A Hybrid MAC Protocol for Emergency Response Wireless Sensor Networks

Lanny Sitanayah[a], Cormac J. Sreenan[a], Kenneth N. Brown[b]

[a] *Mobile & Internet Systems Laboratory, Department of Computer Science, University College Cork, Ireland*
[b] *Cork Constraint Computation Centre, Department of Computer Science, University College Cork, Ireland*

Abstract

We introduce ER-MAC, a novel hybrid MAC protocol for emergency response wireless sensor networks. It tackles the most important emergency response requirements, such as autonomous switching from energy-efficient normal monitoring to emergency monitoring to cope with heavy traffic, robust adaptation to changes in the topology, packet prioritisation and fairness support. ER-MAC is designed as a hybrid of the TDMA and CSMA approaches, giving it the flexibility to adapt to traffic and topology changes. It adopts a TDMA approach to schedule collision-free slots. Nodes wake up for their scheduled slots, but otherwise switch into power-saving sleep mode. When an emergency occurs, nodes that participate in the emergency monitoring change their MAC behaviour by allowing contention in TDMA slots to achieve high delivery ratio and low latency. In its operation, ER-MAC prioritises high priority packets and sacrifices the delivery ratio and latency of the low priority ones. ER-MAC also guarantees fairness over the packets' sources and offers a synchronised and loose slot structure to allow nodes to join or leave the network. Simulations in ns-2 show the superiority of ER-MAC over Z-MAC, a state-of-the art hybrid MAC protocol, with higher delivery ratio, lower latency, and lower energy consumption. When a cluster of nodes in the network detects fire, nodes with ER-MAC deliver twice as many high priority emergency packets and four times faster than Z-MAC. This is achieved by ER-MAC with only one fifth as much energy as Z-MAC.

Email addresses: ls3@cs.ucc.ie (Lanny Sitanayah), cjs@cs.ucc.ie (Cormac J. Sreenan), k.brown@cs.ucc.ie (Kenneth N. Brown)

Keywords: MAC Protocol, Wireless Sensor Networks, Fire Emergency
2000 MSC: 68-68M12

1. Introduction

Wireless Sensor Networks (WSNs) for emergency applications, such as fire, flood and volcano monitoring, must be traffic and topology adaptive. The communication protocol can be delay tolerant during normal monitoring and designed for energy efficiency. However, when an emergency event occurs, energy efficiency is less important than high packet delivery ratio and low latency, and the communication protocol should adapt in response. The protocol must also be able to prioritise high priority packets, as they normally contain important sensed data and require timely delivery. In addition, the protocol must support fairness over the packets' sources. Fairness is important when a hazard occurs, so the sink can receive complete information from all sensor nodes in the network to monitor the spread of the hazard.

Some traffic and topology adaptive Medium Access Control (MAC) protocols have been designed. S-MAC [1], T-MAC [2], B-MAC [3] and X-MAC [4] are contention-based protocols that adapt to both traffic and topology changes, but suffer from collisions, idle listening and overhearing. Another contention-based protocol that satisfies the objectives for emergency response is MaxMAC [5]. However, this protocol does not support packet prioritisation and and does not guarantee fairness. Hybrid MAC protocols such as Z-MAC [6], Funneling-MAC [7] and BurstMAC [8] can adapt to traffic and topology changes as well as guarantee fairness, but they do not support packet prioritisation.

In this paper, we propose ER-MAC, a hybrid MAC protocol for emergency response WSNs. While our scenario assumption is the fire monitoring in buildings, this protocol is also useful in a range of WSN emergency applications. The contributions of this paper are:

- ER-MAC allows contention in TDMA slots to cope with large volumes of traffic. This scheme trades energy efficiency for higher delivery ratio and lower latency.
- ER-MAC maintains two priority queues to separate high priority packets from low priority packets.

- ER-MAC support fairness so the sink can receive complete information from all sensor nodes in the network.
- ER-MAC offers a synchronised and loose slot structure, where nodes can modify their schedules locally. This allows nodes to join or leave the network easily.
- Simulation results validate ER-MAC's performance, which outperforms Z-MAC [6] with higher delivery ratio and lower latency at low power consumption.

The remainder of this paper is organised as follows. In Section 3, we review the related work on traffic and topology adaptive MAC protocols. We formulate the problem definition in Section 2. We present the proposed ER-MAC protocol in Section 4. We show our simulation results in Section 5. Simulation results validate the performance of ER-MAC, which outperforms Z-MAC [6], a state-of-the-art hybrid MAC protocol, with higher delivery ratio and lower latency at low energy consumption. Section 6 concludes the paper. Parts of this work were presented in [9, 10].

2. Problem Definition

In this section, we describe some assumptions for the network and identify the requirements for our MAC protocol.

2.1. Assumptions

We assume a pre-deployed WSN for fire emergency that has a connected finite set of sensor nodes and one or more sinks, which are static. We also assume that there are two types of packets: high priority packets and low priority packets. The priority of a packet is determined based on its content. For example, data from temperature sensors can be tagged as high priority, while light measurements are considered as low priority. As high priority packets are more important than the low priority ones, they must be delivered first either in normal or emergency monitoring.

In "fire emergency situation", a combination of sensors, such as smoke, temperature and CO [11], ION and CO [12], can collaborate to detect the presence of fire when its sensor reading is above a specified threshold. In this hazard situation, the WSN must be able to assist fire fighters by dynamically providing important information such as the location of the fire, the

estimation of the spread of the fire, as well as evacuation routes [13] to both evacuees and the fire fighters.

We assume two different network situations: no-fire and in-fire. No-fire is the normal situation where the communication is delay-tolerant and must be energy-efficient to prolong the network lifetime. When a sensor node or a group of sensor nodes senses fire, it changes the MAC behaviour to emergency mode autonomously. The communication of in-fire nodes is not delay-tolerant and energy efficiency is not as important as achieving high delivery ratio and low latency. However, the rest of the network that is not involved in the fire monitoring must be energy-efficient.

2.2. Requirements for MAC

When designing the MAC protocol for emergency response, there are several important factors that have to be taken into account:

1. ***Traffic load*** of the network depends on the reporting frequency of the sensor nodes. It is light during normal monitoring, but increases significantly when an emergency occurs and may be unbalanced. The MAC protocol is expected to offer reliable delivery when the traffic load increases. That is, when the WSN generates more traffic, its performance does not deteriorate.
2. ***Energy efficiency*** is one of the most critical factors for WSN applications. The lower the energy consumed by each node, the longer the WSN can perform its mission. Therefore, during normal day-to-day monitoring, the network must be energy-efficient to prolong its lifetime. However, energy efficiency can be sacrificed for low latency and high delivery ratio during emergency.
3. Successful communication of the WSN not only requires a robust and reliable communication protocol to transport the important messages to the sink, but also depends on ***delivery latency***. Normal monitoring is delay-tolerant, but emergency monitoring is not, as high priority packets need timely delivery at the sink.
4. The MAC protocol has to achieve high ***delivery ratio*** in both normal and emergency situations.
5. ***Detection delay*** must be bounded, so any messages, especially the emergency ones, can reach the sink within predictable duration.

According to these requirements, the MAC protocol must be energy-efficient when the network performs normal monitoring, has low packet latency and high packet delivery ratio when the network monitors a hazard, adapts to very heavy traffic and topology changes, prioritises high priority packets and has fair packet deliveries. Since none of the existing MAC protocols reviewed in Section 3 are designed for emergency response, none of them address all of our MAC protocol requirements. Specifically, none of them try to address both packet prioritisation and fairness issues at the same time. Hence, we design ER-MAC that satisfies all of these design criteria. Packet prioritisation is necessary during emergency response to prioritise high priority packets, which are more important than the low priority ones. Fairness is important when a hazard occurs, so the sink can receive complete information from all sensor nodes in the network and monitor the spread of the hazard.

3. Related Work

Many MAC protocols have been designed for WSNs. Below, we present a selection of protocols that have relevance to our problem, i.e. traffic and topology adaptive during emergency monitoring. Based on the mechanisms to access the medium for data transmission, we follow the common classification for the MAC protocols: *contention-based*, *schedule-based* and *hybrid* that combines the features of both contention-based and schedule-based protocols. In Table 1, we compare all important issues in MAC protocol design for emergency response WSNs, which include the main objectives of the protocols, such as delivery rate or throughput, delivery latency of packet transmissions and energy efficiency, as well as the ability to adapt to traffic and topology changes, the availability of the design criteria to prioritise high priority packets and to support fairness. A MAC protocol is fair if all nodes have opportunity to access the channel for data transmissions and therefore the sink can receive complete information from all sensor nodes in the network.

