CU-MAC: A Duty-Cycle MAC Protocol for Internet of Things in Wireless Sensor Networks

Tanapoom Danmanee ,
Kulit Na Nakorn , and **Kultida Rojviboonchai***, Non-members

ABSTRACT

Nowadays "Internet of Things" or IoT becomes the most popular technology in the Internet system. Types of devices and sensors have been connected as a network of devices and sensors. While a wireless sensor network is a traditional network of sensors that can be considered as a beginning point of IoT systems. Currently, these sensor data are not only exchanged within a local network but also are delivered to other devices in the Internet. Consequently, well-known organizations such as IEEE, IETF, ITU-T and ISO/IET are trying to set standards for wireless sensor devices in IoT systems. The recommended standard utilizes many of internet stack standards such as CoAP, UDP and IP. However, the traditional design of WSNs is to avoid using internet protocol in the system to reduce transmission overhead and power consumption due to resource limitation. Fortunately, the current technology in both hardware and software allow the internet standard to sufficiently operate in a small sensor. In this paper, we propose a MAC protocol named CU-MAC to efficiently support IoT standard that need request-respond communication or bi-direction communication. CU-MAC uses multi-channel communication to perform continuous and bidirectional data transfer at low duty-cycle. It also has a mechanism to overcome the hidden terminal problem. We evaluated the performance of CU-MAC on both simulation and real testbed based on Contiki OS. The result shows that CU-MAC outperforms other existing MAC protocols in term of packet delivery ratio at 98.7% and requires lower duty-cycle than others to operate in the high traffic environment.

Keywords: Media Access Control Protocol, MAC layer protocol, Wireless Sensor Networks (WSNs), Internet of Things (IoT)

1. INTRODUCTION

Internet of Thing (IoT) is one of the most interested topic for researchers and developers. IoT allows things can communicate directly with each other, such as lights on streets, on-board sensors in vehicles, sensors in medical devices or even regular electronic devices in our daily life [1]. When the things are capable of exchanging their data among same devices or different ones, many smart applications have arisen. For example, fire alarm detecting sensors can tell the area that are on fire using information from nearby sensors or when an accident happens on the road, on-board sensors in the vehicle warn the other cars using the vehicular network or using street light networks. Therefore, the most important part in IoT development is how to manage the network of enormous number of incoming connected devices.

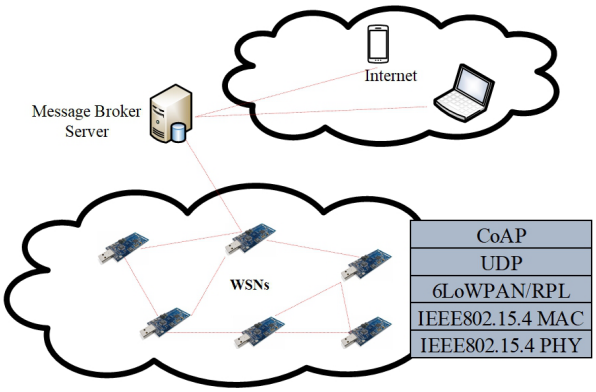

Fig. 1: *A diagram of wireless sensor network standards for IoT.*

In order to achieve the efficient communication for IoT, the traditional wireless sensor network need to be considered. WSNs share the same concept and the same requirement found in the IoT device such as small device size, small data size and low power consumption. WSNs have its own special medium access control (MAC) standard that can reduce the overall power consumption from data transmission. Because each transmission consumes far more power than other activities. Unfortunately, the WSNs are designed and optimized to operate in much lower capability environment than current IoT devices have. Nowadays, some critical applications, such as fire alarm system, medical monitoring system or on-board sensors in the vehicle need high frequent and reliable data transmission. As a result, any loss or stale information can cause seriously damage for such mentioned applications. To take the advantage of WSNs concept in IoT or let devices in WSNs successfully co-

Manuscript received on April 9, 2018 ; revised on June 13, 2018.

The authors are with Chulalongkorn University Big Data Analytics and IoT Center (CUBIC), Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand , Correspondence should be addressed to Kultida Rojviboonchai *E-mail: kultida.r@chula.ac.th.

operate with other devices in IoT, its communication structure improvement is mandatory.

IEEE, IETF, ITU-T and ISO/IET recommendations communication standard for IoT that are worth to mention, such as constrained application protocol (CoAP) and message queue telemetry transport (MQTT) for publish-subscribe messaging pattern, IPv6 over low power wireless personal area networks (6LoWPAN) and IPv6 routing protocol for low-power and lossy networks (RPL)[2]. These standards normally have not been applied in WSNs due to request/response approach which effect to power consumption. This also mean that they are not suitable with the traditional MAC layer protocols in WSNs. But some protocols of WSNs can performance in IoT environment with limitations. WSNs have two types of MAC protocols; synchronous MAC protocol and asynchronous MAC protocol. The asynchronous MAC protocol is the most famous types used in WSNs. It was reported in many publications [3-5] that it provides more bandwidth with less power consumption than synchronous type. The asynchronous MAC protocol can be classified into receiver initiate connection (RI) and sender initiate connection (SI). RI has less power consumption than SI but its drawbacks are the performance in term of delay and bandwidth worse than SI. X-MAC [6] is the most famous SI protocol that is applied to IoT network. X-MAC is a preamble-sampling protocol based on the low power listening mechanism. Although this can minimize the power consumption, loss and delay significantly increase from black spot issue when numbers of data transmission occur [7]. Therefore, the performance of the traditional SI protocol is still not enough for IoT network stack that includes many internet standards.

In this paper, we propose a CU-MAC protocol for IoT network to overcome those limitations in the traditional MAC layer protocol. CU-MAC is an SI MAC protocol that separates communication channels for initiating connection and data transmission for concurrent transmission flows so the multi-channel approach can improve transmission bandwidth. Moreover, we apply data loss protection model in buffer management and request/response communication improvement using data piggyback technique in only one connection instead of two in a traditional way. We evaluated our protocol in both simulation and real environment. The results show that our protocol outperform other traditional protocols in term of packet delivery ratio and delay.

The rest of the paper is organized as follows. Section II reviews existing works in MAC layer protocols for WSNs and IoT system. Subsequently, in Section III, we explain our proposed CU-MAC protocol. In Section IV, we describe our implementation of CU-MAC protocol. In Section V, performance evaluation is shown. Finally, Section VI concludes the paper.

Table 1: *Radio power consumption of the MICA2 mote sensors [13].*

Radio State	Power Consumption
Transmit	81 mW
Receive	30 mW
Idle	30 mW
Sleep	0.003 mW

2. RELATED WORK

In IoT network architecture, we can categorize into four levels; 1) Sensor Level responses for collecting data from interesting environment 2) Network Level provides a connection between sensors and gateways 3) Service Level manages the data collection follow the service provide policy to provide these data to user 4) Interface Level is used for displaying theses data to user[8, 9]. WSNs take important roles in Sensor Level and Network Level. The recommended standards from IEEE and IETF can be illustrated in Fig. 1. Each stack can be described below.

• Constrained Application Protocol (CoAP) [10] is released by IETF to be an application protocol operating on UDP. It is designed for sensor devices to work in client/server way with the same methods found in HTTP such as GET, PUT, POST and DELETE.

• IPv6 over low power wireless personal area networks (6LoWPAN) [11] is a minimized version of IPv6 that is released by IEEE so IPv6 can be used in IEEE802.15.4 which limits each frame packet size to only 150 bytes.

• IPv6 routing protocol for low-power and lossy networks (RPL) [12] is a routing protocol designed for low power device. RPL construct a destination orientated directed acyclic graph (DODAG) containing concerned cost such as power and latency.

• The medium access control layer (MAC) will be discussed in detail later in this section. For physical layer (PHY), all of sensors rely on IEEE802.15.4 which designed for low power communication. It provides limited bandwidth at 150 Kbps. and each frame packet size is limited to only 150 bytes.

