the parameters are \( m = 4 \), \( f_1(x) = x \) and \( f_2(x) = C - 2x \), as in [7].

### 2.2 APCMA decoding scheme

In the APCMA protocol, even if there are collisions and overlaps of some of the received frames, it is possible to separate them with a high probability, when the values of \( C \) and the sleep times between successive transmissions are chosen sufficiently large [7]. We show an example of the decoding of data whereby two pulse trains encoded according to the scheme \( x/C - 2x/x \) are overlapping. There are two transmitters and one receiver as shown in Fig. 4. Figure 5 shows the decoding process. In the received data two pulse trains \( x_1 \) and \( x_2 \) become interleaved, but APCMA correctly extracts the pulse trains, thus decoding the original transmitted data.

**Fig. 4.** Example of setup with one receiver receiving information from two transmitters.

**Fig. 5.** APCMA decoding.

We note that in some cases a set of pulses that is not sent by any transmitters can match a code word due to multiple pulse trains coincidentally overlapping. This leads to misdetection, degrading the performance of the APCMA decoding process. An example of misdetection is shown in Fig. 6, if data \( x_2 \) is transmitted a time interval \( d \) after data \( x_1 \). The receiver incorrectly decodes a message as being \( x = d \) if \( x_2 = 2d \) because of the combination of pulses marked with orange dotted lines.

There are several ways to implement the APCMA decoder. The first, implemented in [5, 7], and [8], is based on the recognition of pulse patterns by automata that are created and deleted during the decoding process. The second implementation extracts the matching pulses by processing the received pulse trains in a shift register. When a pulse train matches a predefined pulse code in the shift register, the corresponding data will be decoded. The second implementation is easier to realize on an FPGA than the first, and accordingly will be used in this paper.
3. Implementation

3.1 Hardware

The pulse code used in this experiment is defined by \( m = 4 \), \( f_1(x) = x \), and \( f_2(x) = C - 2x \). In theory, a pulse has a negligible small width in time, but in practice it has a finite non-zero width. Figure 7 shows the pulse code \( x/C - 2x/x \) assuming a finite non-zero pulse width. We consider \( C\tau \) as the length of the interval between the rising edge of the first pulse and the rising edge of the final pulse, whereby \( \tau \) is the pulse width. Similarly, the lengths of the intervals between the rising edges of the pulses \( P_1 \), \( P_2 \) and \( P_3 \) and the rising edges of their respective next pulses are \( x\tau \), \( (C - 2x)\tau \) and \( x\tau \).

Figure 8 shows a block diagram of the system. Each transmitter consists of a controller (FPGA) and a Tx module (CDT-88 or Fs1000A), while the receiver consists of an Rx module (TRM-315-LT) and the same FPGA as controller (Xilinx Spartan-3E). Figure 9 shows photographs of the transmitter and receiver, and Fig. 10 shows a system with three transmitters and one receiver. The transmitter generates a pulse train by on-off-keying, whereby each pulse is a time-limited sinusoid with baseband 315 MHz. The Rx module on the receiver receives the transmitted pulse train and inputs it to the FPGA module. The FPGA on the receiver is oversampled with a frequency that is 5 times \( 1/\tau \). The nodes are placed in a Micronix MY1530 shielded box.

We assume that overlapping pulses do not cancel each other out. This assumption is reasonable since cancellation only takes place with sinusoidal signals when two pulses are at an approximate 180° phase difference from each other. We show an example of overlapping pulses in Fig. 11. The yellow and blue lines show the input to each Tx module from each FPGA on the transmitter side. The pink line shows the on-off-keying signals. The green line shows the output of the transmitter’s data pin on the receiver (received pulse train). In Fig. 11, the second yellow pulse and the first blue pulse...
are overlapping, and the green line shows that both pulses are detected. Overlapping pulses are not a problem, even if they are detected as a single pulse, because such a single pulse is interpreted by the decoder as being part of multiple messages. Each of these messages can be correctly interpreted because the underlying pattern of pulses is not disturbed by overlaps of pulses. Note that we use on-off keying modulation and demodulation in our implementation, which represents a series of logical ‘1’s and ‘0’s by simply switching on and off the carrier. The radio signals modulated by on-off-keying
are interpreted by detecting the presence of the carrier wave.

We use shift registers to check whether received pulse combinations match code words (Fig. 12). The received data is input to the left of a shift register, whereby the number of Flip-Flops (FFs) in the register is the same as the value of $C$. The outputs from the register’s FFs are connected to an array of AND-gates, each representing its corresponding pulse code word, which allows pulse trains matching the code words to be extracted. The FPGA module in the receiver outputs data that it successfully decoded to a Raspberry Pi, which then checks whether the detected data is correct.

