IEICE TRANS. INF. & SYST., VOL.E95–D, NO.2 FEBRUARY 2012

PAPER Special Section on Architectures, Protocols, and Applications for the Future Internet

MAC^2: A Multi-Hop Adaptive MAC Protocol with Packet Concatenation for Wireless Sensor Networks

Kien NGUYEN[†a)], ***Student Member***, **Ulrich MEIS**[††b)], ***Nonmember***, ***and*** **Yusheng JI**[††c)], ***Member***

SUMMARY Wireless sensor network MAC protocols switch radios off periodically, employing the so-called duty cycle mechanism, in order to conserve battery power that would otherwise be wasted by energy-costly idle listening. In order to minimize the various negative side-effects of the original scheme, especially on latency and throughput, various improvements have been proposed. In this paper, we introduce a new MAC protocol called MAC^2(Multi-hop Adaptive with packet Concatenation-MAC) which combines three promising techniques into one protocol. Firstly, the idea to forward packets over multiple hops within one operational cycle as initially introduced in RMAC. Secondly, an adaptive method that adjusts the listening period according to traffic load minimizing idle listening. Thirdly, a packet concatenation scheme that not only increases throughput but also reduces power consumption that would otherwise be incurred by additional control packets. Furthermore, MAC^2 incorporates the idea of scheduling data transmissions with minimum latency, thereby performing packet concatenation together with the multi-hop transmission mechanism in a most efficient way. We evaluated MAC^2 using the prominent network simulator ns-2 and the results show that our protocol can outperform DW-MAC — a state of the art protocol both in terms of energy efficiency and throughput.
key words: *multi-hop MAC, energy efficient, latency, throughput*

1. Introduction

Advances in wireless communication and sensor technology motivate the development of low-cost, low-power, battery-attached sensor nodes with many potential applications [1]–[3]. In a typical sensor node, the radio module consumes the largest share of the power budget [4]. Therefore, an efficient MAC protocol is vital. Idle listening, i.e. the state in which a node merely awaits incoming packets, is the most significant source of energy wastage. To minimize the idle listening burden most MAC protocols adopt the duty cycling mechanism in which the radio module is turned on and off frequently following the operational cycle.

The duty cycling mechanism achieves energy efficiency at the cost of degrading latency and throughput performance. Meanwhile, an increasing number of prospective applications not only impose requirements on energy efficiency but also on other characteristics such as delay and throughput [5], [6]. Therefore, many improvements to the initial mechanism have been proposed. Among them the idea of forwarding a packet through several hops within one duty cycle, as initially shown in RMAC [7] (Routing enhanced MAC), has proven to be particularly efficient in achieving those goals. The proposed protocols use a single control packet instead of the common RTS/CTS (Request-to-Send/Clear-to-Send) pair to setup a flow across multiple hops. Since next-hop information is required when setting up the flow this is essentially a cross-layer optimization. We call this class of protocols henceforth multi-hop MAC.

The state-of-the art protocol named Demand Wakeup MAC (DW-MAC) [8] can support a wide range of traffic load. It divides the cycle into the sync, data and sleep periods as introduced by Sensor MAC [9]. However, in DW-MAC the sleep period is employed for data transmissions. DW-MAC uses a one-to-one proportional scheduling function to determine the start of data packet transmissions based on the start of the corresponding control packet in the data period. Therefore, nodes in DW-MAC can be involved in multiple traffic flows without causing data-data collisions in the sleep period. However, DW-MAC still incurs idle listening overhead in the data period, allows for data-ack collisions and saturates very fast when the traffic load increases.

In this paper we propose a new multi-hop MAC protocol, named MAC^2, which joins several techniques to overcome the mentioned disadvantages of DW-MAC. MAC^2's advanced characteristics are:

- MAC^2 utilizes an adaptive scheme that can adjust the length of the listening period in a cycle according to the traffic load.
- MAC^2 inherits the demand wakeup manner from DW-MAC to transmit data, but it optimizes the scheduling function to achieve minimum latency and guarantee collision freeness in the sleep period.
- MAC^2 employs a packet concatenation scheme which combines several packets into a bigger one to reduce control overhead.

The rest of the paper is organized as follows: In Sect. 2 we discuss related works and present MAC^2's design in Sect. 3. In Sect. 4, we present a latency analysis and give upper and lower bounds that hold for MAC^2 under low traffic conditions. Section 5 comprises the evaluation in which we illustrate MAC^2's performance under various scenarios. Finally, we conclude the paper in Sect. 6.

Manuscript received May 16, 2011.
Manuscript revised September 20, 2011.
[†]The author is with the Graduate University for Advanced Studies, Tokyo, 101–8430 Japan.
[††]The authors are with National Institute of Informatics, Tokyo, 101–8430 Japan.
a) E-mail: kienng@nii.ac.jp
b) E-mail: meis@nii.ac.jp
c) E-mail: kei@nii.ac.jp
DOI: 10.1587/transinf.E95.D.480

2. Related Work

Due to the limitation in battery capacity, energy efficiency is the primary goal in MAC protocol design for WSN applications. Initially, many proposed protocols focused on energy efficiency at the expense of other parameters (e.g., throughput, latency, fairness). Previous works [9], [10] reported that the major source of energy wastage is idle listening. Many solutions to the problem of idle listening have been proposed utilizing the technique of duty cycling that is originally from Sensor MAC (S-MAC) [9]. In this technique, each sensor node turns its radio on only periodically, alternating between active and sleeping states. The duty cycling protocols can be roughly divided into asynchronous and synchronous protocols.

Asynchronous MAC protocols such as B-MAC [11], X-MAC [12], RI-MAC [13], A-MAC [14], PW-MAC [15] and CyMAC [16] have advantages in deployment and configuration and are widely used in real applications and commercial sensor products because they don't suffer from synchronization overhead. However, they still suffer from the high latency that is caused by forwarding packets only one hop per cycle.