In this paper, we focus on MAC protocol so we review the major existing MAC protocols of WSNs and elucidate them in term of strengths and weaknesses for each protocol in IoT perspective. In WSNs, all of the MAC protocols are developed by applying the duty-cycle concept because the design of WSNs pays much attention on the power consumption. Table 1 shows the radio power consumption of each sensor node process, the duty-cycle concept is the time schedule for sensor node to turn on radio only when it wants to receive or transmit data, in other case, the radio will be turn off for minimizing the power consumption of radio module [13, 14]. We can classify the MAC protocols in WSNs into 2 categories, which are synchronous MAC protocols and asynchronous MAC

protocols.

The synchronous MAC protocols are the protocols that need to exchange the duty-cycle time scheduler in the WSNs for maintaining the time to transmit or receive data in each node. The popular protocols in this category are S-MAC [15], T-MAC [16], R-MAC [17] and P-MAC [18]. S-MAC has a key design for burst event application while T-MAC is a protocol that is improved from S-MAC protocol by using the concept dynamic duty-cycle in nodes for increasing the transmission speed. R-MAC is different from S-MAC in that it uses the sleep state to transmit data to multiple nodes and P-MAC is developed based on R-MAC by using the bidirectional data transfer for increasing the transmission efficiency. The asynchronous MAC protocols are different from the synchronous MAC protocols in that they do not exchange the information of duty-cycle process. This category of MAC protocols is very popular in WSNs because the properties of this category are more efficient power consumption and lower latency than the synchronous MAC protocols. The most popular MAC protocols of this category are:

• B-MAC [19] is the duty-cycle MAC protocol that each node has its own duty-cycle schedule. If the node has data to transmit, the sender will send the preamble frame to the wireless channel. The preamble frame will be sent within one duty-cycle time. At a receiver node, every round of duty-cycle schedule, the node will check the wireless channel usage. If the receiver node found a long preamble in the wireless channel, the receiver node will power on until receiving the target address information from the last section of preamble. If the receiver node is not a target address, it will go to sleep state. On the other hand, if the node is a target node, it will wait for the data from the sender node. The advantage of this protocol is low latency comparing with the synchronous MAC protocols. However, it has a drawback in the case that neighbor nodes waste their power when they receive a long preamble but in fact they are not target receiver nodes. This problem is well-known as overhearing problem

• RI-MAC [20] is a MAC protocol that the receiver node initiates the connection. When the receiver node wakes up, it will send the packet piggybacked with its own address into the wireless channel. If the sender node who needs to send data to the receiver node hears this packet, it will send data to the receiver node. RI-MAC is more efficient than the sender-initiated protocols in that the sender nodes have no need to send long preamble so the sender nodes can save their power. The drawback of this protocol is the high collision when there are more than one sender nodes expecting to send data simultaneously.

• X-MAC [6, 7] is a MAC protocol for improving the drawback of B-MAC protocol by using series of short preamble instead of a long preamble. The short preamble used in X-MAC protocol is piggybacked with the target address. When the node wakes up and receives the short preamble, it also checks the target address in the short preamble. If it is not the target address, it will quickly go to sleep state. On the other hand, if it is the target address, it will send acknowledgment packet for breaking series of short preambles and wait for the data from the sender node. The advantage of X-MAC protocol is low latency. Moreover, it can reduce the overhearing problem in B-MAC protocol but the weaknesses of X-MAC protocol are the hidden terminal problem and the competition of data transfer when there are more than one senders in coverage wireless area.

There are issues that need to be considered when using these previously proposed MAC protocol in IoT environment as following.

• CoAP requires a two-way communication as a server/client approach that has a request message and a response message but for regular WSNs, most of communication direction is from sensors to server.

• Although 6LoWPAN has a minimized header from IPv6, the header size is still larger than traditional WSNs. If size of data and header is larger than 150 bytes, this will increase number of packets due to packet fragmentation.

• RPL need neighbor discovery process for its operation which require broadcast communication in the network. Unfortunately, the synchronous MAC protocol and RI MAC protocol do not capable to broadcast data so they have to use unicast for each neighbor which greatly increase delay and power consumption.

For above reasons, SI MAC protocol is the most suitable MAC protocol in IoT network and X-MAC is one of the most popular SI MAC protocols applied in IoT network[21]. However, X-MAC in IoT environment cannot perform well as reported in [7], X-MAC has a lost preamble acknowledgement issue. The issue occurs in two scenarios. The first one is when the sender finish its transmission before it receives the response preamble. Another scenario is a hidden terminal problem. There are two concurrent preambles from two individual senders but a receiver gets both of preambles at the same time. Consequently, the lost issue will increase when the number of senders and the volume of data increase. If there are more than one sender in the area, other senders will have to wait for long time due to one channel communication of X-MAC called a black spot issue. It can be seen that the main problem is that the above-mentioned protocols cannot support bidirectional and concurrent transmissions.

In order to design an efficient MAC protocol in IoT environment, one should consider bidirectional and concurrent transmissions that occur much more often than traditional WSNs. We proposed CU-MAC to outperform a traditional MAC protocol us-

ing multi-channel technique. The multi-channel communication improves connection initiation and data transmission of concurrent transmission. This allows CU-MAC to overcome request/response message frequently occurred in IoT environment. We also address buffer overflow issue and black spot issue in X-MAC so CU-MAC can operate efficiently in IoT environment.

3. CU-MAC PROTOCOL DESIGN

The design goals of CU-MAC protocol follow three main properties in order to improve the efficiency of WSNs in IoT environment; multi-channel communication support, piggyback data transferring support and continuous data transferring support.

In IEEE802.15.4 standard, there are 9 usable channels for wireless communication. To minimize power consumption, CU-MAC selects one wireless channel to transfer both control information and data when there is only one sender in the same coverage area. But if there are more than one senders in the same coverage area, CU-MAC will select another one usable channel for data transfer for each sender. Specifically, if there are two senders in the same network coverage area, CU-MAC will use one channel for control information which will be shared among the two senders. Then, it will use another two channels for data transfer for each sender. This allows multiple senders to transfer data simultaneously. This technique is different from X-MAC in that X-MAC will use only one channel for both control information and data transfer. If there are multiple senders in the same coverage area, only the first sender can send control information and transfer data whereas the rest of the senders need to wait for the first sender to finish the transfer. The transfer will be done sequentially, not simultaneously. Therefore, our CU-MAC technique can reduce latency of data transfer in overall and address the black spot issue in X-MAC. To share the same control channel for two senders, both of them must send their own preamble without interfering to each other. CU-MAC is comprised of an Advertising Phase and a Data Transferring Phase.

Generally, the Advertising Phase is a phase that a sender needs to send an announcement to its neighbor nodes before transferring data. CU-MAC performs sampling at the channel usage twice to ensure that there is no other sender in the coverage area in order to know the timing of free time slot between two preambles of the other sender. In hidden terminal problem case, if the receiver detects the collision from both sender, the receiver will send a warning message to a new sender so the new sender uses back-off interval to avoid hidden terminal problem. Then the Data Transferring Phase will start after the sender get Ready to Receive Information (RA) from its receiver. Fig. 2 show the pseudocode of CU-MAC protocol.