**3.2 Verification of decoded APCMA signals**

Figure 13 shows an example of decoding in which the pulse trains of A, B, and C are overlapped. In this case, the total code length $C$ is 63 and the pulse width $\tau$ is 1 ms. The messages A, B, and C represent the data values $x = 3$, 11 and 21, respectively. Figure 13a shows the received pulse trains, and Figs. 13b, 13c and 13d show the outputs of the receiver, whereby the receiver generates one pulse for each successful decoding. These pulses are the outputs of the AND-gates in the logic circuit that correspond to the recognized code words for $x = 3$, 11, and 21, respectively. We see that each message is decoded successfully from the received pulse data. This decoding process is quite straightforward, and the resulting transmission of data requires neither carrier sense nor synchronization between transmitters and receivers. This suggests APCMA has a low overhead, compared to CSMA/CA and
Fig. 13. Time series of input and output at the receiver with APCMA decoding functionality.

4. Experiments

4.1 Performance evaluation

In our experiments, there is no obstacle between the transmitter and receiver. All sent and received messages are classified according to Fig. 14, whereby $m_s$ is the number of messages sent, $m_d$ is the number of messages decoded, and $m_r$ is the number of messages successfully decoded. The decoded
messages are those messages that can be interpreted as valid code words, even if they have not been transmitted, i.e., $m_d$ contains the correct messages as well as misinterpretations. The Misdetection Probability is then defined by Eq. (1).

$$\text{Misdetection Probability}[^\%] = \frac{m_d - m_r}{m_d} \times 100$$

(1)

4.2 Evaluation for various lengths $C$ of pulse trains

In the first experiment, each transmitter is assumed to act as a beacon, i.e., each transmitter transmits its address as value $x$. We evaluate the misdetection probability for various numbers of transmitters and values of $C$. Table I shows the parameters used in this evaluation. The number of transmitters is varied from 1 to 6, and the value of $C$ is varied over 31, 63, 127, and 255. The sleep time (i.e., the time during which transmitters do not transmit data) is set between $C$ and $2C$. The pulse width is 1 ms. In this experiment, each transmitter transmits at least 100 frames.

| Parameter                  | value          |
|----------------------------|----------------|
| Number of transmitters     | 1 ~ 6          |
| Number of receivers        | 1              |
| Distance [cm]              | 15 (average)   |
| Code length $C$ [$\tau$]   | 31, 63, 127, 255 |
| Sleep time [\tau]         | $C \sim 2C$    |
| Pulse width $\tau$ [ms]    | 1              |

Table I. Parameter settings for experiment with various message lengths.

Figure 15 shows the experimental results of the misdetection probability. From Fig. 15, we can see that larger values of $C$ reduce the misdetection probability to the extent that the misdetection probability is less than 1% when the value of $C$ is 127 and 255. This suggests that, when the number of transmitters is small, the...
of transmitters is large, the value of $C$ must be set large as well, because larger values of $C$ can reduce the misdetection probability.

4.3 Evaluation of pulse code that includes transmitter address and data

Since frames containing merely addresses, like in the previous section, have limited use in communication, we investigate the more general case where each message consists of both an address part and a data part in this section. We design a pulse code in which a frame is formatted as an address header followed by data, as in Fig. 16, whereby $a$ is the address of the transmitter and $x$ is data. The length of the pulse code expressing the address is $C_a$, and the length of the pulse code expressing data is $C_d$, so $C_a + C_d = C$. If we would use a pulse code in which both the address and data are encoded in a single frame $x/C - 2x/x$, we would be able to reduce the number of pulses from 7 to 4, but this would increase the value of $C$. This would reduce the pulse density, but the size of the shift register would need to be much larger, and throughput would be reduced. For this reason the data format in Fig. 16 has merit, even though it requires more pulses.

![Fig. 16. Pulse code word with address of transmitter and data.](image)

The sender transmits pseudo-random data based on maximum length sequences (M-sequences) it generates. By comparing the transmitted messages with the decoded messages, we check whether a misinterpretation or loss of a message has taken place. We evaluate the misdetection probability for various numbers of transmitters, as well as the probability that frames get lost during transmission. The misdetection probability is as before in Eq. (1) and the loss probability is defined by Eq. (2) in this experiment.

\[
\text{LossProbability}\% = \frac{m_s - m_r}{m_s} \times 100
\]  

(2)

The loss probability includes both the probabilities that messages cannot be detected by the receiver, as well as the probability that messages are lost due to limitations on the data transfer from the FPGA on the receiver side to the Raspberry Pi. These losses of messages are caused by the low quality of the transmitters and the fact that two or more data items cannot be sent from the FPGA to the Raspberry Pi simultaneously in this experiment. In practical applications the loss probability will be less than what we measure here, in case there are no or less limitations on data transfer.