A common contention-based MAC protocol is the Carrier Sense Multiple Access (CSMA) protocol. Even though contention-based protocols are popular due to their flexibility to adapt to changes in node density easily, they cannot cope well when the traffic load increases. Protocols such as S-MAC [1], T-MAC [2] and TA-MAC [2] suffer from periodic sleep of each node, while B-MAC [3], WiseMAC [14] and X-MAC [4] use preambles before data transmissions. Most contention-based protocols sacrifice fairness by letting

Table 1: Comparison of existing MAC protocols

Protocols	Main Objectives	Traffic Adaptability	Topology Adaptability	Packet Priority	Fairness
Contention-based					
S-MAC [1]	↓ energy	medium	good	no	no
T-MAC [2]	↓ energy	medium	good	no	no
B-MAC [3]	↓ energy	medium	good	no	medium
WiseMAC [14]	↓ energy	medium	good	no	no
TA-MAC [15]	↑ delivery, ↓ latency	medium	good	no	no
X-MAC [4]	↓ energy, ↓ latency	medium	good	no	medium
MaxMAC [5]	↓ energy, ↑ delivery, ↓ latency	good	good	no	no
Schedule-based					
TRAMA [16]	↓ energy	medium	good	no	yes
FLAMA [17]	↓ energy	medium	good	no	yes
VTS [18]	bounded latency	medium	good	no	yes
Hybrid					
Z-MAC [6]	↑ throughput	good	good	no	yes
PMAC [19]	↓ energy, ↑ throughput	good	medium	no	yes
Funneling-MAC [7]	↑ throughput	good	good	no	medium
Crankshaft [20]	↓ energy	medium	good	no	medium
RRMAC [21]	↑ delivery, ↓ latency	medium	good	no	yes
EB-MAC [8]	↑ delivery, ↓ latency	medium	good	no	no
BurstMAC [22]	↓ overhead, ↑ throughput	good	good	no	yes
i-MAC [23]	↓ latency	medium	good	no	medium

a node who has more data get more time to access the channel. However, B-MAC and X-MAC utilise random backoff to access the channel.

Time-Division Multiple Access (TDMA) is a schedule-based MAC protocol that controls the access to the channel by scheduling when a node should transmit, receive, or sleep to conserve energy. Schedule-based protocols support fairness by scheduling when a node can get access to the channel. Since the inability to maintain the schedule when the traffic and topology changes are major problems of this kind of protocol, TRAMA [16] and FLAMA [17] utilise CSMA periods to allow new nodes to join the network, while VTS [18] adaptively adjusts superframe length according to the number of nodes in range. To cope with heavy traffic, nodes of TRAMA and FLAMA release their unused slots, while VTS reduces the sleep interval.

Among all existing MAC protocols that we review in this paper, only MaxMAC [5] satisfies the objectives for emergency response WSNs, i.e. energy-efficient during light traffic load, has high delivery rate and low latency when the load increases. MaxMAC also has ability to adapt to traffic and topology changes. However, this protocol does not support packet prioritisation and and does not guarantee fairness. When the traffic load is light, MaxMAC behaves like WiseMAC [14] and it changes to pure CSMA when the load is heavy. Both WiseMAC and CSMA do not guarantee fairness because nodes with lots of data dominate the transmissions. Besides MaxMAC, BurstMAC [22] can be utilised for emergency response as it is designed for event-triggered applications with correlated traffic bursts. It has low over-

head and high throughput because traffic is handled using multiple radio channels. Even though BurstMAC guarantees fairness, it does not support packet prioritisation.

Judging solely from the ability to adapt to traffic and topology changes, besides MaxMAC and BurstMAC, only Z-MAC [6] and Funneling-MAC [7] have these capabilities. While Z-MAC supports fairness, Funneling-MAC only guarantees fairness in the region closer to the sink. Moreover, neither of them distinguish high and low priority packets. We identify a gap in the research literature for a MAC protocol that satisfies all of the requirements, i.e. minimises energy consumption when the traffic load is light, has high delivery rate and low latency when the traffic load increases, adapts to traffic fluctuations and topology changes, supports packet prioritisation and has fair packet deliveries in both normal and emergency situations. Our novel MAC protocol in this paper is designed to satisfy all of these criteria.

4. ER-MAC Protocol Design

The main functions of ER-MAC are to:

1. establish a data gathering tree with a sink as the root of the tree and retrieve neighbourhood connectivity (topology discovery),
2. establish nodes' schedules (TDMA slot assignment),
3. manage local time synchronisation to minimise clock drifts,
4. manage two priority queues for different priority packets,
5. respond to emergency events by changing MAC behaviour (MAC prioritisation) to cope with large volume of traffic, and
6. manage the network when the topology changes.

ER-MAC initially communicates using the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) with a random backoff mechanism to avoid collision, where each transmission follows the sequence of Request-To-Send (RTS)/Clear-To-Send (CTS)/DATA/ACK. During the startup phase, the data gathering tree and TDMA schedules for exclusive communication among nodes are created. We integrate routing functions into ER-MAC because even though it is less flexible [3], it is known to be more efficient in a network protocol design for WSNs [24]. Firstly, it can improve energy efficiency by eliminating the use of unnecessary protocol overheads at both MAC and routing layers. Secondly, it can improve resource management by sharing resources between the two layers. In the data gathering tree,

every node (except the sink) has one parent node and every non-leaf node (including the sink) has one or more children. The TDMA schedules enable each node to send its own data and forward its descendants' data to its parent in collision-free slots. Each node also has a special slot to broadcast a synchronisation message or any messages to its children. Besides contention-free slots, ER-MAC has a contention period at the end of each frame to support the addition of new nodes.

ER-MAC uses two queues for two kinds of packets: high and low priority packets. The low priority packets are transmitted only if the high priority queue is empty. Inside a queue, packets are ordered based on their *slack*. That is, the time remaining until the packet deadline expires. The deadline is assigned by the WSN application to specify the desired bound on the end-to-end latency and is initialised by a source node. When a queue is full, the packet with the shortest slack is dropped because it most likely to miss its deadline.

With the normal mode of ER-MAC, a node only wakes up to transmit and receive messages in its scheduled time slots, and spends most of its lifetime in sleep mode to conserve energy. However, when an emergency event occurs, nodes that are affected by the hazard change their MAC to emergency mode. In the emergency mode, ER-MAC allows nodes within a one-hop neighbourhood to contend for a slot if they have priority data to be sent and if the schedule does not conflict with their two-hop neighbours' schedules. In the contention, the owner of the slot has higher priority to use its own slot than the non-owner of the slot, because it can transmit a packet immediately if it has a high priority packet to send. Furthermore, during an emergency, a node that has changed its MAC to emergency mode will wake up in the beginning of each TDMA slot for possible reception of packets.

4.1. Topology Discovery

During the initial startup phase, the sink initiates the tree construction using a simple flooding mechanism. Our process is similar to the hop tree configuration of the Periodic, Event-driven and Query-based (PEQ) routing protocol [25] and the level discovery phase of the Timing-sync Protocol for Sensor Networks (TPSN) [26]. However, in our context, the goal of the topology discovery is not only to setup a routing tree, but also to find neighbours and to track changes in the tree. Topology discovery is only performed once during the initial startup phase as nodes with ER-MAC can modify their schedules locally during the network lifetime.

The sink generates a *TOPOLOGY_DISCOVERY* message, which consists of:

1. *src_ID* is the sender of the message,
2. *hop_count* stores the number of hops to reach the sink,
3. *new_parent_id* stores the new parent ID of a node, and
4. *old_parent_id* stores a node's previous parent ID.

The format of a *TOPOLOGY_DISCOVERY* message is depicted in Figure 1. This message is broadcast by a node to find its prospective children, as well as a reply to its parent and a notification to its previous parent when it wants to change parent. A node replies to its new parent, so the parent can add it to its children list. When choosing a new parent, which has shorter hop count to the sink, the node has to inform its previous parent to remove it from the parent's children list. Figure 2 illustrates a tree built for data gathering in a network of six nodes. A node has to record its parent ID because it will be used as the next hop destination in every packet transmission toward the sink. A node also needs to maintain a children list, so if it does not receive any messages from a particular child, it may know that the child is dead. We will discuss dead nodes later in Section 4.7.

Field	*type*	*src_ID*	*hop_count*	*new_parent_id*	*old_parent_id*
Field size (bytes)	1	2	2	2	2

Figure 1: *TOPOLOGY_DISCOVERY* packet format

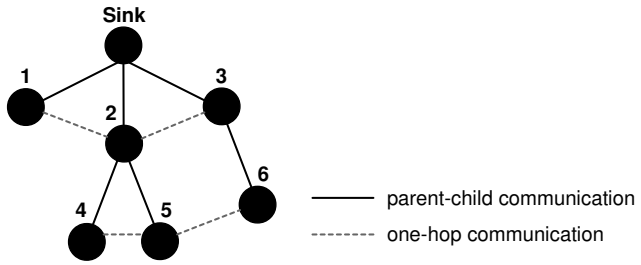

Figure 2: A data gathering tree of six nodes

The sink initialises *hop_count* as zero and leaves both *new_parent_id* and *old_parent_id* as undefined. It broadcasts the message to its neighbours within

its transmission range. In this phase, each node records the number of hop counts to the sink, its parent ID, a list of its children and its one-hop neighbour list. Communications among nodes during this phase use CSMA/CA with random-access to avoid collisions, because the TDMA schedules for exclusive communication have not been created yet.