Pseudocode *CU-MAC protocol*

```
function SEND(target.address)
    set slot.sample to 2;
    set slot.usage to 0;
    for (i=1;i<slot.sample;i++) do
        CCA;                    //Clear Channel Assessment
        if (CCA is positive) then
          slot.usage++;
        end if
    end for
    switch slot.usage do
        case 0:
              send preamble embed with target.address;
              wait for acknowledgment packet;
              send data packet;
              break;
        case 1:
              learn timing of preamble;
              send preamble embed with target.address and data.transfer.channel;
              move wireless channel to data.transfer.channel;
              wait for acknowledgment packet;
              send data packet;
              break;
        case 2:
              set back-off timer for wake-up node;
              sleep;
              break;
        end switch
end function

function RECEIVE (this.address)
    set slot.sample to 2;
    for (i=1;i<slot.sample;i++) do
        CCA;
        if (CCA is positive) then
           get target.address from preamble;
           if (target.address=this.address) then
              get data.transfer.chanel from preamble;
              if (data.transfer.channel) then
                 move wireless channel to data.transfer.channel;
              end if
              repeat
                 send acknowledgment packet embed with ready.to.receive;
              until received data or time of send acknowledgment packet equal duty cycle time
              sleep;
              break;
           end if
        end if
    end for
    sleep;
    break;
end function

start
    node.wake-up;
    if (node have data to send) then
        SEND(target.address);
    else
        RECEIVE (this.address);
    end if
end
```

Fig.2: *Pseudocode of CU-MAC protocol.*

For more clarification, Fig. 3 shows timelines of CU-MAC advertising phase when there are only Sender A and Receiver X. At T0, Sender A wakes up and needs to send data to Receiver X. Firstly, Sender A samplings channel usage twice for checking if there is any sender sending data. If it does not detect any signal in the control channel, which means that there is no other senders, it will send the first short preamble packet with Target Address Information (TA) immediately (In this case TA is Receiver X). After that, Sender A listens for a Ready to Receive Information (RA) from Receiver X. If it cannot receive anything, it will go back to send the second short preamble. This process is an Advertising Phase and CU-MAC repeats the process until the spending time is equal to the duty-cycle time. This is to ensure that all the neighbor nodes hear the preambles. At the state of listening of the Advertising Phase, if Sender A re-

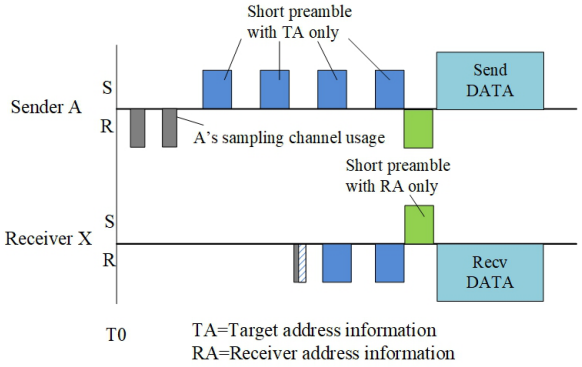

Fig.3: *CU-MAC timelines when there is only one sender.*

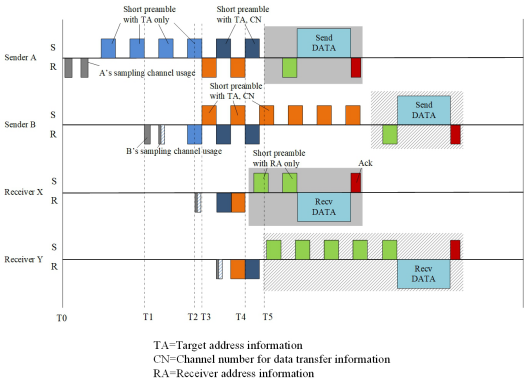

Fig.4: *CU-MAC timelines when there are two senders and two receivers.*

ceives a Ready to Receive Information (RA) packet from Receiver X, it will start Data Transferring Phase and send data to Receiver X at control channel for saving the power consumption. Because currently, there is only one sender.

3.1 Multi-channel Communication Support

To enhance the communication performance and support number of devices, CU-MAC utilize 9 usable channels in IEEE 802.15.4 to make multi-channel communication. We use the scenario in Fig. 4 with additional Sender B and Receiver Y to explain how CU-MAC works if there are more than one sender in the same area. We suppose that, at T1, Sender B wakes up and needs to send data to Receiver Y. Sender B is going to start its Advertising Phase as Sender A did but when it checks the channel usage, it found that other senders are using control channel. Sender B will listen the control channel. After hearing the preamble from Sender A, Sender B learns the interval time of Sender A's preambles and sends its own short preamble with TA and Data Transferring Channel Number (CN) in free slot in control channel.

At T3, Sender A hears the preamble from B. It checks the CN that Sender B selects. After that, Sender A will choose its own CN that is different from Sender B's CN and will include its own CN in the next sending preamble. Later at T3, Receiver X wakes up from scheduler. It checks the control channel usage and found that the control channel is in use. It listens to the control channel and hears the preambles from Sender A and Sender B. When it decrypts the packet, it found that it is the target address of Sender A. Therefore, Receiver X will switch its wireless channel number to the CN that is embedded in the preamble of Sender A.

At T4, Sender A starts Data Transferring Phase at Receiver X. The gray box in the figure means the node switches the wireless channel from control channel to data transferring channel already. In this Data Transferring Phase, Receiver X will send series of short preambles embedded with Ready to Receive Information (RA) until it receives a data packet from Sender A. In case that Receiver X keeps sending short preambles for longer than the duty-cycle time but there is no data packet from Sender A. It is most likely that Sender A left the network already so Receiver X will switch back to the control channel.

At T5, Sender A switches its wireless channel from control channel to data transfer channel. When it comes to data transfer channel, it is waiting for the short preamble with RA information from Receiver X. After it receives the preamble form Receiver X, it will send data to Receiver X and waiting for the corresponding acknowledgement packet from Receiver X. If Sender A receives the acknowledgement packet properly, it will switch its wireless channel to control channel. Otherwise, if it cannot receive anything after switching to data transfer channel or cannot receive the acknowledgement after sending the data packet, it will stay in this channel until it receives the preamble from Receiver X or until it stays in this channel longer than the duty-cycle time. Sender B and Receiver Y follows the same process as Sender A and Receiver X do.

Fig. 5 shows another scenario when three senders want to transfer data at the same time. As can be seen from the figure, Sender A and Sender B are doing the Advertising Phase as we explained earlier. At T6, when Sender C wakes up and needs to send data. Firstly, it samplings the channel usage and found that the control channel is used by other senders. At T7, Sender C listens to the control channel for hearing the preamble packets of other senders. In this case, Sender C learns and knows that there is no available slot for sending its own preamble packet. At T8, Sender C will random the back-off timer to avoid hidden terminal problem, supposed that, it wakes up again at T9 (the random back-off timer is less than a duty-cycle time value). After Sender C wakes up again at T9, it will do its Advertising Phase again. During that time, Sender A switched to the data transfer channel already so Sender C can take the

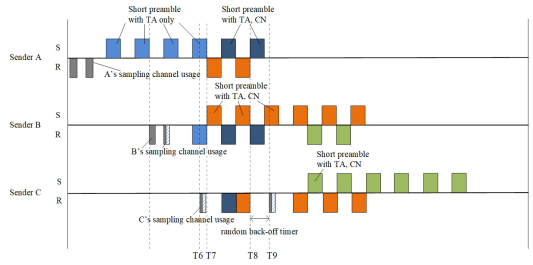

Fig.5: *CU-MAC timelines when there are three senders at control channel.*

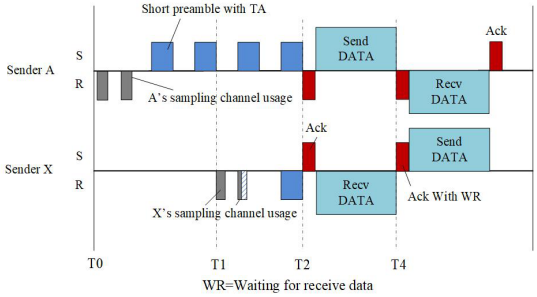

Fig.6: *CU-MAC piggyback data transferring.*

free slot.