Recently, advanced synchronization protocols, e.g. [17], can help the sensor nodes synchronize accurately with low power consumption, so the synchronous protocols can relax the synchronous overhead. There are several approaches to reducing delivery latency, such as S-MAC with adaptive listening [18] and T-MAC [19]. In S-MAC with adaptive listening, a node that overhears a control packet (e.g., RTS or CTS) of another node's communication during the data period will wake up for a short time when the communication finishes. If this node is the next-hop along a multi-hop path, its neighbors can immediately forward the data packet to this node rather than waiting until the data period in the next operational cycle. T-MAC reduces latency by adaptively changing the ending time of a data period when there is no traffic transmission near the node. The data period is ended whenever both the physical and virtual carrier sensing find the channel idle for the given duration of the timeout. Both T-MAC and S-MAC with adaptive listening can generally forward a packet at most two hops within one operational cycle.

Another approach is D-MAC [20] which reduces latency only for data gathering in which multiple nodes try to send data to a sink node through a unidirectional tree of paths. However, D-MAC has the disadvantage that it makes specific assumptions on the communication pattern among nodes (tree-based).

In contrast to each of the above protocols multi-hop MAC protocols [7], [8], [21] support multi-hop transmission (more than two hops) in a single operational cycle. Like single-hop protocols they usually divide the cycle into the sync, data and sleep periods as introduced by Sensor MAC [8]. A synchronization protocol employs the sync period to align the start of the cycle among all nodes in the network. In the subsequent data period multi-hop MAC protocols let nodes with pending packets compete for corresponding time slots within the subsequent sleep period. Hence, the actual data is transmitted in the sleep period.

RMAC [7] adds cross-layer information to the traditional RTS packet to create a new efficient control packet named pioneer (PION). A PION can traverse multiple hops, play both RTS and CTS roles, and it has a scheduling function. Compared to S-MAC and S-MAC with adaptive listening, RMAC significantly reduces the end-to-end latency and uses much less power. RMAC-CS (RMAC with carrier sensing) [21] improves RMAC's performance under low traffic environments by adding a short carrier sensing period after the synchronization period. During the added period RMAC-CS utilizes carrier sensing as a binary signal that lets the nodes know the current traffic status. Subsequently, if there are no packets to transmit, the nodes turn off their radios to save energy.

While RMAC performs well in light traffic environments it still suffers from large control overhead and low throughput. In an attempt to support a wide range of traffic load RMAC was extended to Demand Wakeup MAC (DW-MAC). The main difference between the two is the way packets are scheduled in the sleep period. In RMAC, a traffic flow always starts at the beginning of the sleep period. Hence, there can only be one flow per cycle.

DW-MAC [8] uses the same control packet exchange idea as RMAC. In DW-MAC, a scheduling (SCH) packet replaces the PION packet, and the SCH packet is used to schedule the wake-up time of the nodes that will participate in data transmission during the sleep period. However, both RMAC and DW-MAC perform idle listening during the data period when there is no traffic in the network, and RMAC-CS will incur additional energy overhead in a high traffic environment. MAC^2 employs an adaptive method without adding an extra period to bypass these problems of multi-hop MAC protocols.

Moreover, WSNs also experience bursty and high traffic loads. DW-MAC can support multiple data flows at a node by using the mapping function. MAC^2 takes advantage of DW-MAC's mechanism and supports much higher traffic load by using the packet concatenation scheme.

3. MAC^2 Design

3.1 Overview of MAC^2

MAC^2 is a synchronous contention based protocol. The protocol employs Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to access the channel. The packet structure and the concepts of Distributed Inter-Frame Space and Short Inter-Frame Space are taken from IEEE 802.11, but otherwise MAC^2 does not depend on a particular standard. Similar to other duty cycling protocols, the operational cycle of MAC^2 contains three periods: Sync, Data, and Sleep with their lengths denoted as T_{Sync}, T_{Data}, and T_{Sleep}. In the Sync period, MAC^2 adopts an adaptive mecha-

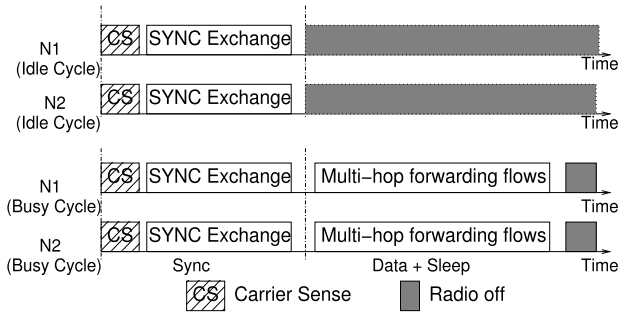


Fig. 1 SYNC exchange in busy and idle cycle.

nism which can adapt to the traffic load and lets nodes follow either a busy or an idle cycle. In the idle cycle nodes sleep to save energy after the sync period. In the busy cycle on the other hand they keep their radio on and exchange cross-layer control packets to reserve time slots for data transmissions in the subsequent Sleep period. MAC^2 inherits Demand Wakeup MAC's (DW-MAC's) [8] on-demand manner in which nodes are woken up during the Sleep period in order to transmit or receive a data packet.

3.2 Adaptive Scheme in Sync Period

MAC^2 uses the synchronization protocol proposed in S-MAC just like RMAC and DW-MAC. All nodes use the exchange of SYNC packets to choose and maintain the sleep/awake schedule. In addition, MAC^2 employs an adaptive method in which nodes can adjust their listening periods themselves according to the network traffic. When there is no data in the network, nodes follow an idle cycle with a short listening period (T_{Sync}), otherwise they follow a busy cycle with a long listening period ($T_{Sync} + T_{Data}$). We use the first bit of the SYNC packet to convey this information. When the bit is set, the new SYNC packet, called signaling SYNC, tells all the listeners to extend their listening periods. Thus, if a node has data to transmit, it constructs a signaling SYNC first and broadcasts it to its neighbors. When a node receives a signaling SYNC from one of its neighbors, it sets its clock to the long listening duration. After that, if the remainder of the Sync period is long enough to transmit another SYNC packet, the node will broadcast a new signaling SYNC to its neighbors. We assume that the length of the Sync period is long enough to rebroadcast the SYNC to the entire network, and each node sends only one SYNC packet. We also assume that signaling SYNC packets always win the channel over a normal SYNC (i.e. the first bit is not set). The assumption can be achieved by setting a smaller carrier sense duration for signaling SYNC transmissions. However, if the signaling SYNC does not win the channel, the protocol still works since it repeats the same process in the subsequent cycle.