Figure 3 shows the message exchange between a node, its parent, its child(ren) and a new parent during the topology discovery. Note that we do not show message overhearing in this figure because we want to focus the illustration on messages received and broadcast by a node. When a node receives its first *TOPOLOGY_DISCOVERY* message, it sets the sender of the message as its parent, increments the *hop_count* by one and sets it as its hop count to the sink. The node then stores its parent ID in *new_parent_id*, sets *old_parent_id* as undefined, waits for a random amount of time and re-broadcasts the message. If the node has already received a *TOPOLOGY_DISCOVERY* message before, it compares the new message's *hop_count* with its current hop count. If the new message's *hop_count* incremented by one is less than its hop count, it updates its parent ID and its hop count value. Then, it stores the new parent ID in the message's *new_parent_id*, the previous parent ID in *old_parent_id*, waits for a random amount of time and re-broadcasts the message. Otherwise, if the new message's *hop_count* incremented by one is greater or equal to its hop count, the node ignores and does not re-broadcast the message.

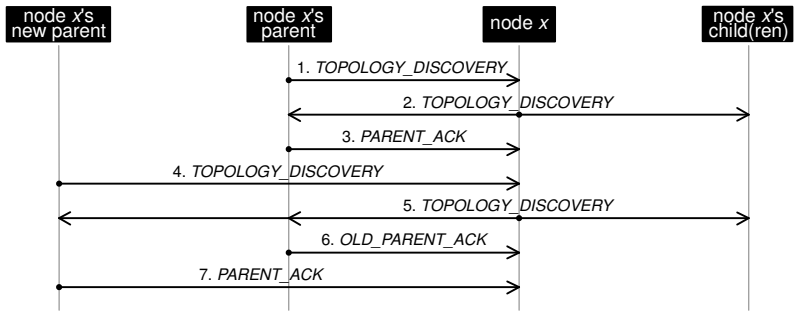

Figure 3: Message exchange in topology discovery

Upon receiving a *TOPOLOGY_DISCOVERY* message, a node also checks the message's *new_parent_id* and *old_parent_id*. If *new_parent_id* is the same as the node's ID, it adds the sender's ID to its list of children. If the node's

ID is the same as *old_parent_id*, it removes the sender's ID from its list of children.

For reliability, a parent node replies to its children with a *PARENT_ACK* message to confirm that each child has been added to its children list. If a node does not receive a *PARENT_ACK* message after broadcasting a *TOPOLOGY_DISCOVERY* message for a certain period of time (user parameter), it re-broadcasts the message. The node keeps broadcasting the *TOPOLOGY_DISCOVERY* message until it receives a *PARENT_ACK* message or exceeds the number of maximum retransmission. In another case, if a node updates its parent ID and its hop count value, it also needs a reply from its old parent after re-broadcasting the *TOPOLOGY_DISCOVERY* message. The old parent replies to the node with an *OLD_PARENT_ACK* message to inform the node that it has been removed from the children list. If the node does not receive the *OLD_PARENT_ACK* message, it will re-broadcast the *TOPOLOGY_DISCOVERY* message. The *OLD_PARENT_ACK* message helps keep the children list up to date. If the children list is not updated, the old parent may waste energy in idle listening, tries to receive some packets from the child for several data gathering cycles before deciding to remove it from the list. The node will keep broadcasting the *TOPOLOGY_DISCOVERY* message until it receives the *PARENT_ACK* and the *OLD_PARENT_ACK* messages, or exceeds the number of maximum retransmission.

During the topology discovery phase, a node may overhear transmissions from other nodes within its transmission range. The node records the senders of the messages as its one-hop neighbours in the one-hop neighbour list. This phase ends when all nodes in the network have already received the *TOPOLOGY_DISCOVERY* message. When this phase ends, each node knows the number of hops to reach the sink, its parent, the children list and the one-hop neighbour list. The pseudocode for topology discovery is given in Algorithm 1. The worst case performance of this algorithm is $O(mn^2)$, where m is the maximum retransmission and n is the number of nodes in the network.

4.2. TDMA Slot Assignment

During this phase, nodes perform slot assignment and exchange schedules, so no two nodes within a two-hop neighbourhood use the same slot. If two nodes are two hops away from each other and have the same time slot, their transmissions may collide at a node that is one hop away from both of them. At the end of this phase, each node maintains its own schedule, as well as its

Algorithm 1: Topology Discovery

```
if this.id = sink_id then
    generate and broadcast TOPOLOGY_DISCOVERY;
end
if receive TOPOLOGY_DISCOVERY then
    if this.hop_count > TOPOLOGY_DISCOVERY.hop_count + 1 then
        update this.hop_count and this.parent_id;
        generate and broadcast TOPOLOGY_DISCOVERY;
    end
    if this.id = TOPOLOGY_DISCOVERY.new_parent_id then
        add TOPOLOGY_DISCOVERY.src_id to this.children;
        generate and send PARENT_ACK to TOPOLOGY_DISCOVERY.src_id;
    else if this.id = TOPOLOGY_DISCOVERY.old_parent_id then
        remove TOPOLOGY_DISCOVERY.src_id from this.children;
        generate and send OLD_PARENT_ACK to TOPOLOGY_DISCOVERY.src_id;
    end
    add TOPOLOGY_DISCOVERY.src_id to this.one_hop_neighbour;
end
wait;
if does not receive PARENT_ACK and OLD_PARENT_ACK and not exceed
maximum_retransmission then
    re-broadcast TOPOLOGY_DISCOVERY;
end
```

one-hop and two-hop neighbours' schedules to avoid schedule conflict. Our TDMA slot assignment follows a bottom-up approach, where a leaf node (a node with no children) starts the slot assignment. Our purpose of starting the slot assignment from the leaf nodes is to have transmission schedules that can support message flow toward the sink. During the TDMA slot assignment phase, all communications that are used to schedule conflict-free slots still use CSMA/CA.

Figure 4 shows the message exchange between a node, its parent and its two-hop neighbourhood during the TDMA slot assignment. A node deems itself as a leaf node if it has no children after broadcasting *TOPOLOGY_DISCOVERY* messages for a certain period of time. It selects its own time slot to send data to its parent. A leaf node always selects the smallest available slot. It then generates a *SCHEDULE_ANNOUNCEMENT* message, appends its schedule (the ID of the slot) and broadcasts the message to its one-hop neighbours. Nodes in its one-hop neighbourhood then re-broadcast this message to the two-hop neighbours.

When a node receives a *SCHEDULE_ANNOUNCEMENT* message, it copies the schedule into its one-hop neighbours' schedules if the sender is its direct neighbour. Otherwise, the schedule is copied into the two-hop neighbours' schedules. Nodes that receive *SCHEDULE_ANNOUNCEMENT* mes-

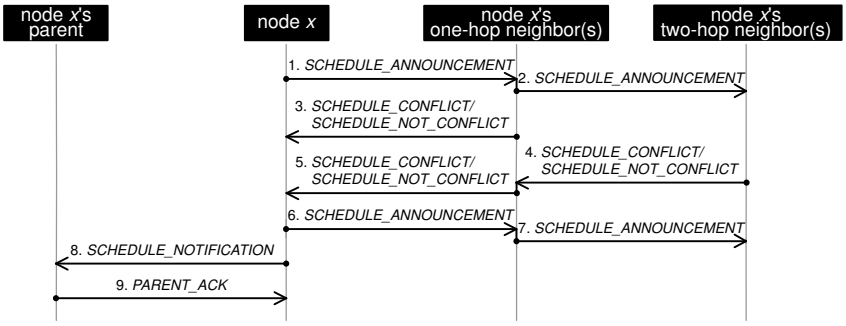

Figure 4: Message exchange in TDMA slot assignment

sages from the sender's one-hop neighbours know that they are two hops away from the sender. Every node within a two-hop neighbourhood of the message's sender checks if there is a possible conflict between its own schedule and the newly announced schedule. If it happens to be a conflict, the node generates a *SCHEDULE_CONFLICT* message, appends its schedule to the message and sends it back to the sender of the *SCHEDULE_ANNOUNCEMENT* message. When the sender of *SCHEDULE_ANNOUNCEMENT* receives the *SCHEDULE_CONFLICT* message, it updates the conflict schedule in either its one-hop neighbours' schedules or its two-hop neighbours' schedules, depends on the origin of the *SCHEDULE_CONFLICT* message. Then, it reassigns the schedule and broadcasts a new *SCHEDULE_ANNOUNCEMENT* message to its two-hop neighbourhood.

Keeping in mind that collisions on the channel exist during this random-access period, we take into account lost and duplicate messages. Because the *SCHEDULE_CONFLICT* message may be lost during transmission, we make other neighbours that receive the *SCHEDULE_ANNOUNCEMENT* message send *SCHEDULE_NOT_CONFLICT* messages to the sender of *SCHEDULE_ANNOUNCEMENT* if their schedules do not conflict. In order to reduce further collisions, the *SCHEDULE_ANNOUNCEMENT* sender saves a list of neighbours' ID whom it receives *SCHEDULE_NOT_CONFLICT* messages from and appends this list to the *SCHEDULE_ANNOUNCEMENT* message. Neighbours do not send *SCHEDULE_NOT_CONFLICT* messages if they are in the list. The sender of *SCHEDULE_ANNOUNCEMENT* is convinced that its schedule does not conflict with its two-hop neighbours' schedules if it receives no more *SCHEDULE_NOT_CONFLICT* messages from its two-hop neighbourhood after broadcasting *SCHEDULE_ANNOUNCEMENT*

messages several times. The node then sends its assigned schedule in a *SCHEDULE_NOTIFICATION* message directly to its parent. The parent acknowledges the reception of this message with *PARENT_ACK*.