3.2 Piggyback Data Transferring Support

Most of IoT applications using request/response communication that normally transfer data in both directions. CU-MAC supports such bi-direction data transfer by piggyback the data using only one communication connection. This technique can reduce an additional overhead and delay comparing with the traditional MAC protocol that need each connection initiation for each data transferring direction. For more clarification, Fig. 6 shows the example for piggyback data transferring in CU-MAC. At T0, Sender A wakes up and needs to send data to Sender X. It checks the control channel usage. If the channel is empty, Sender A will send the short preamble. At T1, Sender X wakes up and needs to send data to Sender A. It checks the control channel and found that the channel is used by other senders. It hears the preamble from Sender A and found that it is the Target Address (TA) of Sender A. After that, it uses piggyback data transferring by sending an acknowledgement at T2 in order to tell Sender A that Sender A can send data packet now. After Sender X receives the data packet already, Sender X will send an acknowledgement with Waiting for Receiving Data Information (WR) to tell Sender A that Sender A must wait for receiving data packet before going to sleep. Then, Sender X can send a data packet to Sender A.

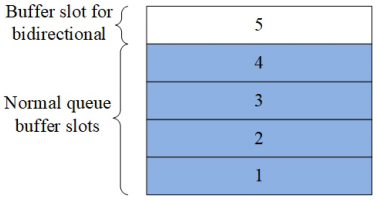

Fig.7: *Buffer stack diagram.*

3.3 Continuous Data Transferring Support

In IEEE 802.15.4, the data frame can contain 150 bytes at maximum. That means if the sensor node needs to send data bigger than the data frame size, it will separate the data into many packets and send those packets to the destination node. Most of IoT devices contain various types of data that their sizes are bigger than this data frame size. For this case, CU-MAC supports multiple continuous data packets at the same time, thus reducing the latency of the packets. The key that makes CU-MAC be able to send multiple continuous packets is how to set the policy in the buffer stack to protect buffer overflow

The buffer stack in CU-MAC is designed to contain 2 parts. The first part is one buffer slot that is dedicated for bidirectional data transfer, so-called bidirectional buffer slot. This slot can help sensor nodes support piggyback data transferring. The second part is a group of buffer slots that are dedicated for queuing the normal data, so-called normal buffer slots. Each buffer slot is 150 bytes as the maximum data fame size. Fig. 7 shows the diagram of buffer stack in CU-MAC.

We set policies for buffer stack in CU-MAC as follows.

- Drop a packet in the buffer only when the packet is expired.
- If the normal buffer slots are full, the receiver node will deny receiving any packets except the case that the packet comes from the target node that the receiver node needs to send data to. In this case, the receiver node can use the bidirectional buffer slot to receive a packet.
- The bidirectional buffer slot can be used for bidirectional data transfer only.

To enable continuous data transfer, CU-MAC needs to embed the following fields into the short preamble.

Target Address Information (TA)
Channel Number for Transfer Data (CN)
Number of Packets to be Sent (NS)
Number of Empty Normal Buffer Slots (NE)
Number of Packets to be Received (NR)
Ready to Receive Information (RA)
Waiting for Receiving Data Information (WR)

Fig. 8 shows the timeline of Sender A and Sender X in bidirectional and continuous data transfer mode. Suppose that Sender A is sending continuous data to

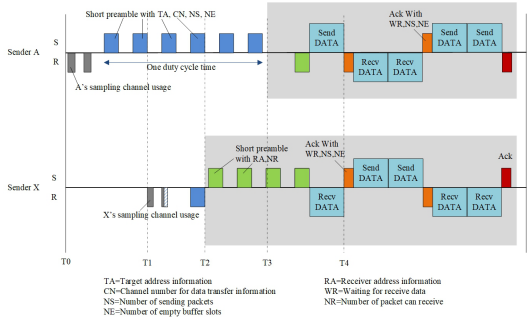

Fig.8: *CU-MAC continuous data transfer.*

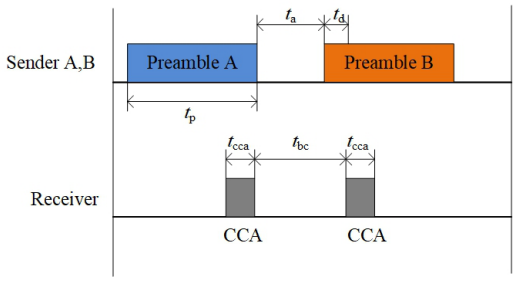

Fig. 9: *CU-MAC preamble and CCA timing.*

Sender X and Sender X is also sending continuous data to Sender A. We put the number of packets to be sent and the number of empty normal buffer slots into the short preamble packet because we need the receiver node to know the boundary of packets that can be received and sent when it stays in the data transfer phase so that it can manage the pool buffer slots by itself.

As can be seen in Fig. 8, all of the nodes have one bidirectional buffer slots and four normal buffer slots. At T2, Sender X receives the short preamble from Sender A. According to the short preamble, Sender X will know that Sender A needs to send 3 packets to Sender X but now Sender X does not have any empty normal buffer slots. Moreover, Sender X has 2 packets to send to Sender A (other packets in the buffer slots may be sent to other nodes). In this case, Sender X will switch to data transfer channel. Then, Sender X will send a short preamble embedded with NR equal to 1 because now Sender X has one bidirectional buffer slot that is available. At T3, when Sender A completes to send its short preambles in control channel, it will switch to data transfer channel and wait for receiving a short preamble from Sender X. When it receives the short preamble from Sender X, it will know that Sender X has only one available buffer slots so it will send only one data packet to Sender X. After Sender X receives the packet from Sender A, at T4, Sender X sends an acknowledgement with WR=on, NS=2 and NE=0. WR is set on in order to tell Sender A to receive data packets. NS is set to 2 because Sender X has 2 packets to send to Sender A. NE is set to 0 because Sender X wants to tell Sender A that it does not have empty normal buffer slot. After Sender A receives the acknowledgement from Sender X, it will stay to receive 2 data packets from Sender X. The continuous data transfer between Sender A and Sender X will repeat like this until there is no data packets to be sent.

4. IMPLEMENTATION

Our CU-MAC protocol has been implemented in Contiki OS 2.7[22] using IEEE802.15.4 standard wireless module. We have used the Clear Channel Assessment (CCA) mechanism for indication the activity in radio channel. The CCA applies Received Signal Strength Indicator (RSSI) of the radio transceiver as an indicator to detect whether the wireless channel is busy or not. Specifically, if the RSSI level is greater than a given threshold, the CCA returns negative value indicating that the wireless channel is busy. If the RSSI level is less than the given threshold, the CCA returns positive value indicating that the wireless channel is clear. The major key to make CU-MAC operation successful is to set all the interval times for sending and receiving preamble packets carefully.

The interval times that are important for CU-MAC operation are shown in Fig. 9. Each of them are defined as follows.

t_p : the transmission time of the preamble packet

t_a: the interval time between each preamble packet transmission

t_d: the time required for the other sender to successfully detect the preamble packet

t_{cca}: the time used for a CCA indicator process

t_{bc}: the interval time between each CCA process

Eq. (1) shows the timing constraints in CU-MAC. In order for the CCA to detect that the wireless channel is busy, t_{bc} must be greater than ta and must be less than t_a+t_d because if the first CCA process cannot detect the preamble packet, the second CCA process will definitely detect the preamble packet. The transmission time for sending preamble (t_p) must be greater than t_{bc}+$2t_{cca}$ because if tp is less than $t_b c$+$2t_{cca}$, the CCA process may not be able to detect the preamble packet.

$$t_a < t_{bc} < t_a + t_d < t_{bc} + 2t_{cca} < t_p \qquad (1)$$

The interval time between each preamble packet transmission is defined by the IEEE802.15.4 specification as 12 symbols. In IEEE802.15.4, one symbol is 4/250 milliseconds that means the possibility least time of t_a is 12 symbols*4/250 = 0.192 milliseconds. In our implementation, we use 4 bytes long preamble and 1 byte start of frame delimiter for declaring preamble packet. The second sender needs to send 5 bytes transmission time before the first sender can detect preamble packet. Thus, t_d= (5 bytes*8) / 250

kbps = 0.16 milliseconds. In Chipcon CC2420 radio transceiver specifications, the time for a CCA indicator process is 0.192 milliseconds. According to the above explanation, Eq. (1) becomes Eq. (2).

$$0.192 < t_{bc} < 0.352 < t_{bc} + 0.384 < t_p \qquad (2)$$

According to Eq. (2), the transmission time of the preamble packet t_p must be greater than 0.736 milliseconds. We use 25 byte preamble packet in order to include all the keys mentioned in Section 3. Since the preamble packet size is 25 bytes and the physical bitrate is 250 kbps in CC2420 radio transceiver, the t_p is equal to 0.8 milliseconds.