Figure 1 illustrates the operation of MAC^2 in both an idle and a busy cycle. At the beginning of each Sync period, N1 and N2 implement carrier sense. If there is no traffic in the network, N1 and N2 go to sleep at the end of the Sync period; they follow the idle cycle. If node N1 has data to transmit, it sends a signaling SYNC packet to its neighbors including N2. The nodes follow the busy cycle.

This adaptive method can be applied to other synchronous protocols, e.g. RMAC, as we have done in a previous work [22]. The results show that by using an adaptive method RMAC consumes less energy while achieving comparable latency.

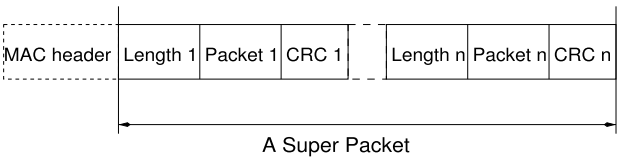


Fig. 2 Super packet structure.

3.3 Data Forwarding in a Busy Cycle

In this subsection, we introduce the concatenation scheme which concatenates several packets in the queue before scheduling the transmission, and MAC^2's operation in a busy cycle.

3.3.1 Concatenation Scheme

In all duty cycling protocols nodes often have to queue pending packets for an extended amount of time since the sleep period is typically much longer than the active period. This is especially true since although the radio is switched off periodically the sensors are for most sensor network applications kept on at all times. The resulting frequent packet bursts suggest packet concatenation as a means of both improving energy efficiency and latency. Two properties of WSNs make concatenation especially attractive. Firstly, all packets are predominantly addressed to one and the same sink, thereby fulfilling the major requirement of packet concatenation. Secondly, data packets are usually small and therefore the gains from reducing control packet overhead are comparatively large.

Similar to DW-MAC and S-MAC, the data packet usually contains three parts: payload, length field and CRC field. When the MAC-layer processes the data packet, it attaches the MAC header. As we mentioned before all MAC headers share the same destination address in single sink networks since all data is directed towards the sink. In our scheme, we concatenate several packets into a super packet before adding the MAC-header to it. We denote the length of the resulting super packet as l_{SP}, which is always less than or equal to the concatenation threshold l_{TH}.

Figure 2 shows the structure of a super packet. Super packets are constructed from one up to n smaller packets and contain, for each encapsulated packet, its payload plus the length and CRC fields. Although these fields could theoretically serve other purposes as well we only use the length field to reconstruct the original packets from a super packet at the receiver. In case a super packet can not be transmitted

because the node does not win the channel, its packets are requeued for later transmission.

3.3.2 Multi-Hop Forwarding in a Busy Cycle

In a busy cycle, nodes keep their radio on after the Sync period. At the start of the Data period, nodes with pending data construct a super packet (SP) from one or more packets in their queue and set up the multi-hop flow by sending a control packet. The control packet, named scheduling packet (SCH) as in DW-MAC, is constructed by adding cross-layer fields (next-hop, destination, and number of hops) from the routing layer to the IEEE 802.11 RTS [23] packet. If the multi-hop flow is successfully initialized, node i in the flow calculates its wakeup time in the next Sleep period following the proportional mapping function:

$$\frac{T_S^i}{T_D^i} = R \tag{1}$$

where T_D^i is the duration from the beginning of the Data period to the starting moment of the SCH transmission, T_S^i is the duration from the beginning of the Sleep period to the starting moment of the super packet transmission, and R is the mapping function value. In DW-MAC the value of R is $R_{org} = \frac{T_{Sleep}}{T_{Data}}$, but in MAC^2 we use the R_{min} value that we are going to introduce in the next section in order to minimize latency. By using a proportional mapping function, we can schedule more than one SCH exchange in the Data period and therefore more than one data flow.

We illustrate the operation of MAC^2's multi-hop scheduling and multi-hop forwarding in the simple 4-node scenario in Fig. 3. Node N1 has data pending in its queue; it uses the concatenation scheme, as explained in the previous section, to construct a super packet. N1 uses the CSMA/CA protocol to contend for the channel. If N1 wins, it will start by sending the 1st SCH after a Distributed Inter-Frame Space (DIFS). The intermediate nodes N2 and N3 relay the SCH packet and at each node, the time taken to process the SCH is one Short Inter-Frame Space (SIFS). During the relay process, the SCH packet serves as an RTS for the next hop and a CTS for the previous hop. In conclusion, the SCH packet from the source node only serves as an RTS and the SCH packet of the destination node only serves as a CTS. Based on R_{min}, nodes will wake up at the correct time to transmit/receive data in the Sleep period.

3.4 Achieving Minimum Latency

Using the proportional mapping (1) with $R = \frac{T_{Sleep}}{T_{Data}}$, any two data packets (in this case SP packets) never collide at an intended receiver as proven in the original DW-MAC paper [8]. However, there is a possibility of collision between SP and ACK packets. In this subsection we present the conditions of the mapping function to guarantee that there will be no SP-ACK collisions. In the following analysis and the remainder of this paper we assume that the bit error rate is zero for all transmissions.

We investigate a multi-hop relay in a busy cycle. Node i and its neighbor, node $(i + 1)$, always satisfy the following equation:

$$R = \frac{T_S^i}{T_D^i} = \frac{T_S^{i+1}}{T_D^{i+1}} \tag{2}$$

The intervals between the time transmitting SCH packets and SP packets of two nodes, ΔT_S^i, ΔT_D^i, are defined as follows: $\Delta T_S^i = T_S^{i+1} - T_S^i$ and $\Delta T_D^i = T_D^{i+1} - T_D^i$.We can obtain R by using the property of equal fractions series from (2):

$$R = \frac{\Delta T_S^i}{\Delta T_D^i} \tag{3}$$

However, when a node receives an SCH packet from its previous hop in the Data period, it needs a duration of $SIFS$ to process and relay the packet to its next hop. Assume l_{SP}, l_{ACK}, l_{SIFS} to be the length of SP, ACK, and SIFS, respectively, hence $\Delta T_D^i = l_{SCH} + l_{SIFS}$. To guarantee no SP-ACK collision in the Sleep period, for example at node $(i + 1)$, node $(i + 1)$ should start to transmit its SP packet after node