Figure 5 illustrates the format of a message that is used to exchange schedules, i.e. *SCHEDULE_ANNOUNCEMENT*, *SCHEDULE_CONFLICT*, *SCHEDULE_NOT_CONFLICT* and *SCHEDULE_NOTIFICATION*. A schedule packet consists of:

1. *src_ID* is the sender of the message,
2. *dest_ID* is broadcast if used by *SCHEDULE_ANNOUNCEMENT*, the destination's ID if used by *SCHEDULE_CONFLICT* and *SCHEDULE_NOT_CONFLICT*, the parent's ID if used by *SCHEDULE_NOTIFICATION*,
3. *neighbour_level* specifies whether a node is in one-hop or two-hop neighbourhood of the sender of *SCHEDULE_ANNOUNCEMENT*,
4. *slot_list* records the schedule,
5. *highest_slot* specifies the TDMA frame length, and
6. *neighbour_list* is a list of neighbours' ID.

Field	*type*	*src_ID*	*dest_ID*	*neighbour_level*	*slot_list*	*highest_slot*	*neighbour_list*
Field size (bytes)	1	2	2	2	2 x *num_slot*	2	2 x *num_neighbour*

Figure 5: Schedule packet format

We introduce an idea of broadcast slot, so a node can send a *SYNCHRONISATION* message to synchronise its children. A non-leaf node (except the sink) waits until all of its children inform it of their schedules before assigning:

1. one unicast slot to send its own data,
2. several unicast slots to forward its descendants' data, and
3. a broadcast slot to synchronise its children.

A node assigns a slot to itself by selecting the smallest available slot which is not used within its two-hop neighbourhood. This means the same slot can be used by two nodes that are separated by more than two hops away. The node also assigns several slots that are equal to the number of descendants it has to forward its descendants' data. For each forwarding slot, the node selects

the smallest available collision-free slot. In addition, the node also selects a special broadcast slot to synchronise its children. This assigned schedule is then informed to the two-hop neighbourhood.

Nodes execute the slot assignment until *SCHEDULE_NOTIFICATION* reaches the sink. The slot assignment phase ends when the sink receives *SCHEDULE_NOTIFICATION* messages from all of its direct children and assigns a broadcast slot to synchronise them. Figure 6 illustrates assigned transmit slots in a data gathering tree of six nodes.

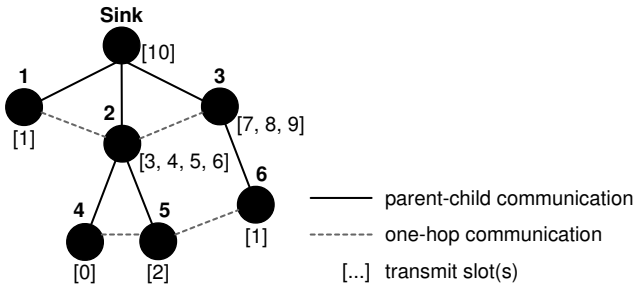

Figure 6: ER-MAC nodes' transmit schedules

The sink switches the communication mode to TDMA by sending the first *SYNCHRONISATION* message to all of its children, together with the information about the TDMA frame length. The purpose of propagating the TDMA frame length is to allow nodes in the network to keep the period of one TDMA frame length up to date. When a child receives the *SYNCHRONISATION* message, it switches its communication mode to TDMA and synchronises its children using its special broadcast slot. When all leaf nodes in the network receive a *SYNCHRONISATION* message, the whole network is switched to TDMA mode, synchronised, and each node in the network knows the exact duration of one TDMA frame. The pseudocode for TDMA slot assignment is given in Algorithm 2. The worst case performance of this algorithm is $O(\delta^2 n)$, where δ is the maximum size of a two-hop neighborhood and n is the number of nodes in the network.

4.3. Local Time Synchronisation

Time synchronisation is important in MAC protocols that adopt the schedule-based mechanism because nodes that have the same schedules for communication need to be active at the same time to transmit and receive

Algorithm 2: TDMA Slot Assignment

```
if leaf node or receive SCHEDULE_CONFLICT then
    assign schedules;
    generate and broadcast SCHEDULE_ANNOUNCEMENT;
end
if receive SCHEDULE_ANNOUNCEMENT then
    copy SCHEDULE_ANNOUNCEMENT.slot_list to this.taken_slot;
    if SCHEDULE_ANNOUNCEMENT.neighbour_level = 0 then
        re-broadcast SCHEDULE_ANNOUNCEMENT;
    end
    if schedule conflict then
        generate and send SCHEDULE_CONFLICT to
        SCHEDULE_ANNOUNCEMENT.src_id;
    else if schedule not conflict and this.id is not in
    SCHEDULE_ANNOUNCEMENT.neighbour_list then
        generate and send SCHEDULE_NOT_CONFLICT to
        SCHEDULE_ANNOUNCEMENT.src_id;
    end
end
wait;
if does not receive SCHEDULE_NOT_CONFLICT then
    generate and send SCHEDULE_NOTIFICATION to this.parent_id;
end
if receive SCHEDULE_NOTIFICATION then
    generate and send PARENT_ACK to SCHEDULE_NOTIFICATION.src_id;
end
```

messages. If the synchronisation messages are sent too often, they will incur a large amount of protocol overhead. If they are sent rarely, nodes will experience a large clock drift [27, 28].

We design ER-MAC with a local time synchronisation. Note that during the topology discovery of ER-MAC, each node discovers its parent and its children. Then, during the TDMA slot assignment, each node is assigned a special broadcast slot for synchronisation purposes. ER-MAC manages the local time synchronisation using a parent-children broadcast synchronisation, which is similar to the root-neighbours synchronisation of Flooding Time Synchronisation Protocol (FTSP) [29]. This simple mechanism is sufficient for our approach because each child only needs to have the same clock as its parent to ensure that the parent is in receive mode when it starts transmission and vice versa.

In the synchronisation slot, a parent broadcasts a *SYNCHRONISATION* message, which consists of:

1. *src_ID* is the parent's ID.
2. *current_slot* informs the current slot number to allow nodes that are not synchronised, such as new nodes, to synchronise themselves when

they overhear this message.

3. *highest_slot* is the highest number of contention-free slot, that informs the TDMA frame length to allow nodes in the network to keep the period of one TDMA frame length up to date.
4. *clock* that informs the parent's clock to help children to synchronise their clock.
5. *hop_count* is the parent's hop count to the sink. This information helps a new node to select its prospective parent by choosing a parent node with the lowest hop count to the sink.

The format of a *SYNCHRONISATION* message is shown in Figure 7.

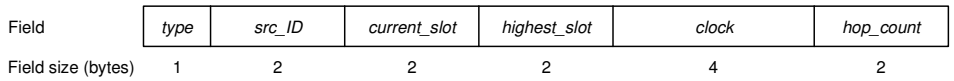

Figure 7: *SYNCHRONISATION* packet format

In ER-MAC, the local time synchronisation is performed once by each node that has child(ren) in each data gathering cycle to minimise clock drift. If a network has n nodes, there will be less than n *SYNCHRONISATION* messages sent during one data gathering cycle period because leaf nodes do not send these messages. This amount of overhead is fair and fixed regardless of the traffic rate. This scheme is more efficient than the scheme that requires a network-wide synchronisation before several contention-free slots, which is adopted by RRMAC [21]. There is also a traffic-based synchronisation, which is adopted by PMAC [19] and Z-MAC [6]. In the traffic-based synchronisation, each node sends one synchronisation message according to the traffic rate in the network. With PMAC, a node sleeps for several time frames when there is no traffic in the network. It only sends a synchronisation message when it wakes up. With Z-MAC, a node sends a synchronisation message after sending 100 data packets. Compared to the traffic-based scheme, ER-MAC has more synchronisation overhead if the traffic load is light. However, the traffic-based scheme incurs large clock drift because of infrequent synchronisation. Additionally, if the traffic load is heavy, which is expected during an emergency monitoring, ER-MAC has less overhead. In the case of synchronisation error, an ER-MAC node can turn on its radio to overhear its neighbours' *SYNCHRONISATION* messages.

4.4. Priority Queue

ER-MAC uses two queues to separate high priority from low priority packets. This multiple-queue system for sensor networks has been suggested in [30, 31, 32]. In our implementation of the priority queue, a packet is ordered based on its *slack*, i.e. the time remaining until the global packet deadline expires and is part of the packet header [33]. The format of ER-MAC's data packet is shown in Figure 8. The deadline is assigned by the WSN application to specify the desired bound on the end-to-end latency. A source node, which generates a data packet, initialises the slack with a deadline. The slack is updated at each hop by subtracting the queuing and transmission delays from it. To measure the queuing delay, a packet is timestamped when it is enqueued and dequeued. The queueing delay is the time difference between the enqueue and dequeue time. Then, to measure the transmission delay, a packet is timestamped when it is transmitted by a sender and received by a receiver. When a packet is re-transmitted, the slack is updated. The transmission delay is the time difference between the transmission time and the arrival time of a packet, given that the sender and receiver are locally synchronised.