According to Eq. (1) and (2) discussed above, t_a must be at least 0.192 milliseconds. However, in our real implementation, we use the RTC (Real Time Clock) from Contiki OS, which can adjust the clock skew every 0.1 milliseconds, so we set ta to be 0.2 milliseconds. According to our experiment, there exist some processing delay from WiFi modules. As a result, t_{bc} and t_p need to be set to 0.4 and 0.8, respectively, in order to be able to detect the preamble.

Finally, in order to satisfy all the constraints, CU-MAC implementation uses the following configurations:

t_a = 0.2 milliseconds

t_{bc} = 0.4 milliseconds

t_p = 0.8 milliseconds

In our implementation, CU-MAC reserves 750 bytes memory for packet buffer memory, which is equal to 5 slots because in IEEE802.15.4 the MTU size is 150 bytes. 4 slots of memory are used for sending or receiving the data packet and 1 slot of memory is used for bidirectional data transfer. Thus, CU-MAC in our implementation can send or receive 4 sequential data packets at the maximum.

5. EVALUATION

To evaluate the protocol performance, we ran an extensive set of experiments and simulations. The wireless nodes in our testbed are Tmote Sky motes, which use an 8MHz TI MSP460 processor and have 1MB of external flash. The radio chipset used by Tmote Sky is the Chipcon CC2240[23]. It is an IEEE802.15.4 compliant device, has a data rate of 250kbps and operates in the 2.4GHz ISM band. For our simulations, we used Cooja, the standard simulation for Contiki OS. The metrics used for evaluating the protocol performance are packet delivery ratio, fairness, hidden terminal problem robustness and IoT evaluation. We compare CU-MAC with the default code of the X-MAC protocol based on Contiki 2.7 operating system. The duty-cycle is configured at 25Hz, 10Hz and 5Hz (40ms, 100ms and 200ms sleep time).

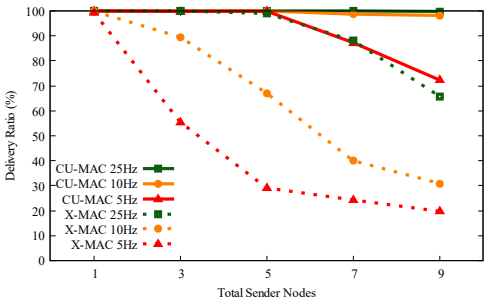

Fig.10: *Delivery ratio results at one receiver node.*

5.1 Packet Delivery Ratio

For evaluation this metric, we set two experiments on the real testbed. In the first experiment, there is only one receiver node but the number of the sender nodes is varied from 1 to 9. The transmission rates at the sender nodes are 4 packets per second and the packet size is 120 bytes. We use UDP to send the packets for 180 seconds. All of the sender nodes will drop packets only when they have packets more than 4 packets in their queue. The result shows in Fig. 10.

As can be seen from the figure, X-MAC cannot deliver data to the receiver node when the number of the sender nodes increases. In general, with the duty-cycle 5 Hz, 10 Hz and 25 Hz, the maximum number of packets that X-MAC can transfer is 5, 10 and 25 packets, respectively. As a result, when the number of the sender nodes increases, a lot of packets are dropped. On the other hand, CU-MAC can deliver higher traffic than X-MAC. This is because CU-MAC has ability to support continuous transferring. CU-MAC is different from X-MAC in that when the sender node has multiple data packets to send to the same receiver, CU-MAC will transmit all of the data packets to the receiver within one connection. See Section 3.3 for more details. With the same situation, in contrast to CU-MAC, X-MAC can transmit only one data packet per one connection, leading to long waiting time and buffer overflow.

In the second experiment, all parameters are set as same as the first experiment but the number of the receiver nodes is changed from 1 to 2 and the number of the sender nodes is varied from 1 to 9 per one receiver nodes. Fig. 11 shows the result. As can be seen from the figure, CU-MAC outperforms X-MAC. This is because, in the advertising phase, CU-MAC allows preambles from different senders to be sent at the same time. Thus, different receivers can receive the preambles and data packets simultaneously. In this figure, the result of X-MAC at the duty-cycle 5Hz is not shown because most of the sender nodes crash and stop transferring after 10 to 30 seconds of the runtime.

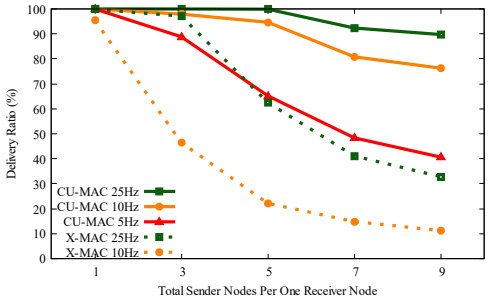

Fig.11: *Delivery ratio results at two receiver nodes.*

Fig.12: *Line topology experiment scenario.*

5.2 Fairness

To show the fairness property of CU-MAC, we run 3 experimental scenarios in our real testbeds. The first experiment scenario is the line topology with 3 nodes as depicted in Fig. 12. We test by sending data packets in 2 flows simultaneously. The first flow is sent from node 1 to node 3, while the second flow is sent from node 3 to node 1. All flows consist of 150 UDP packets, the payload of which is 120 bytes. All nodes have the maximum queue buffer of 4 packets and use the duty-cycle at 5 Hz. The sender nodes at both sides can hear each other so there is no hidden terminal problem in the scenario.

Fig. 13 shows the fairness of CU-MAC and X-MAC. Even though both CU-MAC and X-MAC have fairness property, the packet delivery ratio when using CU-MAC is higher than X-MAC. Moreover, the time used to send the packets when using CU-MAC is also less than X-MAC. When using CU-MAC, the packet delivery ratio is 99.7% and the time is 270 seconds. When using X-MAC, the packet delivery ratio is 81.2% and the time is 452 seconds. This is because CU-MAC utilizes the benefit of piggybacking to send the packets bi-directionally and send multiple packets in one connection. Moreover, CU-MAC also has buffer overflow prevention mechanism. So, the packet delivery ratio of CU-MAC is higher. On the other hand, X-MAC does not have mechanism to prevent buffer overflow and not support bi-directional communication. Moreover, X-MAC can send only one packet per connection.

In the second scenario, we use the same topology as the first scenario. However, different packet sizes are applied to observe whether the sizes and the number of packets have impact on fairness or not. The first flow consists of 150 UDP packets, the payload of which is 120 bytes. The second flow consists of 300 UDP packets, the payload of which is 60 bytes.

Fig. 14 shows the result of the second scenario. It

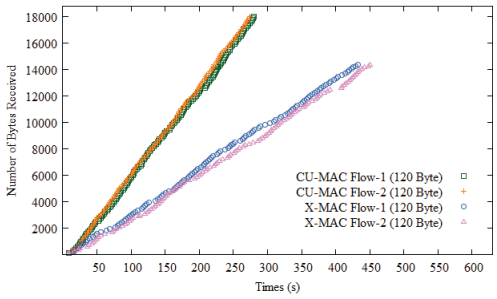

Fig.13: *Fairness of 2 flows.*

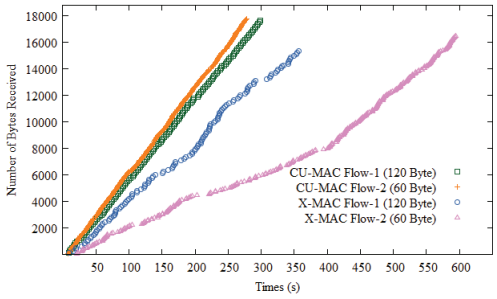

Fig.14: *Fairness of different packet size.*

can be seen that CU-MAC has fairness property between 2 flows, but X-MAC does not. CU-MAC provides higher packet delivery ratio and shorter time to send the packets than X-MAC. When using CU-MAC, the packet delivery ratio is 98.7% and the time is 310 seconds. When using X-MAC, the packet delivery ratio is 88.2% and the time is 598 seconds. This is because CU-MAC utilizes the benefit of piggybacking to send the packets bi-directionally and send multiple packets in one connection. As a result, the second flow, which has smaller packet size and more packet numbers, can have the number of received bytes as same as the first flow. In contrast, X-MAC can send only one packet per connection and there is no bi-directional communication support. As a result, the second flow has smaller number of received bytes.