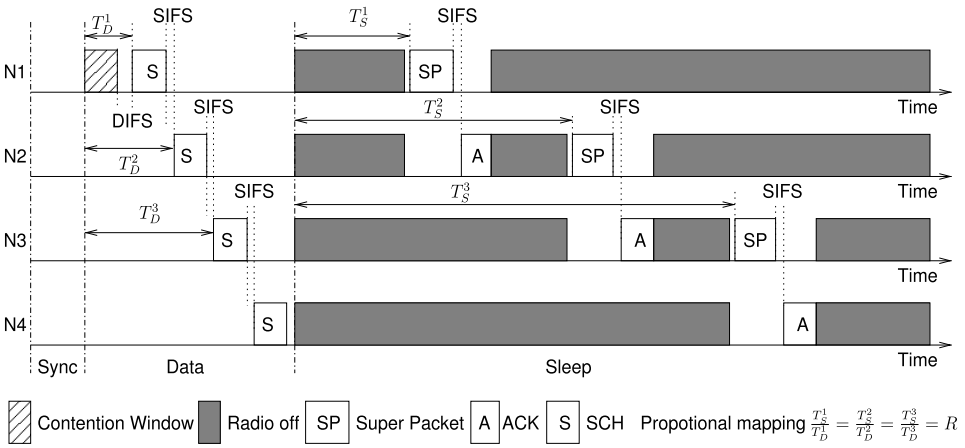


Fig. 3 MAC^2 in a four-node scenario.

i receives its ACK.

$$\Delta T_S^i \geq l_{ACK} + l_{SP} + l_{SIFS} + \delta \tag{4}$$

δ can be considered as the state transition time, and it is negligible compared with other timing parameters. Using (3) and (4) we can get the condition which guarantees no SP-ACK collision with all possible length of SP in a multi-hop flow:

$$R \geq \frac{l_{ACK} + l_{TH} + l_{SIFS}}{l_{SCH} + l_{SIFS}} \tag{5}$$

In the following we will present a theorem and proof for a collision free receiver:

Theorem 1: There will be no collision at the intended receivers in the Sleep period if R satisfies condition (5).

Proof 1: If the nodes are the intended receivers in the Sleep period, that means their SCH packets have been successfully exchanged in the previous Data period. We then prove the above theorem by contradiction. Assume that there is a collision at a receiver i. If there is one flow in the network we can easily find that it can only be an SP-ACK collision. And with (5) that collision does not occur. If there is a collision between two flows at node i, node i takes part in two SP-ACK transmissions of two neighbors $n1$, $n2$. We denote time to start to transmit these two SP packets (SP1 and SP2) by T_S^{n1} and T_S^{n2}. Since there is a collision at node i, and $\max\{l_{SPj}\} \leq l_{TH}$ $(j = 1, 2)$:

$$|T_S^{n1} - T_S^{n2}| < l_{TH} + l_{SIFS} + l_{ACK} \tag{6}$$

If node i is the intended receiver of two data packets it means that it successfully relayed two SCH packets (SCH1, SCH2) in the previous Data period. Assuming $T1_D^i = T_D^{n1} + l_{SIFS} + l_{SCH}$, and $T2_D^i = T_D^{n2} + l_{SIFS} + l_{SCH}$. Because the node has to wait at least for the confirmation SCH in a flow before joining another flow, we have:

$$|T1_D^i - T2_D^i| \geq l_{SIFS} + l_{SCH} \tag{7}$$

From (6) and (7) we have:

$$R = \frac{|T_S^{n1} - T_S^{n2}|}{|T_D^{n1} - T_D^{n2}|} < \frac{l_{ACK} + l_{TH} + l_{SIFS}}{l_{SCH} + l_{SIFS}} \tag{8}$$

Hence, we can conclude that the transmissions are collision free at any intended receiver.

We set $R_{min} = \frac{l_{ACK}+l_{TH}+l_{SIFS}}{l_{SCH}+l_{SIFS}}$ and show that among all values of R, R_{min} gives the minimum delay. The delivery latency is proportional to the number of hops from the source node. In duty cycling protocols, the data transmission can only be initialized during the active period of the radio. Thus, a newly generated packet is initially delayed by $T_{Cycle}/2$ on average (denoted as T_{Init}), $T_{Cycle} = T_{Sync} + T_{Data} + T_{Sleep}$.

Assuming that the destination is h hops away from the source node, the average delivery latency of a node h hops away from the source node $L(h)$ is:

$$L(h) = T_{Init} + N_C(h, N) \times T_{Cycle} + T_{last}(h, N) \tag{9}$$

Here, N is the number of hops a packet passes in one cycle and $N_C(h, N)$ is the number of cycles a packet passes before the last cycle, and $N_C(h, N)$ is specified as follows:

$$N_C(h, N) = \begin{cases} \lfloor \frac{h}{N} \rfloor & \text{if } h\%N > 0 \\ \lfloor \frac{h}{N} \rfloor - 1 & \text{otherwise} \end{cases} \tag{10}$$

where $\lfloor . \rfloor$ is the floor function and % is the modulo operation. $T_{last}(h, N)$ is the latency in the last cycle and can be calculated as:

$$T_{last}(h, N) = T_S^{N_{last}(h,N)} + l_{SP} + l_{SIFS} + l_{ACK} \tag{11}$$

where $N_{last}(h)$ is the number of hops that the packet passed in the last cycle.

$$N_{last}(h, N) = \begin{cases} h\%N & \text{if } h\%N > 0 \\ N & \text{otherwise} \end{cases} \tag{12}$$

Since SCH relays are similar for all R, from (1)(9–12) we obtain that the minimum latency is achieved when R is at a minimum. The same proof is shown in our previous work for DW-MAC [24]. Since we use l_{TH} to express the packet length in the mapping function, the delivery latency of a single packet becomes longer in case $l_{SP} < l_{TH}$. However, this slight inefficiency exists only in low traffic environments. Once the traffic load increases, l_{SP} will often be equal to l_{TH} and the difference in latency will become negligible. In any case, the throughput per cycle is not affected.

4. Upper and Lower Bound of Delivery Latency

In this section we give bounds of delay under low traffic conditions. In MAC^2 and other multi-hop protocols, the number of hops (N) that a data packet can reach in a duty cycle mainly depends on T_{Data}. The equation below expresses the relationship between N and T_{Data}, which can be considered as the most important factor in the protocol.

$$T_{Data} = CW + l_{DIFS} + N \times (l_{SIFS} + l_{SCH}) \tag{13}$$

Note that because of the small SCH packet, the duration of the contention window (CW) could be used to send the SCH. In this case, the maximum and minimum values of N can be determined as: $N_{max} = \lfloor \frac{T_{Data}}{l_{SIFS}+l_{SCH}} \rfloor$ and $N_{min} = \lfloor \frac{T_{Data}-CW}{l_{SIFS}+l_{SCH}} \rfloor$.