The parameters are shown in Table II. The number of transmitters varies from 10 to 100 in steps of 10, and the value of $C$ is 520. The sleep time is fixed to values between $20C$ and $30C$ for each transmitter. The pulse width is 1.25 ms, and each transmitter transmits at least 100 frames. From

| Table II. Parameters for experiment with frames that include address and data. |
|--------------------------|--------------------------|
| Number of transmitters   | 10 ~ 100                 |
| Number of receivers      | 1                        |
| Distance [cm]            | 20 (average)             |
| Address [bits]           | 7                        |
| Data [bits]              | 7                        |
| Total code length $C$ [τ] | 520                      |
| Address code length $C_{ad}$ [τ] | 260                  |
| Data code length $C_d$ [τ]    | 260                      |
| Sleep time [τ]           | $20C$ ~ $30C$             |
| Pulse width $τ$ [ms]     | 1.25                     |

735
Fig. 17, we see that the misdetection probability is 12.5% and the loss probability is 21.5% when the number of transmitters is 100. We also see that the more transmitters there are, the higher the loss probability becomes. One reason for this is that in the communication between the FPGA on the receiver side and the Raspberry Pi the loss increases, because the probability that multiple messages are transmitted simultaneously increases with an increase in the number of transmitters. In this experiment pulse loss also occurs because of the low received signal strength. Improved quality of the Tx and Rx hardware is expected to ameliorate this.

![Graphs showing misdetection and loss probability](image-url)

(a) Misdetection Probability.

(b) Loss Probability.

Fig. 17. Evaluation with pulse code containing both address and data.

5. Conclusions

Previous work has identified that there is an overhead in conventional multiplexing methods. In this paper, we implement the APCMA protocol based on a shift-register on IoT devices. We evaluated the misdetection probability and the loss probability under varying numbers of transmitters from 1 to 6 and various lengths of code words in the first experiment, and from 10 to 100 transmitters with a fixed code length in the second experiment. The results of the first experiment shows that the misdetection probability tends to grow with an increase in the number of transmitters, but it is reduced when the code length increases. In the second experiment the misdetection probability and the loss probability both grow with an increase in the number of transmitters up to an misdetection probability of 12.5% and a loss probability of 21.5%, when the number of transmitters is 100.

Our future challenge is to improve the APCMA protocol to have less misdetections and a higher delivery probability of correct messages. To this end, the simple pulse code used in this paper will be extended to more pulses, to improve the coding scheme. In the physical layer we plan to use a LoRa-like modulation with baseband 920 MHz instead of on-off-keying. Finally, for a more detailed evaluation of the proposed method in a massive IoT environment, we will use more IoT devices (around five hundred devices) and evaluate the misdetection probability to further clarify the effectiveness of the APCMA protocol.

Acknowledgments

This research and development work is supported by MIC/SCOPE JP205007001.

References

[1] B. Rashid and M.H. Rehmani, “Applications of wireless sensor networks for urban areas: A survey,” J. Netw. Comput. Appl., vol. 60, pp. 192–219, January 2016.
[2] International Data Corporation (IDC), IDC Media Center, Available on: https://www.idc.com/getdoc.jsp?containerID=prUS45213219 (19 November 2019).
[3] IEEE, 802.15.4e-2012-IEEE Standard for Local and metropolitan area networks-Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer, 2012.
[4] C. Tang, L. Song, J. Balasubramani, S. Wu, S. Biaz, Q. Yang, and H. Wang, “Comparative investigation on CSMA/CA-based opportunistic random access for internet of things,” IEEE Internet of Things Journal, vol. 1, no. 2, pp. 171–179, 2014.

[5] F. Peper, K. Leibnitz, M. Hasegawa, and N. Wakamiya, “Spike-based communication networks with error correcting capability,” The Brain & Neural Networks, vol. 25, no. 4, pp. 167–164, 2018.

[6] Y. Zhu and R. Sivakumar, “Challenges: Communication through silence in wireless sensor networks,” In: Proceedings of the 11th Annual International Conference on Mobile Computing and Networking, MobiCom’05, NewYork, NY, USA, ACM, pp. 140–147, 2005.

[7] F. Peper, K. Leibnitz, K. Theofilis, M. Hasegawa, N. Wakamiya, C. Tanaka, J. Teramae, S. Sekizawa, and A. Li, “On high-density resource-restricted pulse-based IoT networks,” Globecom 2020.

[8] C. Tanaka, F. Peper, and M. Hasegawa, “Application of APCMA protocol to power packet networks for multiplexing power packet transmissions,” NOLTA, vol. 11, no. 4, pp. 443–445, 2020.

[9] J. Toyoda and H. Saitoh, “Proposal of an open-electric-energy-network (OEEN) to realize cooperative operations of IOU and IPP,” Proc. 1998 Int. Conf. Energy Manage. Power Del. (EMPD), vol. 1, 1998.

[10] R. Takahashi, K. Tashiro, and T. Hikihara, “Router for power packet distribution network: Design and experimental verification,” IEEE Trans. Smart Grid, vol. 6, no. 2, pp. 618–626, 2015.

[11] T. Matsuda, J. Ma, and M. Hasegawa, “Design and implementation of a power packet network protocol for flexible power routing,” NOLTA, vol. 10, no. 4, pp. 455–464, 2019.