The third experiment scenario is Y topology with 4 nodes as shown in Fig. 15. There are three flows in the scenario. All flows consist of 500 UDP packets, the payload of which is 120 bytes. Each flow starts to send data at different timings. The first flow starts to send data at 5 seconds. The second flow starts to send data at 65 seconds. The third flow starts to send data at 125 seconds. The duty-cycle is set as same as the first experiment. This scenario can represent the node topology in WSN which connect to a gateway.

The results of the third scenario are shown in Fig. 16 and 17. Fig. 16 shows the result for X-MAC. It can be seen that at 125 seconds, all the nodes are trying to send the data and there are none of them receive any data. This results in buffer overflow problem, and the whole system cannot go on until the buffer is

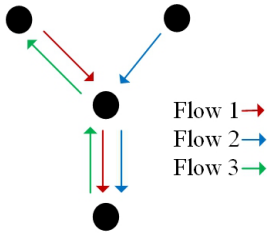

Fig. 15: *Y topology experiment.*

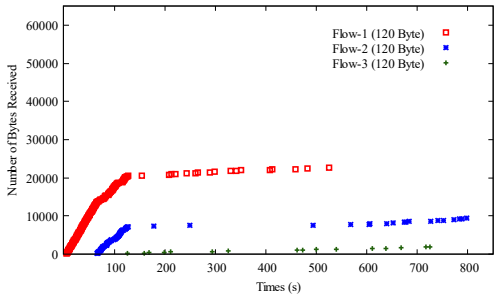

Fig. 16: *X-MAC fairness of 3 flows.*

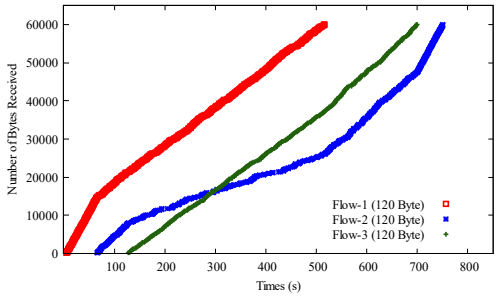

Fig. 17: *CU-MAC fairness of 3 flows.*

empty. Even when the whole system can continue its process, the buffer overflow problem will occur again inevitably. Fig. 17 shows the results for CU-MAC. At the beginning, there are only the first flow so it can fully utilize the whole bandwidth. Then, when the second flow starts to send data, the first flow and the second flow compete for the bandwidth. As can be seen from the figure, during 65 to 125 seconds, the overall bandwidth is shared by the first flow and the second flow. After the third flow starts to send data, the throughput of the second flow decreases whereas the throughput of the first flow remains unchanged. This is because the third flow takes advantages from the bidirectional buffer slot for transferring data at the same time with the first flow.

In Fig. 18, we observe that the data transfer rate of the second flow becomes zero many times. This is because the buffer of the middle node is full of packets from the first and the third flows. Since these two flows are sending in opposite side, it utilizes the benefit of the bi-directional communication. So, the

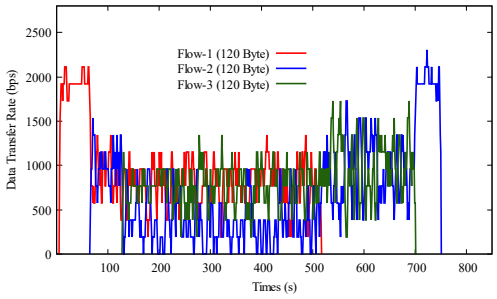

Fig. 18: *CU-MAC fairness in bandwidth utilization.*

second flow need to wait for the first and the third flows finish transferring data packets.

According to the above results, CU-MAC shows its fairness property in the first and second scenarios when there are multiple flows in the topology. Moreover, CU-MAC can overcome X-MAC in all the scenarios. Even if the scenario that X-MAC cannot work properly, CU-MAC can work with fairness for all flows.

5.3 Hidden Terminal Problem

The hidden terminal problem is one of major problems that degrades performance of MAC layer protocols because it can lead to a lot of collisions. Most of MAC layer protocols relieve the impact of this problem by setting the random back-off timer for the next transmission. CU-MAC solves this problem by allowing the receiver who detects collision to send an alert acknowledgement packet immediately to the senders. If the receiver still detects collision, it means that the alert acknowledgement packet cannot solve the problem. Then, CU-MAC allows the receiver to set the random back-off timer before waking up the next scheduler.

For evaluation, a line topology with three real sensor motes are used as shown in Fig. 12. Since we want the situation that the hidden terminal problem happens, Node 1 and Node 3 are located far away and cannot hear each other whereas Node 2 can listen to both Node 1 and Node 3. The duty-cycle is set to 5Hz. According to this setting, the hidden terminal problem will occur at the middle node of the line topology. There are two flows in this experiment. The first flow is sending data packets from Node 1 to Node 3. The second flow is sending data packets from Node 3 to Node 1. Both of the flows are sending at 4 packets/second and each packet contains 120 bytes payload. The runtime is 100 seconds. The experiment is repeated for 20 times. Table 2 show the result of this experiment. We compare CU-MAC with the alert acknowledgement implementation with CU-MAC without the alert acknowledgement implementation. As can be seen from Table 2, the average transmission success rate of CU-MAC with the alert acknowledgement implementation is higher than that

Table 2: *Delivery ratio results in the hidden terminal problem scenario.*

CU-MAC	min.	max.	$\bar{x}$	σ
Without alert ACK	35.4%	93.7%	64.6%	12.4%
With alert ACK	89.6%	98.2%	96.3%	1.7%

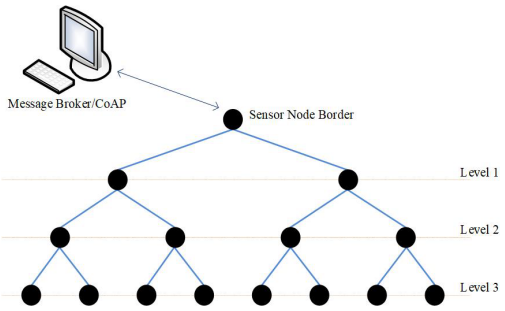

Fig.19: *Topology IoT testbed.*

without the alert acknowledgement implementation about 31.7

5.4 IoT Evaluation

In order to evaluate CU-MAC in IoT environment, we use a tree topology as illustrated in Fig. 19. This tree topology represents the IoT device connection that requires a gateway to collect data from the sensors in the area. The route connection is fixed to avoid additional packet from neighbor discovery in RPL protocol which can affect to evaluation results. In this real testbed, every node produces a packet every 3, 5 and 10 seconds. The data are transmitted with the same behaviour of CoAP that send data to a message broker server which has an acknowledge for every transmission. The evaluation compares between CU-MAC, X-MAC and RI-MAC. All of protocols are Asynchronous protocol while CU-MAC and X-MAC are the Sender Initiate Connection protocol and RI-MAC is the Receiver Initiate Connection protocol. The duty cycle of every protocol is 10Hz. The evaluation will be done 10 times in period of 30 minutes and the results will be shown in the averaged result of each tree level.