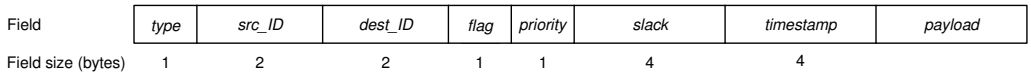

Figure 8: Data packet format

We put the packet with the shortest slack in the front of the queue. Therefore, the shorter the slack, the sooner the packet should be transmitted. The rule of getting packets out of the queue is the high priority packets are transmitted first until the high priority queue is empty. If the high priority queue is empty, the packet in the front of the low priority queue is transmitted. A packet may be enqueued in a full queue. If this situation happens, we drop a packet with the shortest slack because it is most likely to miss its deadline and we assume that a packet that misses its deadline is useless. The consequence of this technique, however, is that messages from leaf nodes are dropped more frequently than others.

We also modify the implementation of the priority queue by considering fairness over the packets' sources, so the sink can have a balance of information from all sensor nodes. When the reporting frequency increases, a node may have lots of its own data in the queue. If the node always takes

a packet from the head of the queue, it may happen that the node sends its own generated data more than its descendants' data. So, we modify our priority queue to transmit one packet from each descendant during one data gathering cycle period. We use an array, where the indexes correspond to nodes' ID, to record sources whose packets have been forwarded. We mark cell i in the array if node i's packet is dequeued. This array is reset every data gathering cycle. To dequeue a packet, we search through the queue to find a packet whose source has not been marked in the array. If such a packet exists, it will be dequeued and the source's cell is marked. If one packet from every descendant has been forwarded, we take the packet from the head of the queue. This approach, however, has search time equivalent to the length of the queue in the worst case, because we may need to search the queue to the end for each transmitted packet. The pseudocode for this technique is presented in Algorithm 3.

Algorithm 3: Fair Dequeue

```
Input: Queue, Source
Output: packet_to_send
packet_to_send ← null;
if Queue is not empty then
    foreach packet in Queue do
        if Source_{packet.source_address} = 0 then
            packet_to_send ← packet;
        end
    end
    if packet_to_send = null then
        packet_to_send ← Queue.head.packet;
    end
    Source_{packet_to_send.source_address} ← 1;
end
return packet_to_send;
```

4.5. MAC Prioritisation

The ER-MAC frame consists of contention-free slots with duration t_S each and a contention period with duration t_C as depicted in Figure 9. In each contention-free slot, except for the synchronisation slot, there are sub slots t_0, t_1, t_2 and t_3, which only appear in emergency mode for contention. Note that in the emergency mode, the period of $t_S - (t_0 + t_1 + t_2 + t_3)$ is sufficient to carry a packet and a sub slot is big enough to carry a MAC

header (a source, a destination and a flag). However, the sub slots are not used in the normal mode, where a sender occupies a slot from the beginning of the slot and sleeps after transmitting a packet or at the end of the slot. We include a contention period at the end of each frame to support addition of new nodes. When a new node joins the network after a startup phase, it can use this contention period to find its parent and exchange schedules with its neighbours. The exchange schedule process due to the addition of a new node, which will be discussed in Section 4.6, is carried out to the sink in each contention slot of a data gathering cycle.

In normal monitoring, communication between sensor nodes follows the nodes' schedules. Every node sends its own data and forwards its descendants' data to its parent in collision-free slots. A node also has a special slot to broadcast synchronisation message or any messages to its children. To further conserve energy, a sender node turns off its radio if it has no data to send and a timeout forces a receiver node back to sleep if it does not receive any packets.

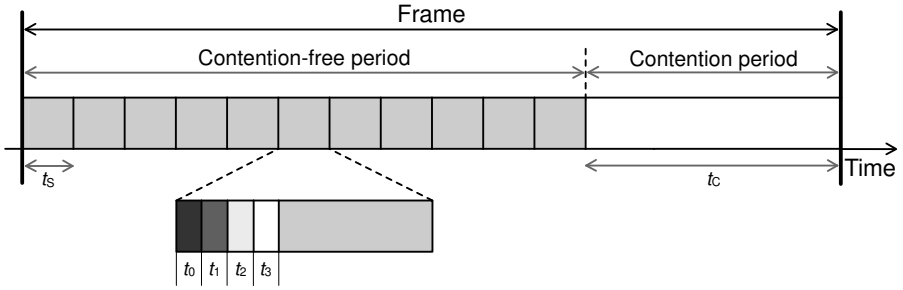

Figure 9: ER-MAC's frame structure

When fire is detected by some nodes' sensors, they change their MAC to emergency mode and set the emergency flag in their high and low priority packets. Note that only their parents can receive the packets with the emergency flag because they are scheduled to wake up. To inform other neighbours of the emergency event, nodes that detect fire also broadcast *FIRE* messages to their one hop neighbours using their contention slots. The one-hop neighbours that receive *FIRE* messages change their MAC to emergency mode so they can give up their transmit slots when needed by the nodes sensing the fire. The ancestors of the nodes caught in fire change their MAC to emergency mode when they receive data packets with the emergency flag. These ancestors inform their one-hop neighbours to switch to emergency mode by

broadcasting *FIRE* messages using their contention slots. The ancestors' one-hop neighbours change their MAC so they can give up their transmit slots when needed by the nodes that are relaying emergency traffic. During the emergency situation, the whole network's MAC protocol is not switched in an instant, but hop by hop depending on the spread of the hazard. Nodes that do not participate in the emergency monitoring remain in the normal mode of ER-MAC.

Nodes change the behaviour of their MAC to emergency mode to achieve high delivery ratio and low latency by allowing contention in TDMA slots with the following rules:

1. An owner of a slot wakes up in the beginning of its own transmit slot. If it has a high priority packet to send, it transmits the packet immediately. If the owner has no high priority packet to send, it allows its one-hop neighbours with high priority packets to contend for the slot.
2. All non-owners of the slot wake up in the beginning of every slot to listen to the channel for possible contention or reception of packets. If a non-owner with a high priority packet senses no activities on the channel during t_0, it contends for the slot during t_1 by sending a *SLOT_REQUEST* message to the owner of the slot. The owner of the slot replies the request by sending *SLOT_ACKNOWLEDGEMENT* to the requester.
3. The owner of the slot with low priority packets can only use its own slot if during t_0+t_1 it does not receive any *SLOT_REQUEST* messages from its neighbours.
4. A non-owner with low priority packet can contend for the slot if during $t_0 + t_1 + t_2$ it senses no activities on the channel. Then, it contends for the slot during t_3 by sending a *SLOT_REQUEST* message to the owner of the slot. The owner of the slot replies the request by sending *SLOT_ACKNOWLEDGEMENT* to the requester. Therefore, a node with low priority packets has a chance in every slot to contend for sending a packet.

A node that has switched to emergency mode may have neighbours that still operate in normal mode. Hence, it has to sense the channel before contending for its neighbours' transmit slot to avoid collision. If it does not sense any activities, it may contend for the slot, else it knows that the neighbour which is not in emergency mode is using its transmit slot. Moreover, to

prevent a node sending an emergency packet to a sleeping parent, the first emergency packet is sent in a scheduled transmission slot. This will allow the ancestors of the node to switch their MAC protocol to the emergency mode when they receive the emergency packet. The delivery latency of the first emergency packet is however the same as in normal situation, but when nodes that involve in the emergency monitoring have switched their MAC protocol to the emergency mode, the latency of high priority packets is reduced.

A false alarm may happen in the network, where a node mistakenly thinks that it detects fire. If it happens, this node will inform its one-hop neighbours by sending a *FALSE_ALARM* message to change their MAC behaviour back to the normal mode. The ancestors of the node on the route to the sink that have already switched to emergency mode will change their MAC back to the normal mode if they do not receive any emergency packets after n data gathering cycles. They will also inform their one-hop neighbours regarding the false alarm.

4.6. New Nodes

The length of ER-MAC frame depends on the number of nodes in the routing tree. When a new node is added, the number of TDMA slots increases. ER-MAC supports addition of new nodes by utilising the contention slot at the end of each TDMA frame. When a new node is deployed, it has to listen to its neighbours' *SYNCHRONISATION* and data messages for at least one data gathering cycle. The *SYNCHRONISATION* message has several pieces of information that are useful to support addition of new nodes. The information about sender's ID and sender's hop count to the sink help the new node to select its parent. The new node will select a parent that has the lowest hop count to the sink. The *SYNCHRONISATION* message also reports the current slot number, the highest slot number and the clock of the prospective parent to help the new node synchronises its clock and wait until the next contention slot to perform schedule exchange.

The slot assignment for a new node is similar to the slot assignment during the initial setup phase as described in Section 4.2, except that the new node takes the highest slot number incremented by one to be its slot number and the schedule exchange is performed in a contention slot. The new node generates a *SCHEDULE_ANNOUNCEMENT* message, appends its schedule and broadcasts the message to its one-hop neighbours. Nodes in its one-hop neighbourhood then re-broadcast this message to the two-hop

neighbours. The new node has to wait until it receives no more *SCHEDULE_NOT_CONFLICT* messages from its two-hop neighbourhood after broadcasting the *SCHEDULE_ANNOUNCEMENT*. The new node then sends its assigned schedule in a *SCHEDULE_NOTIFICATION* message directly to its new parent. When the parent receives *SCHEDULE_NOTIFICATION* message from the new node, it acknowledges the new node as its child and adds the new node's transmit schedule to its receive schedule.