First, we investigate the single packet scenario in which the destination is h hops away from the source node. From (9) the minimum delivery latency of a node h hops from the source node $L(h)$ can be calculated as follows:

$$L(h) = T_{Init} + N_C(h, N_{max}) \times T_{Cycle} + T_{last}(h, N_{max}) \tag{14}$$

In addition the $T_S^{N_{last}(h)}$ can be calculated from (1): $T_S^{N_{last}(h)} = T_D^{N_{last}(h)} \times R_{min}$, and $T_D^{N_{last}(h)}$ can be calculated from T_D^1, which is the time of transmission of the first SCH in the last cycle.

$$T_D^{N_{last}(h)} = l_{DIFS} + (N_{last}(h) - 1) \times (l_{SIFS} + l_{SCH}) \quad (15)$$

On the other hand, the minimum value of T_{Init} is zero, so from (14)(15), we have the lower bound of the latency to be:

$$\begin{aligned} L_{lower}(h) = {} & N_C(h, N_{max}) \times T_{Cycle} + ((N_{last}(h, N_{max}) \\ & -1) \times (l_{SIFS} + l_{SCH})) \times R_{min} + l_{DIFS} + \\ & l_{SP} + l_{SIFS} + l_{ACK} \end{aligned} \quad (16)$$

Now we find the value of the upper bound. A packet incurs the maximum latency when it can traverse only N_{min} hops in a cycle. Moreover, T_{Init} will be at its maximum when the packet is generated at the beginning of the Data period in a idle cycle. Thus, the upper bound of latency is:

$$\begin{aligned} L_{upper}(h) = {} & T_{Sleep} + T_{Data} + N_C(h, N_{min}) \times T_{Cycle} + \\ & (l_{DIFS} + (N_{last}(h, N_{min}) - 1) \times (l_{SIFS} + \\ & l_{SCH})) \times R_{min}) + l_{SP} + l_{SIFS} + l_{ACK} \end{aligned} \quad (17)$$

Secondly, we investigate a scenario in which M transmitters try to send a packet to one sink, and the number of retransmissions allowed is $Nret$. We assume no packet is generated while the previously generated packets are being transmitted. When a node fails in the channel contention process, it waits until the next cycle to contend again. Hence, the maximum latency will be at the node which wins the channel last. After $min\{Nret, M\}$ cycles, it will start to setup the data flow:

$$L_{upper}^{Nret,M}(h_{max}) = \min\{Nret, M\} \times T_{Cycle} + L_{upper}(h_{max}) \quad (18)$$

Here, h_{max} is the maximum hop-count to the destination among the M transmitters. If M is greater than $Nret$, $(M - Nret)$ nodes will fail to transmit data.

Now let us investigate a scenario in which there are some SCH packets that failed to be transmitted to next hops. If a node receives a SCH packet but cannot relay it to its neighbor, it receives the corresponding data packet and keeps it in its queue. The reason is that there is no collision during this Sleep period. The node can be considered as the transmitters in the upcoming cycle. If the scenario contains several transmitters, we can use (18) to investigate the upper bound of latency. Since this analysis is applied for a low traffic environment, we can assume there will be few failed SCH packets. The upper bound can be derived from (18) by replacing M with M_{max} the maximum number of transmitters or the maximum number of neighboring nodes for the specific topology.

5. Evaluation Results

We use the network simulator ns-2 [25] to evaluate our analysis under a simple scenario and the protocol's performance under various scenarios. Each sensor node had a single omni-directional antenna through which the combined free space and two-ray ground reflection radio propagation models were employed. Key networking parameters are shown in Table 1, power consumption parameters were set to typical values for a Mica2 radio (CC1000). The 250 m transmission range and the 550 m carrier sensing range were modeled after the 914-MHz Lucent WaveLAN DSSS radio interface; although not typical for a sensor node, we used these parameters to make our results comparable to DW-MAC. In our evaluation of power efficiency, we focused on the energy consumed by radios, but ignored the energy consumed by other components such as CPU and memory [26]. The original size of one data packet is 50 bytes, which leads to a transmission time of $l_{DATA} = 43\,ms$. The super packet threshold is set to 300 bytes or $l_{TH} = 243\,ms$. Note that in our simulations the transmission time of a packet is calculated the same way as described in the RMAC and DW-MAC papers as follows:

$$l_{packet} = \frac{packetsize}{EB} + l_{preamble} + l_{processingtime} \quad (19)$$

where $EB = 10\,Kbps$ stands for the effective bandwidth, $l_{preamble} = 2\,ms$ for the transmission time of a preamble, and $l_{processingtime} = 1\,ms$ for the processing time. The capacity of the queue between the network and link layers (ifq) is set to 2500 bytes and other values are kept similar to those used in the evaluation of DW-MAC. All time duration parameters are shown in Table 2.

To simplify our evaluations, we did not include the routing traffic in the simulations. We also assumed that there was a routing protocol deployed to provide the shortest path between any two nodes and ensured that the networks we used in our simulations are connected networks. As it has been shown in DW-MAC's original paper, DW-MAC outperforms other protocols in a wide range of traffic load conditions. Therefore, we based our protocol on DW-MAC and also used it as a benchmark in our evaluation.