5.4.1 Packet Delivery Ratio

The packet delivery ratio results are shown in Fig. 20-22 with data transmission every 10, 5 and 3 seconds respectively. When we look at the results on every level of the tree, the SI MAC protocols outperform RI MAC protocol because when more than one sender are sending data to the same destination, the RI MAC protocol has to wait until the receiver is available in each cycle. This can cause an additional delay and the packet is expired when it is delivered to the destination. On the other hands, the sender in CU-MAC and X-MAC can immediately transmit

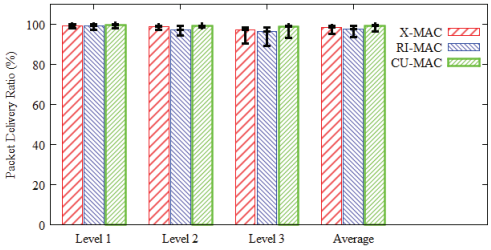

Fig.20: *Packet delivery ratio at IoT every 10 seconds.*

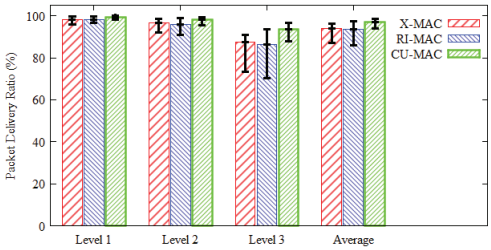

Fig.21: *Packet delivery ratio at IoT every 5 seconds.*

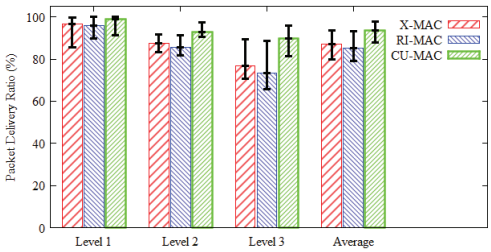

Fig.22: *Packet delivery ratio at IoT every 3 seconds.*

the data upon the first sender finish its transmission. CU-MAC has higher packet delivery ratio than X-MAC because CU-MAC has the acknowledgement mechanism whereas X-MAC does not have. In particular, the nodes in CU-MAC can retransmit the packet within the same connection initiation whereas the nodes in X-MAC will close their connection once they send out the packet. If the packet is lost, there is no retransmission.

When the data transmission increases to every 5 and 3 seconds, the results have the same trend with 10-second result. However, the performance of nodes at the leaves of tree has lower packet deliver ratio than nodes in other levels. Because there are many nodes at the leaves that cause congestion problem. RI-MAC has the lowest packet delivery ratio due to delay in transmission and wait packets in queue. While CU-MAC utilizes the multi-channel communication that can handle numbers of transmissions, so it also provides more packet delivery ratio than X-MAC and RI-MAC.

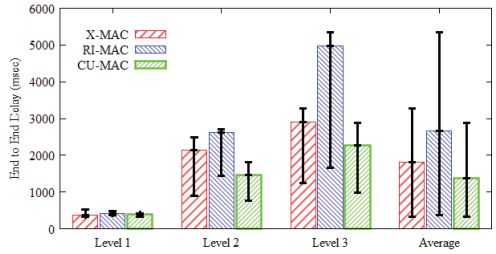

Fig. 23: *End to end delay at IoT every 3 seconds.*

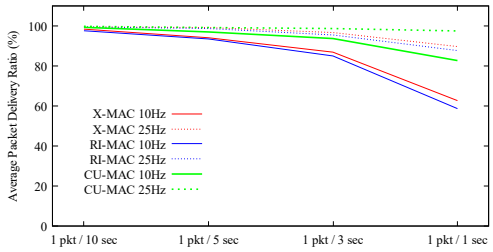

Fig. 24: *Average packet delivery ratio at duty-cycle 10Hz and 25Hz.*

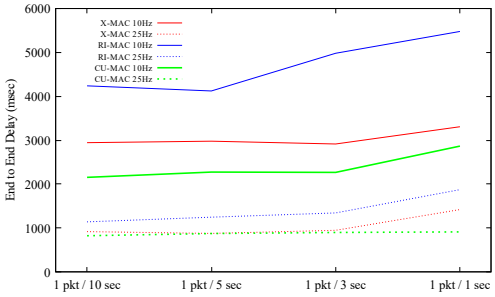

Fig. 25: *Average level 3 end to end delay at duty-cycle 10Hz and 25Hz.*

5.4.2 End-to-End Delay

Fig. 23 shows the end-to-end delay from each node to gateway when each node produces a packet in every 3 seconds. As a result, every protocol can deliver the data from nodes in level 1 and level 2 within 3 seconds. However, CU-MAC is only a protocol that can deliver every packet from nodes in level 3 within 3 seconds. Therefore, packets in X-MAC and RI-MAC has the end-to-end delay more than 3 seconds which is longer than the period that each node produce a packet. These packets will wait in the queue and the number of delayed will increase until they were dropped due to full buffered. This decreases the number of usable packet that were delivered to the gateway.

5.4.3 Performance in short duty cycle

In this section, we set the duty cycle at 25Hz which is shorter than the previous tests to handle more frequent data transmission at 1 packet/s. Therefore, the result will show the packet delivery ratio and end-to-end delay from every protocol with duty cycle at 10Hz and 25Hz and packet rate at every 1, 3, 5 and 10 seconds.

Packet Delivery Ratio results are shown in Fig. 24. When the duty cycle is set to 25Hz. CU-MAC outperform the other protocol. Its packet delivery ratio with duty cycle only 10Hz is close to the performance of X-MAC and RI-MAC with duty cycle 25Hz. Because CU-MAC can transmit the continuous data with one connection. CU-MAC also separates the control channel and the data channel so while nodes transfer the data, they will node interfere with other control message transmission. On the other hands, X-MAC uses only single channel for both control message and data. This leads to performance degradation in dense area. RI-MAC has a problem since its approach that a sender has to wait until the receiver is ready. As a result, a race condition will happen at the receiver and causes to loss packets.

End-to-End Delay at nodes in level 3 are illustrated in Fig. 25. As previously mentioned, X-MAC and RI-MAC cannot deliver the packet within the time that the new packet is generated. So, it will increase the number of packets in the buffer and also the number of lost packets. In this evaluation, we decrease the period of duty cycle in order to decrease the end-to-end delay. The results show that our protocol can operate as same as the others protocol with longer duty cycle period. It means that CU-MAC can support higher data transmission rate with lower duty cycle.

5.4.4 Power Consumption

An important metric that needs to be considered in IoT devices is power consumption. We apply our evaluation on Cooja simulation which is a simulation on Contiki OS with the same topology in Fig. 19. Fig. 26 shows the results of power consumption per delivered packet. In low data rate, CU-MAX and X-MAC have very close power consumption while RI-MAC has the lowest power consumption at every scenario but it also the lowest packet delivery ratio and the highest end-to-end delay. The results of packet delivery ratio and end-to-end delay from our simulation are shown in Fig. 27 and Fig. 28. Although CU-MAC increases its power consumption when the data rate increase, it can deliver every packet within the time that a new packet will be transmit. Therefore, CU-MAC is suitable with IoT environment where numbers of devices need to send their data back to gateway and when these data are still valuable to services and applications.