The parent then has to allocate one transmit slot to forward the new node's data and one slot to synchronise it if the parent has no children before. The transmit slot and the synchronisation slot are the new node's slot number incremented by one and two, respectively. The parent then performs schedule exchange during the next contention slot. The process of allocating new transmit slots because of the addition of the new node is carried out along the new node's routers toward the sink in each contention slot of a data gathering cycle. It takes approximately $(l+1) \times t$ seconds until the slot assignment reaches the sink after the new node is deployed, where l is the new node's hop count to the sink and t is one data gathering cycle period. Note that the one additional data gathering cycle is used by the new node to overhear its neighbours' *SYNCHRONISATION* and data messages prior to assigning its own schedule. The process of allocating new transmit slots because of the addition of a new node is illustrated in Figure 10.

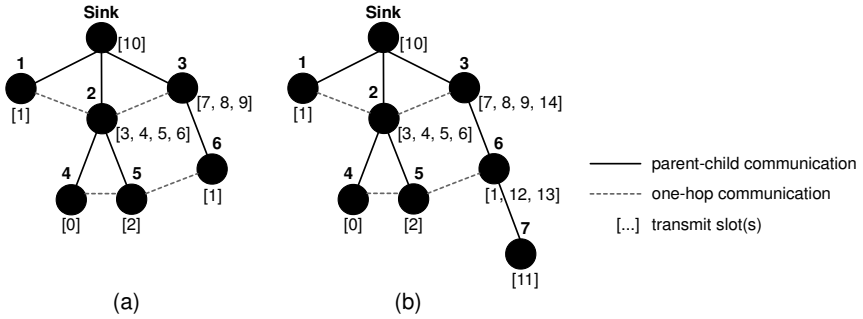

Figure 10: Addition of a new node, where (a) is the original network and (b) shows the network after node 7 is added

The addition of new slots lengthens the TDMA frame. Therefore, these changes must be informed to all nodes and they have to adjust their TDMA frame length simultaneously in the beginning of a data gathering cycle. The new node and the routers also start using their new allocated slots in the same

data gathering cycle. This will prevent schedule clash where some part of the network has already changed its TDMA frame length while some other still use the old TDMA frame length. To apply the changes, a count down timer, set to be $l_{\max} \times t$ seconds, is piggybacked in the sink's *SYNCHRONISATION* message and is propagated to the whole network when a node synchronises its children. $l_{\max}$ is the highest hop count of the network. As the timer expires, all nodes simultaneously use the new schedules. The process of disseminating the new frame length proceeds until all nodes change their TDMA frame length and takes at most $l_{\max}$ data gathering cycle periods. Hence, the total time needed for a new node to operate in TDMA mode after it is deployed is $(l_{\max} + l + 1) \times t$ seconds.

The frame length inconsistencies, where some nodes use old frame length and some nodes use new frame length, are unlikely to happen because the synchronisation slots are collision-free. Moreover, it takes $l_{\max}$ data gathering cycle periods to disseminate the new frame length information. If a node does not receive a *SYNCHRONISATION* message due to temporary noisy links, it will receive it in the next period. If the links are permanently noisy, the node will find a new parent.

4.7. Dead Nodes

A node is dead if it runs out of battery or is destroyed by fire. We can also assume a node is dead if it cannot communicate with its parent or its children due to noisy links or obstacles. If a parent does not receive any data during all scheduled receive slots of a child after n consecutive data gathering cycles (user parameter), where n is usually greater than one to deal with temporary link failure, it assumes that the child is dead. The parent then removes the child from its children list. It also removes m scheduled receive slots that are associated with that child to prevent idle listening. If the child is the only child of that parent, it also removes the synchronisation's broadcast slot. Moreover, the parent also removes m transmit slots to forward that child and its descendants' data. The parent is then responsible to inform all the routers toward the sink to remove m receive slots associated with the removal of one of its children and m transmit slots from their schedules. This information is piggybacked on the data packet sent in the immediate transmit slot. All of the unused slots are then informed within two-hop neighbourhood in the contention slot.

When a node does not receive *SYNCHRONISATION* messages after n data gathering cycles from its parent, it may assume that its parent is dead.

The orphan node then finds a new parent by following the same procedure as the new node deployment. In a contention slot, the orphan node will send its transmit slots' schedule in a *SCHEDULE_NOTIFICATION* message directly to its new parent, so the descendants of the orphan node do not need to rebuild their schedules. When the parent receives *SCHEDULE_NOTIFICATION* from the orphan node, it acknowledges the orphan node as its child and adds the orphan node's transmit schedule to its receive schedule. The parent then assigns new transmit slots to forward its new child and new descendants' data. This schedule assignment is the same as the new node's assignment, except that there may be more than one slot that needs to be allocated because the orphan node may have children and descendants. To reuse some released slots, this schedule assignment will firstly search for the smallest available slot, which does not conflict with the schedule of the node's two-hop neighbours.

4.8. Protocol Overhead

ER-MAC incurs higher protocol overhead at the beginning, i.e. during topology discovery and TDMA slot assignment phases. During this initial startup, ER-MAC communicates using CSMA/CA, where there are RTS/CTS/ACK in each transmission. After the initial startup, ER-MAC's protocol overhead during normal monitoring is only caused by *SYNCHRONISATION* messages, which are sent once each data gathering cycle by every node with child(ren). This amount of overhead is fixed regardless of the traffic rate and bounded by the number of nodes.

When the network monitors emergency, besides the *SYNCHRONISATION* messages, *FIRE* or *FALSE_ALARM* messages, *SLOT_REQUEST* and *SLOT_ACKNOWLEDGEMENT* messages also contribute to the amount of protocol overhead. These four types of messages are generated only by nodes involved in emergency monitoring. Therefore, the amount of the overhead depends on those nodes. While *FIRE* or *FALSE_ALARM* messages are sent once every data gathering cycle in the contention slot, *SLOT_REQUEST* and *SLOT_ACKNOWLEDGEMENT* messages might be sent by several nodes in every TDMA slot for possible contention.

4.9. Network Lifetime

In this paper, we define the lifetime of the network T as the time until the first node fails, i.e.

$$T = \min_{v \in V} T(v) \tag{1}$$

The lifetime of a sensor node $T(v)$ depends on how much energy is available for its use $E_{\text{initial}}(v)$ and how much energy it consumes over time $E_{\text{usage}}(v)$.

$$T(v) = \frac{E_{\text{initial}}(v)}{E_{\text{usage}}(v)} \tag{2}$$

The predominant amount of energy consumed by a sensor node is for sensing, communication, and data processing activities. However, since the communication cost is the most dominant factor in a sensor's energy consumption, we estimate the amount of energy used by a node v as the amount of energy used to transmit $E_{\text{tx}}(v)$, receive $E_{\text{rx}}(v)$ and for other causes $E_{\text{other}}(v)$, such as sensors and LEDs.

$$E_{\text{usage}}(v) = E_{\text{tx}}(v) + E_{\text{rx}}(v) + E_{\text{other}}(v) \tag{3}$$

The amount of energy consumption of a node to transmit is proportional to the number of transmit slots in one data gathering cycle. A node has several transmit slots to forward its descendants' data, one slot to forward its own data and one slot to broadcast synchronisation message to its children. Energy consumption of a node v to transmit is formulated as

$$E_{\text{tx}}(v) = num\_cycle \times (num\_descendants(v) + 2) \times P_{\text{tx}} \times t_{\text{tx}} \tag{4}$$

where *num_cycle* is the number of data gathering cycles, $num\_descendants(v)$ is the number of v's descendants, P_{tx} is the transmit power and t_{tx} is the transmit time.

The amount of energy consumption of a node to receive is proportional to the number of receive slots in one data gathering cycle. A node has to wake up to receive data from its descendants and one synchronisation message from its parent. Energy consumption of a node v to receive is formulated as

$$E_{\text{rx}}(v) = num\_cycle \times (num\_descendants(v) + 1) \times P_{\text{rx}} \times t_{\text{rx}} \tag{5}$$

where P_{rx} is the receive power and t_{rx} is the receive time.

5. Evaluation of ER-MAC

By these experiments, we want to show that ER-MAC delivers low latency for high priority packets especially during emergency monitoring, it has fair packet delivery and nodes in non-emergency mode behave in an energy-efficient manner. In the simulation, we use the following metrics to measure the performance of ER-MAC:

1. ***Average energy consumption per node*** is presented to compare the energy efficiency of communication protocols. The average energy consumption per node is calculated as the total energy consumed by the entire network during the simulation period for listening, transmitting, receiving, switching from sleep to idle mode and vice versa, averaged over the total number of nodes in the network. The unit of energy is the Joule. We want to show that ER-MAC is energy-efficient in normal situations.
2. ***Packet delivery ratio*** is the total number of packets received at the sink divided by the total number of packets generated by the source nodes during the lifetime of an experiment. Delivery ratio takes a value in the interval [0, 1]. We want to show that ER-MAC has high packet delivery ratio, especially for high priority packets, for both normal and emergency situations.
3. ***Average per packet latency*** measures the total time needed for each packet to reach the sink since it was sent by the source node, averaged over the total number of packets received at the sink. The unit of latency is the second. We want to show that the average per packet latency, especially for high priority packets, is reduced during an emergency situation.
4. ***Completeness*** measures the percentage of transmitted packets received by the sink. We want to show that the sink receives a fair distribution of packets from across the network, regardless of distance from the sink, and so we measure the completeness averaged over all nodes at a given hop count from the sink. In addition, we distinguish between high and low priority packets.