5.1 Latency Bounds Evaluations

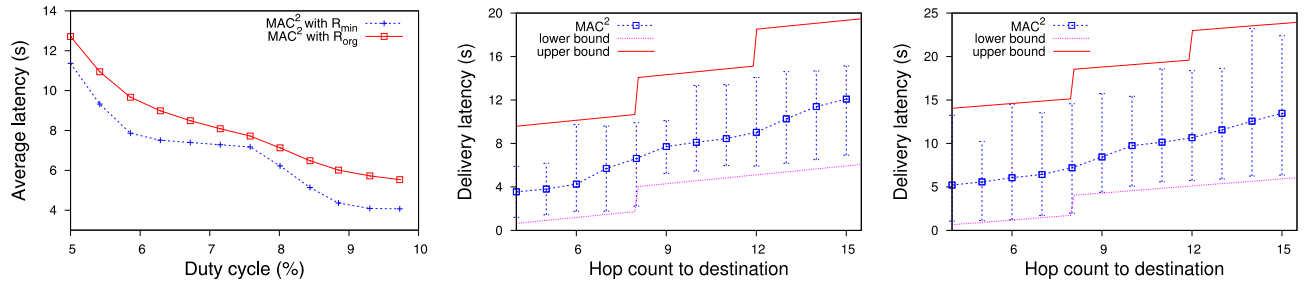
For the sake of simplicity, we used a chain scenario with 15 nodes. The distance between two neighboring nodes is 200 meters. We compared the performance of MAC^2 with two

Table 1 Networking parameters.

Rx Power	22.2 mW	Sleep Power	3 μW
State Transition Power	31.2 mW	Tx Power	31.2 mW
Idle Power	22.2 mW	Tx Range	250 m
Carrier Sensing Range	550 m	SIFS	5 ms
Contention Window (CW)	64 ms	DIFS	10 ms
Retry Limit	5	l_{DATA}	43 ms
l_{ACK}	11 ms	l_{SCH}	14.2 ms
l_{TH}	243 ms	ifq	2500 bytes

Table 2 Time duration parameters.

	T_{Cycle}	T_{Sync}	T_{Data}	T_{Sleep}
DW-MAC	4465 ms	55.2 ms	168 ms	4241.8 ms
MAC^2	4465 ms	55.2 ms	168 ms	4241.8 ms



(a) Minimum latency (b) Latency bounds in single source scenario (c) Latency bounds in two sources scenario

Fig. 4 Latency bounds evaluation results.

values of R: R_{min} and R_{org} (where $R_{org} = T_{Sleep}/T_{Data}$). As we discussed before, the number N in (11) is one of the key factors in designing the multi-hop MAC protocol. We kept the value of T_{Cycle} at 4465 ms and varied T_{Data} in steps of $l_{SIFS} + l_{SCH}$ to allow more hops to be passed in one cycle. Hence, the duty cycle value (DC) also varied. Note that the DC is calculated as follows: $DC = (T_{Sync} + T_{Data})/T_{Cycle}$.

In this evaluation, the source node 0 at the beginning of the chain sends to the destination node 14 which is the farthest node from the source, once every 30 seconds, and the total number of packets sent is one hundred. We measured the average delivery latency and plotted it in Fig. 4 (a). The average delivery latency decreased when increasing DC. With a DC smaller than 6% almost all packets needed three cycles to reach the destination. Furthermore, with a DC in the range 7% – 10% they needed two cycles and one cycle otherwise. However, in all cases, we found that MAC^2 (with R_{min}) has lower latency than MAC^2 with R_{org}.

We evaluated the lower and upper bounds of latency in various scenarios. First, we considered the chain scenario with a single source. The length of the chain varied from 4 to 15 hops, and the source sent a packet every 30 seconds to the destination at the end of the chain. We kept the duty cycle at 5%, the default value in MAC^2. Fig. 4 (b) and Fig. 4 (c) show the simulations' results and the numerical bounds' results. The graphs of the lower and upper bounds show that a packet can reach a node 8 hops away in the best case and 5 hops away in the worst case, respectively. We can see that the maximum, minimum, and average values of the delivery latency are in the range of the upper and lower bounds. The difference between the maximum and minimum values depends on T_{Init}. The delay never reached the upper bound because there were no collisions in any of the cycles. This evaluation can be a guide for us to investigate the hop length to the sink after deployment and to choose a suitable duty cycle to satisfy the delay requirements.

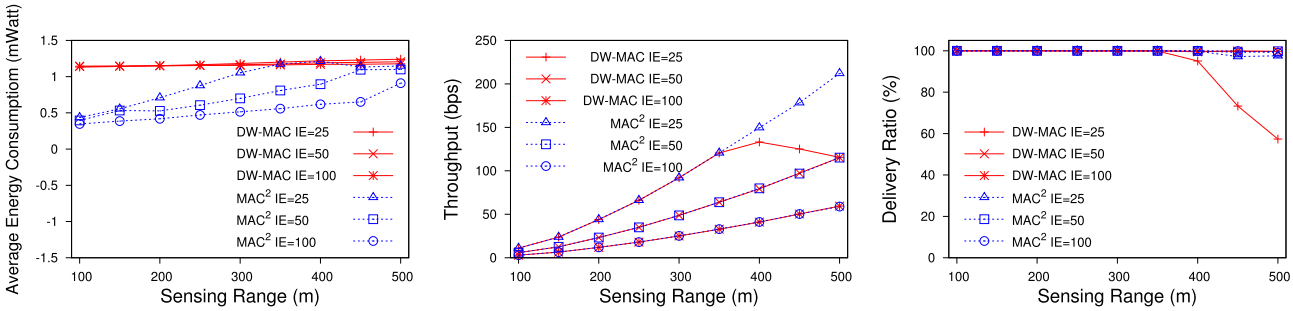
After that, we investigated a scenario with two transmitters. We assumed that between node 0 and node 1 there was an event that occurred during each 30-second interval. The sensing range of both nodes was 200 meters. After sensing the event, nodes sent a packet to the sink node, the farthest node in the chain. The chain length varied from 4 to 15 hops. Node 0 and node 1 contended for the channel every 30 seconds. That is, when there was an event, only one node could send a packet; the other one had to wait until the next cycle. Since $Nret = 5$ in this case, we have $min(M, Nret) = 2$. We plotted the upper bound as given in (18). The results shown in Fig. 4 (c) lead us to the same conclusion as in the previous experiment.

5.2 Grid Scenario

In this scenario we use a grid of 7×7 cells. Each node is 200 meters apart from its direct neighbors and the sink node is at the center of the grid. We use the Random Correlated-Event (RCE) traffic model introduced in [8], which creates events at random locations. All nodes that can sense a given event, i.e. nodes for which the event lies within the sensing range, generate one data packet in response. We investigated scenarios with all nodes having the same sensing range of 100, 150, 200, 250, 300, 350, 400, 450 and 500 meters. For each of the sensing range's value we ran a simulation with 200 events. With the specific number of events we set the interval between two events (IE) to 25, 50, and 100 seconds and compared the performance between MAC^2 and DW-MAC.