6. CONCLUSION

In this paper, we have proposed CU-MAC, an asynchronous duty cycle MAC protocol for Internet of

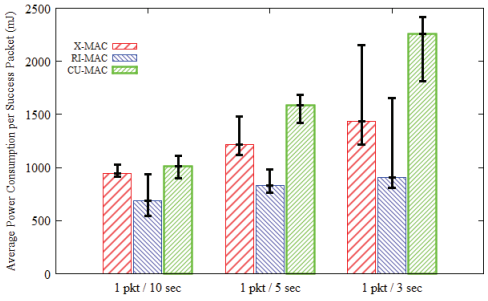

Fig.26: *Average power consumption per success packet.*

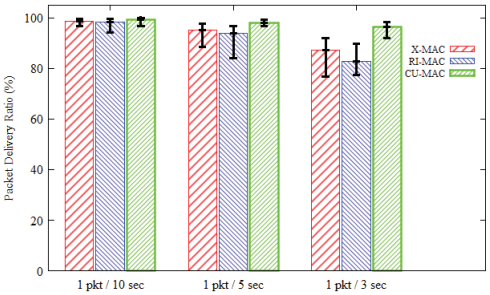

Fig.27: *Average packet delivery ratio in Cooja simulation.*

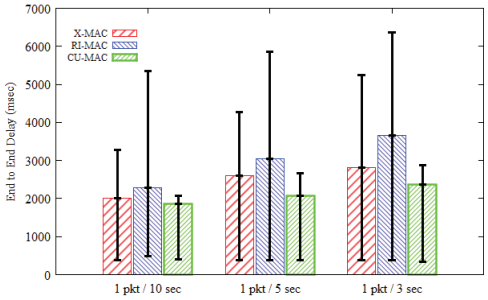

Fig.28: *End to end delay in Cooja simulation.*

Things in Wireless Sensor Networks. CU-MAC uses multi-channel approach and piggyback technique to operate in high traffic density of IoT environment efficiently and effectively. CU-MAC separates control channel from data transferring channel to minimize the time that sender and receiver using the medium channel and to support multiple data transmission flow at the same time. We implemented CU-MAC in ContikiOS in a testbed of TSky motes and Cooja simulation for performance evaluation. CU-MAC achieves higher throughput and higher packet delivery ratio under high-density IoT traffic loads than the existing protocol. Especially when there are contending flows, such as burst traffic or multi-transmission flows at the same routing path, CU-MAC significantly improves throughput and packet delivery ratio. In our evaluation on ContikiOS testbed, when packets is produced from IoT nodes every 3 second, CU-MAC provides packet delivery ratio at 98.7% and has 18.4% lower end-to-end delay than X-MAC protocol.

7. ACKNOWLEDGEMENT

This research was supported by Chula Computer Engineering Graduate Scholarship for CP Alumni and the Wireless Network and Future Internet Research Unit, Ratchadaphiseksomphot Endowment Fund, Chulalongkorn University.

8. REFERENCES

References

[1] M. R. Palattella et al., "Standardized protocol stack for the internet of (important) things," *IEEE Commun. Surveys Tutorials*, Article vol. 15, no. 3, pp. 1389-1406, 2013, Art. no. 6380493.

[2] A. Meddeb, "Internet of things standards: who stands out from the crowd?," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 40-47, 2016.

[3] E. H. Callaway "The Wireless Sensor Network MAC," *in Handbook of Sensor Networks: Algorithms and Architectures*, 2005, pp. 239-276.

[4] G. P. Halkes, T. van Dam, and K. G. Langendoen, "Comparing Energy-Saving MAC Protocols for Wireless Sensor Networks," *Mobile Networks Applicat.*, vol. 10, no. 5, pp. 783-791, 2005.

[5] S. Gehlaut, J. Koti and K. Sakhardande, "To improve the lifetime of wireless sensor network," in *2016 Int. Conf. Inventive Computation Technologies (ICICT)*, vol. 2, 2016, pp. 1-5.

[6] M. Buettner, G. V. Yee, E. Anderson and R. Han, "X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks," in *SenSys'06: Proceedings of the Fourth International Conference on Embedded Networked Sensor Systems*, 2006, pp. 307-320.

[7] J. Beaudaux, A. Gallais, J. Montavont, T. Noel, D. Roth and E. Valentin, "Thorough empirical analysis of X-MAC over a large scale internet of things testbed," *IEEE Sensors J.*, Article vol. 14, no. 2, pp. 383-392, 2014, Art. no. 6600871.

[8] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surveys & Tutorials*, vol. 17, no. 4, pp. 2347-2376, 2015.

[9] L. Shancang, X. L. Da, and Z. Shanshan, "The internet of things: a survey," *Inform. Syst. Frontiers*, vol. 17, no. 2, pp. 243-259, 2015.

[10] Z. Shelby, K. Hartke, and C. Bormann, "RFC 7252 - The Constrained Application Protocol (CoAP)," *Internet Eng. Task Force (IETF)*, 2014.

[11] G. Montenegro et al., "RFC 4944 -Transmission of IPv6 packets over IEEE 802.4," *Internet Engineering Task Force (IETF)*, 2007.

[12] T. P. Winter T., Brandt A., Hui J., Kelsey R. , Pister K. , Struik R. , Vasseur JP. and Alexander R., "RFC 6550 - RPL: IPv6 routing protocol for low-power and lossy networks," *Internet Eng. Task Force (IETF)*, 2012.

[13] F. Alfayez, M. Hammoudeh and A. Abuarqoub, "A Survey on MAC Protocols for Duty-cycled Wireless Sensor Networks," *Procedia Comput. Sci.*, vol. 73, pp. 482-489, 2015.

[14] M. J. Miller and N. H. Vaidya, "Power save mechanisms for multi-hop wireless networks," in *First Int. Conf. Broadband Networks*, 2004, pp. 518-526.

[15] Y. Wei, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. Twenty-First Annu. Joint Conf. IEEE Comput. Commun. Soc.*, 2002, vol. 3, pp. 1567-1576.

[16] T. v. Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," *Proc. 1st int. conf. Embedded networked sensor syst.*, Los Angeles, California, USA, 2003.

[17] P. Xie and J. H. Cui, "R-MAC: An Energy-Efficient MAC Protocol for Underwater Sensor Networks," in *Int. Conf. Wireless Algorithms, Syst. Applicat.(WASA 2007)*, 2007, pp. 187-198.

[18] F. Tong, W. Tang, R. Xie, L. Shu and Y. C. Kim, "P-MAC: A Cross-Layer Duty Cycle MAC Protocol Towards Pipelining for Wireless Sensor Networks," in *2011 IEEE Int. Conf. Commun.(ICC)*, 2011, pp. 1-5.

[19] J. Polastre, J. Hill and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys'04 - Proc. 2th Int. Conf. Embedded Networked Sensor Syst.*, 2004, pp. 95-107.

[20] Y. Sun, O. Gurewitz and D. B. Johnson,"RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks," *Proc. 6th ACM conf. Embedded network sensor syst.*, Raleigh, NC, USA, 2008.

[21] T. Bhosale, M. Patil and V. Wadhai, "A smart farming alternative for small Pomegranate farms of India," in *2015 Int. Conf. Commun. Signal Proc. (ICCSP)*, 2015, pp. 0540-0544.

[22] Contiki OS, "http://www.contiki-os.org."

[23] Chipcon cc2240, "http://www.chipcon.com."

Tanapoom Danmanee was born on October 2, 1973 in Bangkok province, Thailand. He received his B.Eng. degree in Civil Engineering from Price of Songkla University, Thailand in 1997 and M.Sc. degree in Computer Science from Chulalongkorn University, Thailand in 2008. Then he is currently pursuing his Ph.D. program in Computer Engineering at Computer Engineering, Chulalongkorn University. His current research interests concern the design and simulation of network routing and MAC layer in WSNs and IoT networks.

Kulit Na Nakorn received Ph.D. degree in Computer Engineering from Chulalongkorn University, Thailand in 2013. He is currently a researcher in Information and System Engineering Laboratory (ISEL), Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Thailand. His research interests are vehicular networks, IoT, ITS services and smart city application

Kultida Rojviboonchai received B.Eng. degree in Electrical Engineering from Chulalongkorn University in 2000, M.Sc. degree and Ph.D. degree in Frontier Sciences from the University of Tokyo in 2003 and 2006, respectively. During her study in Japan, she acquired Monbukagakusho Scholarship from 2000 to 2006 and Microsoft Research Asia Fellowship in 2004. After receiving Ph.D. degree, she worked as a researcher at Hitachi, Ltd., Japan during 2006-2008. She is currently a professor at Computer Engineering, Chulalongkorn University, Thailand. Her research interests include vehicular networks, software-defined networks, IoT, ITS services, smart city platforms and applications