We implemented ER-MAC in ns-2 [34]. Our simulation results are based on the mean value of five different network deployments that are simulated five times each using random seeds, enough to achieve a 95% confidence in the standard error interval. The network consists of 100 nodes deployed within randomly perturbed grids. This is an approximation of manual deployments of sensor nodes, for example in a building layout. In the random perturbed grids, each node is placed in one unit grid square of 8 m $\times$ 8 m and the coordinates are slightly perturbed. This grid size is chosen in relation to the use of 10-metre transmission range, which is realistic for 0 dBm transmission power in an indoor environment [35]. The location of the sink was fixed at the top-left corner of the network. We use a simple wireless channel using

Table 2: Simulation parameters in ns-2 based on Tmote sky hardware [36]

Simulation parameters	**Default value**
Transmit power	52.2 mW
Receive power	59.1 mW
Idle power	59.1 mW
Sleep power	3 μW
Transition power	59.1 mW
Transition time	580 μs
Node initial energy	20,000 J (2 $\times$ AA batteries)
ER-MAC TDMA slot size	50 ms
ER-MAC TDMA sub-slot size	5 ms
Z-MAC TDMA slot size	50 ms
Z-MAC owner contention window size (T_{o})	8
Z-MAC non-owner contention window size (T_{no})	32

the two-ray ground radio propagation model. We also randomly select up to n links and for each drop up to m packets, where m is large enough to model unreliable links. Moreover, for simplicity, we assume the links are symmetric. Our simulation parameters were based on Tmote sky hardware [36]. Table 2 presents our simulation parameters.

5.1. Protocol Comparison

We compared the performance of ER-MAC with Z-MAC, because this protocol has several similar characteristics with ours, such as hybrid designs and allowing contention in TDMA slots when the traffic load increases. We followed the Z-MAC ns-2 installation manual detailed in [37] and configured Z-MAC according to the default settings in [6]. Z-MAC's configuration is shown in the simulation parameters's table (Table 2). In addition, we use the same 10-metre transmission range as in ER-MAC's simulations. In each experiment, we simulated data gathering for 300 seconds, where every node except the sink is a source node that generates packets with fixed intervals.

In the simulations, we compared the performance of ER-MAC with Z-MAC in terms of average energy consumption per node, packet delivery ratio, average per packet latency, and completeness of packets received at the sink. For ER-MAC simulations, we considered two network scenarios, i.e. no-fire and in-fire situations. In the no-fire situation, communication among nodes follows their TDMA schedules. However, in the in-fire situation, ER-MAC allows contention for the TDMA slots within a one-hop neighbourhood if

the owner of the slot has no data to send. To simulate the in-fire situation, we assume all nodes operate in emergency mode from the beginning of the simulation. For Z-MAC simulations, we forced Z-MAC to operate in either Low Contention Level (LCL) or High Contention Level (HCL) to model our no-fire and in-fire situations, respectively. Note that in LCL, any node can compete to transmit in any slots, but in HCL only the owner of the current slot and their one-hop neighbours are allowed to compete for the slot. Our simulation results show that ER-MAC outperforms Z-MAC especially when the traffic load increases.

Figure 11 shows the average energy consumption per node during the simulations. ER-MAC nodes in both no-fire and in-fire situations consume less energy than Z-MAC nodes that operate in LCL and HCL modes. This is because in ER-MAC, the owner of the slot does not need to contend to access the channel if it has data to send. However, in Z-MAC, although the owner of the slot has priority to access the medium, it has to contend for the medium before sending its own data. The figure also shows that during the in-fire situation, ER-MAC nodes spend more energy than the no-fire situation, because they wake up in every slot for possible contention. The energy consumption of ER-MAC nodes during the in-fire situation is high when the traffic load is low (less than 0.1 packets/node/sec) because more nodes do not use their own transmit slots to send their data, but contend for their one-hop neighbours' transmit slots if the neighbours have no data to send. In other words, during the in-fire situation, the lighter the load, the more the possibilities for contention in the network.

We also extended our simulations by increasing the traffic load up to 1 packet/node/sec, but saw no further variation in energy consumption. In our simulation, the network reaches its peak load at around 0.2 packets/node/sec. Hence, the energy consumption of nodes above the peak load is stable as the nodes can only communicate using their own scheduled time slots even though they have more data to send in the queues. The possibility of contention above the peak load is also minimal because nodes always have data to send in their own slots.

To compare the delivery ratio of high and low priority packets, we force source nodes to generate the two kinds of packets at the same time. Figure 12 shows that ER-MAC's high priority packets achieve better delivery ratio than Z-MAC's packets and ER-MAC's low priority packets. In the figure, the lines for Z-MAC's low priority packets are hidden below the high priority. Z-MAC delivers the same delivery ratio for the two types of packets because it does

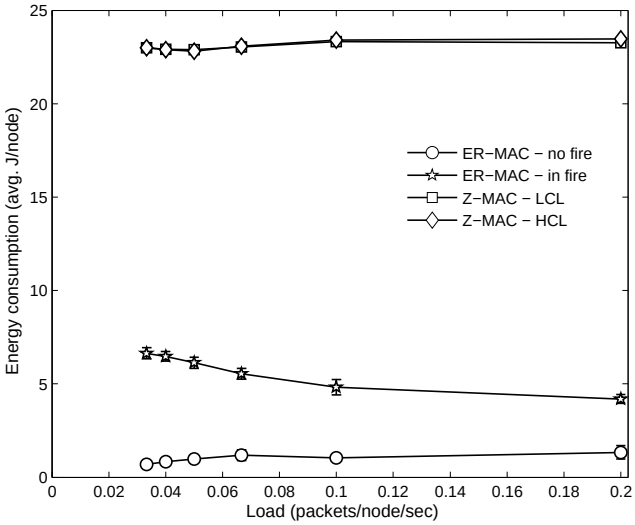

Figure 11: Energy consumption of ER-MAC versus Z-MAC

not prioritise the high priority ones. When the traffic is very light, Z-MAC that operates in HCL mode achieves higher delivery ratio than ER-MAC because of data retransmissions when the packets are lost and the senders do not receive acknowledgements. On the other hand, ER-MAC does not acknowledge every data packets and so it does not retransmit lost data. Even though the delivery ratios of ER-MAC's high priority packets decrease when the traffic load increases, its delivery ratio in the in-fire situation is slightly higher than in the no-fire situation. This phenomenon is caused by contention in TDMA slots to prioritise the propagation of high priority packets during the emergency. However, the delivery ratio of ER-MAC's high priority packets does not change much from no-fire to in-fire because when nodes generate more traffic, the chance for contention is minimal.

Figure 13 shows the average per packet latency of our simulations. ER-MAC's high priority packets generally have lower latency compared to Z-MAC's high priority packets. This is because ER-MAC maintains two priority queues that separate high priority packets from low priority ones and the high priority packets are always transmitted first until the queue is empty. On the other hand, Z-MAC only uses one queue and sends the high and low priority packets one after another, and so the latency of Z-MAC's high and low priority packets is almost identical. Moreover, ER-MAC prioritises high

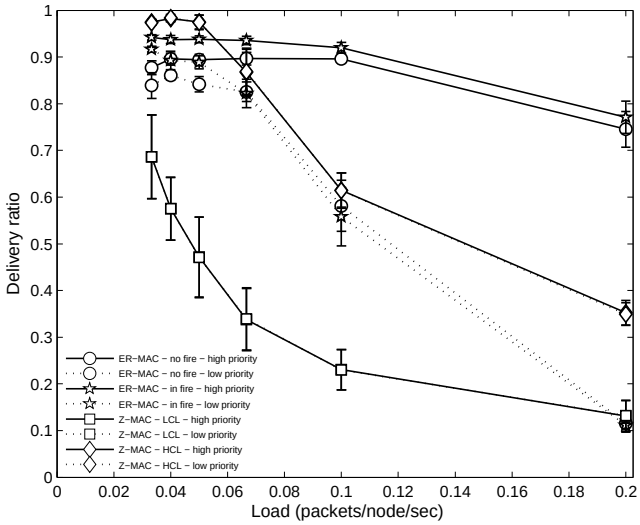

Figure 12: Delivery ratio of ER-MAC versus Z-MAC

priority packets and so the latency of low priority packets is high. During the in-fire situation, ER-MAC's high priority packets' latency is reduced because nodes can propagate data quickly by contending for some unused slots.

When we increase the traffic load up to 1 packet/node/sec, the latency of ER-MAC's high priority packets rises. On the other hand, the latency of Z-MAC's packets and ER-MAC's low priority packets drops as we observe in the simulations that fewer packets are received at the sink and most of them are from nodes near it.