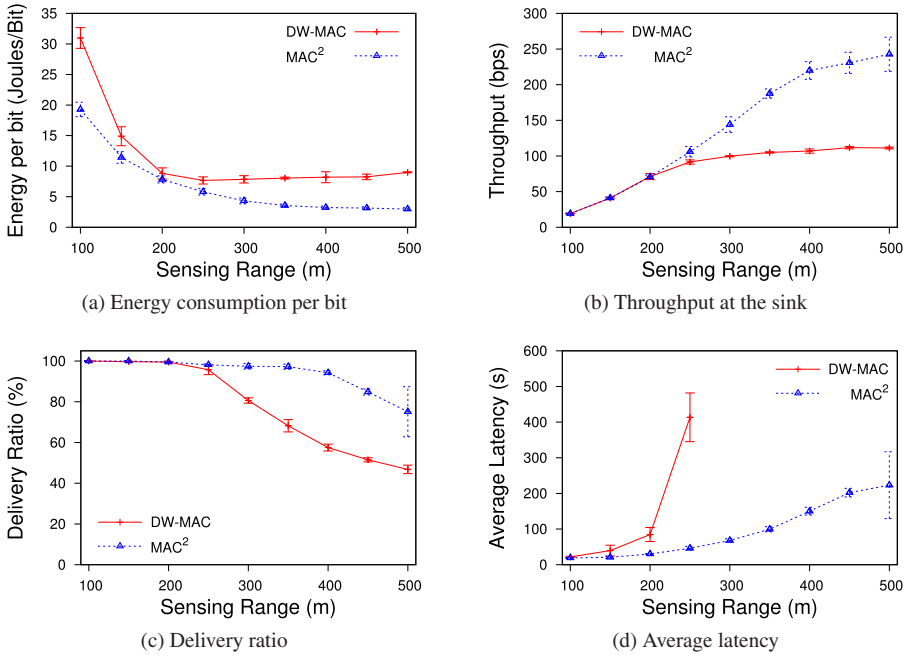
Figure 5 (a) shows the average energy consumption in the evaluation. In DW-MAC, the average energy consumption increases slightly when the sensing range increases as well as the value of IE. In MAC^2, when the sensing range increases the average energy consumption increases but the value is lower than in DW-MAC because more data packets are generated and the nodes have to keep their radio on for more cycles. We come to the same conclusion when IE increases. If IE is large enough (IE = 50, 100 seconds), most packets generated by a previous event reach the sink before the next event occurs and the adaptive mechanism can save power in some idle cycles. Therefore, we can conclude that MAC^2 achieves better energy efficiency than DW-MAC.

When the IE value is at 50 or 100 seconds, the throughput at the sink keeps increasing along with the sensing range as shown in Fig. 5 (b) and Fig. 5 (c). But when IE is smaller, there are still some packets that can not go to the sink when new events occur, DW-MAC starts to drop packets when the sensing range is at 350 meters, and the throughput slightly decreases due to packet collision. But in the case of MAC^2, the concatenation scheme is applied to handle this situation, the throughput continuously increases and the delivery ratio is kept at almost at 100%.



(a) Average energy consumption in grid scenario (b) Throughput at the sink (c) Delivery ratio

Fig. 5 Evaluation results in 49-node grid scenario.



(a) Energy consumption per bit (b) Throughput at the sink

(c) Delivery ratio (d) Average latency

Fig. 6 Evaluation results in random network scenario.

5.3 Random Network Scenario

We also evaluated MAC^2's performance under a random network scenario. A network contains 100 nodes which are randomly deployed in the square area 1500 × 1500 meter. The sink node is located at the top right corner of the area. We generated 10 different scenarios with 200 events each inside the area. We slightly modified the RCE traffic model to make it more realistic. Each event is randomly generated and the interval between two subsequent events was set to a random interval between 0 and 50 seconds. After the sink received the final packet, we kept the network running for another 100 seconds and compared the performance of MAC^2 against DW-MAC. The evaluation results are shown in Fig. 6, each average value is calculated from the results of 10 random runs and the error bars show the 95 % confidence interval.

Figure 6 (a) shows the energy consumption per bit which is calculated by the ratio of total energy consumption to the throughput. We use this value to reflect the energy efficiency of the network. When the sensing range is smaller than 200 meters, traffic load is low and MAC^2 is more energy efficient than DW-MAC since MAC^2 employs an adaptive mechanism, the nodes keep their radio off when there is no data and therefore save energy. At this level of traffic load, the two protocols share the same throughput performance and handle 100% delivery ratio. When the sensing range increases, the traffic load becomes heavier and MAC^2 still consumes less power than DW-MAC. The reason is the concatenation scheme is applied and the control overhead is largely reduced. Moreover, since there are fewer packets in MAC^2's network, the collision probability is smaller.

The throughput performance of the two protocols is shown in Fig. 6 (b). When the sensing range grows larger than 250 meters, in MAC^2 the throughput keeps increasing. In the case where the sensing range is 500 meters, MAC^2's

throughput is even 2.5 times higher than DW-MAC's. That is because of the effectiveness of the concatenation scheme. In DW-MAC, although more packets are generated, the amount of data received at the sink stays almost the same. That means more packets are dropped. The same conclusion can be drawn from Fig. 6 (c), the delivery ratio sharply decreases in DW-MAC's case when the traffic load gets higher. That means DW-MAC's network reaches the saturated state faster then MAC^2's network does. Figure 6 (d) shows the end-to-end latency of the two protocols where MAC^2 also outperforms DW-MAC. When the sensing range is larger than 250 meters DW-MAC's latency increases excessively due to the saturation of nodes' buffers. Therefore, those values are not shown in the figure. Note that, in MAC^2 we use the earliest generated packet in the super packet as the start point of super packet transmission. We can conclude that MAC^2 is more energy, latency and throughput efficient than DW-MAC for all investigated traffic loads.

6. Conclusion

In this paper we propose MAC^2, an energy efficient, high throughput multi-hop MAC protocol. The protocol employs multi-hop forwarding to overcome the latency burden of duty cycling. Besides that, it is optimized to perform well in a wide range of traffic load conditions. MAC^2 derives its wake up on demand manner from DW-MAC to support multiple traffic flows in a single cycle. Furthermore, MAC^2 adopts an adaptive mechanism that adjusts the listening period according to traffic load, minimizing idle listening. Finally, MAC^2 utilizes a concatenation scheme that can combine several queued packets for the same destination and send them as one super packet. Throughput is significantly increased and control overhead is reduced and evaluations of MAC^2 show that it outperforms other state-of-the-art multi-hop protocols.

In our future work we will investigate the efficiency of our protocol in noisy environments. Currently, we focus on two possible solutions to overcome the drawbacks of the concatenation scheme. The first solution would be that MAC^2 adopts a bit-error-rate (BER) adaptive method in which nodes send a small or a large packet based on a BER threshold. The second would be that MAC^2 separates a received superpacket into several fragments (or original packets), checks for errors in each fragment and resends only the fragments with errors.

References

[1] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," Proc. 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp.263–270, 1999.

[2] A. Ledeczi, A. Nadas, P. Volgyesi, G. Balogh, B. Kusy, J. Sallai, G. Pap, S. Dora, K. Molnar, M. Maroti, and G. Simon, "Countersniper system for urban warfare," TOSN, vol.1, no.2, pp.153–177, 2005.

[3] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications, WSNA '02, pp.88–97, 2002.

[4] V. Raghunathan, C. Schurgers, S. Park, M. Srivastava, and B. Shaw, "Energy-aware wireless microsensor networks," IEEE Signal Process. Mag., vol.19, no.2, pp.40–50, 2002.

[5] M. Ali, U. Saif, A. Dunkels, T. Voigt, K. Römer, K. Langendoen, J. Polastre, and Z. Uzmi, "Medium access control issues in sensor networks," ACM SIGCOMM Computer Communication Review, vol.36, no.2, pp.33–36, April 2006.

[6] B. Yahya and J. Ben-Othman, "Towards a classification of energy aware mac protocols for wireless sensor networks," Wireless Communications and Mobile Computing, vol.9, pp.1572–1607, 2009.

[7] S. Du, A. Saha, and D. Johnson, "RMAC: a Routing-Enhanced Duty-Cycle MAC protocol for wireless sensor networks," Proc. 26th IEEE INFOCOM, pp.1478–1486, 2007.

[8] Y. Sun, S. Du, O. Gurewitz, and D.B. Johnson, "DW-MAC: A low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks," Proc. 9th ACM MobiHoc, pp.53–62, 2008.

[9] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," Proc. 21st IEEE INFOCOM, pp.1567–1576, 2002.

[10] L.M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," Proc. 20th IEEE INFOCOM, pp.1548–1557, 2001.

[11] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," Proc. 2nd SenSys, pp.95–107, 2004.

[12] M. Buettner, G.V. Yee, E. Anderson, and R. Han, "X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks," Proc. 4th ACM SenSys, pp.307–320, 2006.

[13] 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 SenSys, pp.1–14, 2008.

[14] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.J.M. Liang, and A. Terzis, "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," Proc. 8th ACM SenSys, pp.1–14, 2010.

[15] L. Tang, Y. Sun, O. Gurewitz, and D.B. Johnson, "Pw-mac: An energy-efficient prediction-wakeup mac protocol for wireless sensor networks," Proc. 30th IEEE INFOCOM, pp.1305–1313, April 2011.

[16] Y. Peng, Z. Li, D. Qiao, and W. Zhang, "Delay-bounded mac with minimal idle listening for sensor networks," Proc. 30th IEEE INFOCOM, pp.1314–1322, April 2011.

[17] A. Rowe, V. Gupta, and R.R. Rajkumar, "Low-power clock synchronization using electromagnetic energy radiating from AC power lines," Proc. 7th ACM SenSys, pp.211–224, 2009.

[18] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," IEEE/ACM Trans. Netw., vol.12, pp.493–506, 2004.

[19] T. van Dam and K. Langendoen, "An adaptive energy efficient and low-latency mac for data gathering in wireless sensor networks," Proc. 1st ACM SenSys, pp.171–180, 2003.

[20] G. Lu, B. Krishnamachari, and C.S. Raghavendra, "An adaptive energy-efficient and low-latency mac for data gathering in sensor networks," Proc. 18th IEEE International Parallel and Distributed Processing Symposium, IPDPS'04, pp.224–231, 2004.

[21] K. Nguyen and Y. Ji, "Using carrier sensing to improve energy efficiency of mac protocol in sensor networks," Proc. 10th IEEE International Symposium on Communications and Information Technologies, ISCIT '10, pp.432–436, 2010.

[22] K. Nguyen and Y. Ji, "AM-MAC: an energy efficient, adaptive multi-hop mac protocol for sensor networks," Proc. 6th ACM International Wireless Communications and Mobile Computing Conference, IWCMC '10, pp.432–436, 2010.

[23] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." IEEE Standard 802.11, June 1999.

[24] K. Nguyen and Y. Ji, "Achieving minimum latency in multi-hop mac protocol for wireless sensor networks," Proc. IEEE 73rd Vehicular

Technology Conference, VTC Spring '11, pp.1–5, May 2011.
[25] "The network simulator – ns2 http://www.isi.edu/ns/index.html."
[26] V. Shnayder, M. Hempstead, B. rong Chen, G.W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," Proc. 2nd ACM SenSys, pp.188–200, 2004.

Kien Nguyen received a B.E in Electronics and Telecommunication in 2004 from Hanoi University of Technology, Vietnam. He is currently PhD student at The Graduate University for Advanced Studies. His research interests include design and performance evaluation of communication protocols for wireless sensor networks, quality of service provisioning in wired and wireless networks.

Ulrich Meis received a diploma in Computer Science in 2006 from RWTH Aachen University, Germany, and subsequently entered the PhD course. He is currently a visiting researcher at the National Institute of Informatics, Tokyo, Japan. His research interests include wireless sensor networks, wireless mesh networks and distributed file systems.

Yusheng Ji received B.Eng., M.Eng. and D.Eng. in electrical engineering from the University of Tokyo in 1984, 1986 and 1989 respectively. She jointed the National Center for Science Information Systems in 1990. She is currently an Associate Professor at the National Institute of Informatics, and the Graduate University for Advanced Studies. Her research interests include network architecture, traffic control and performance analysis for quality of service provisioning in wired and wireless networks. She is also a member of IPSJ and IEEE.