As explained in Section 4.4, we implement priority queues by considering fairness over the packets' sources. The reason behind this modification is we want the sink to have a balance of information from all sensor nodes in the network. Figure 14 shows the completeness of the packets received at the sink when the network reaches its peak load, i.e. 0.2 packets/node/sec. We measure the completeness as the percentage of packets received plotted against hop count. The graph shows that the completeness of ER-MAC's high priority packets for both no-fire and in-fire situations are higher than Z-MAC's packets and ER-MAC's low priority packets. This happens because of packet prioritisation and priority queue modification in ER-MAC to transmit one packet from each node during one data gathering cycle period.

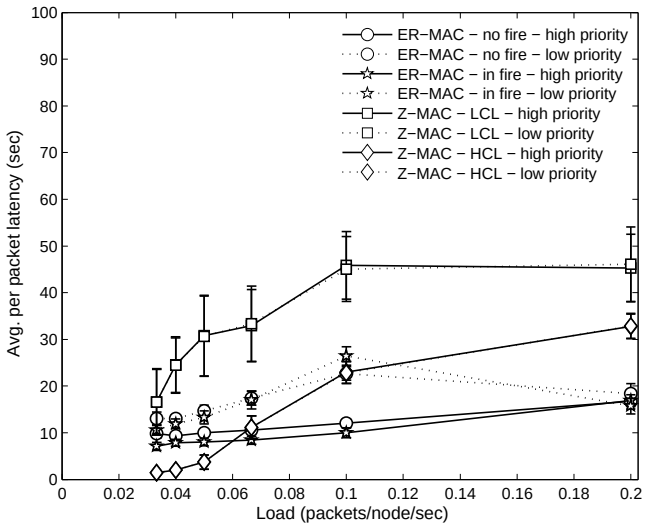

Figure 13: Latency of ER-MAC versus Z-MAC

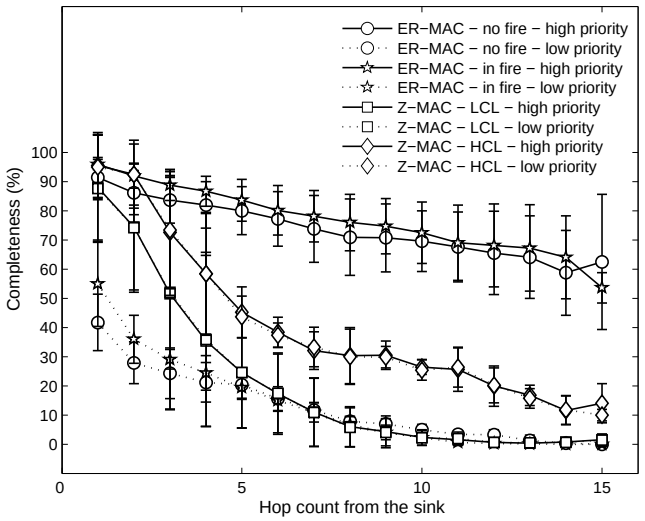

Figure 14: Completeness of ER-MAC versus Z-MAC

5.2. Behaviour When a Cluster of Nodes Detects Fire

We consider the situation when some nodes in a network detect fire. Nodes that detect fire become in-fire nodes. They change their MAC behaviour to emergency mode, set the emergency flag in each of their high and low priority packets and broadcast emergency messages to their one-hop neighbours during contention slots. The one-hop neighbours also switch to emergency mode but do not set the emergency flag in their data packets. When the ancestors of the in-fire nodes receive data packets with the emergency flag, they change their MAC to emergency mode and broadcast emergency messages to their one-hop neighbours to change their MAC. Neither the ancestors nor their one-hop neighbours set the emergency flag in any of their packets. This situation is illustrated in Figure 15.

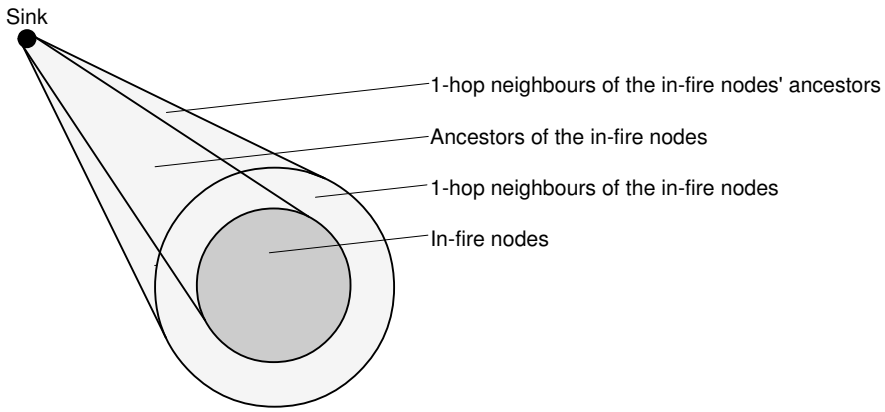

Figure 15: A cluster of nodes detects fire

We evaluate the performance of ER-MAC against Z-MAC when a cluster of nodes detects fire. For each simulation, we run a 500-second data gathering, where all nodes are the sources of high and low priority packets. They generate a constant 0.1 packets/node/sec traffic rate. 100 seconds after the simulation starts, a random location in the network is on fire. We choose five nodes, which are the closest nodes to the fire location, as in-fire nodes. The in-fire nodes double the traffic generation rate to 0.2 packets/node/sec and halve the packet deadline.

Figure 16 shows the average energy consumption per node during the 500-second simulations. The results reported at the 100^{th} second is when the network is not on fire. As the fire starts at the 100^{th} second, we start to

plot the emergency monitoring results from the 200^{th} second. The simulation results show that ER-MAC is energy-efficient during this emergency monitoring as nodes consume less than one fifth of Z-MAC's energy consumption. The figure also shows that with ER-MAC, nodes that participate in the emergency monitoring, i.e. the in-fire nodes, the one-hop neighbours of the in-fire nodes, the ancestors of the in-fire nodes and the ancestors' one-hop neighbours, dominate the energy consumption of the network. Conversely, the rest of the network, which operates in the normal mode of ER-MAC, is very energy-efficient. Z-MAC does not distinguish between nodes that participate in the emergency monitoring and the normal monitoring. It switches from LCL to HCL mode if it detects heavy traffic loads. In addition, nodes that operate in the HCL and LCL modes of Z-MAC have been shown to consume almost the same amount of energy in Figure 11.

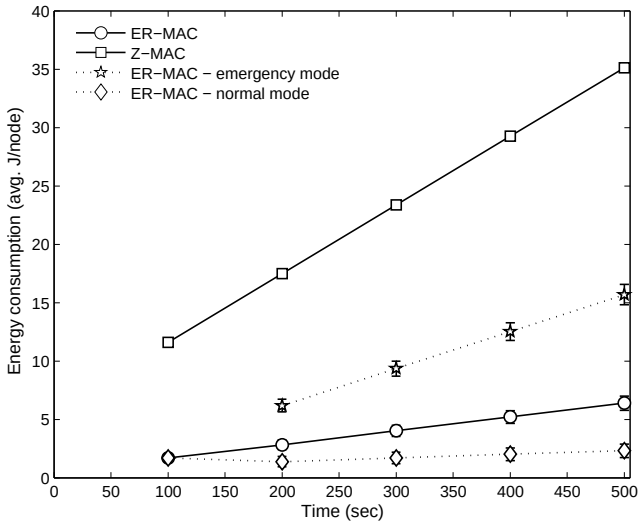

Figure 16: Energy consumption of ER-MAC versus Z-MAC when a cluster of nodes detects fire

Figure 17 presents the delivery ratio of high and low priority packets with and without an emergency flag. Recall that only in-fire nodes generate packets with the emergency flag and their reporting frequency is twice as high as the normal data. Because of the ability to prioritise packets, ER-MAC achieves higher delivery ratio for emergency and normal high priority packets compared to Z-MAC, even though it sacrifices the low priority ones.

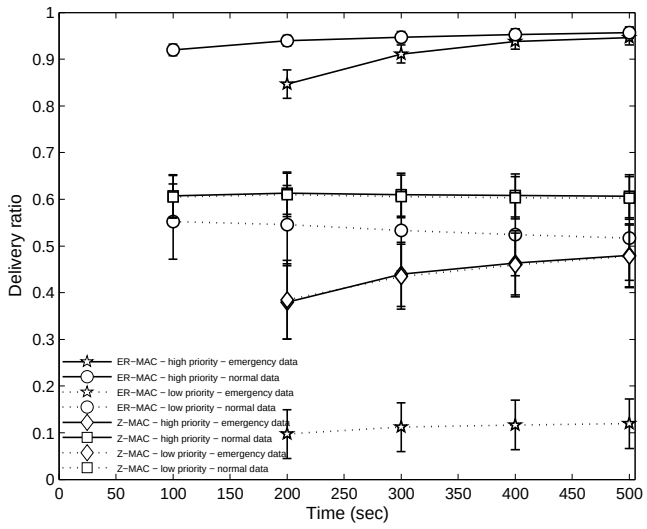

Figure 17: Delivery ratio of ER-MAC versus Z-MAC when a cluster of nodes detects fire

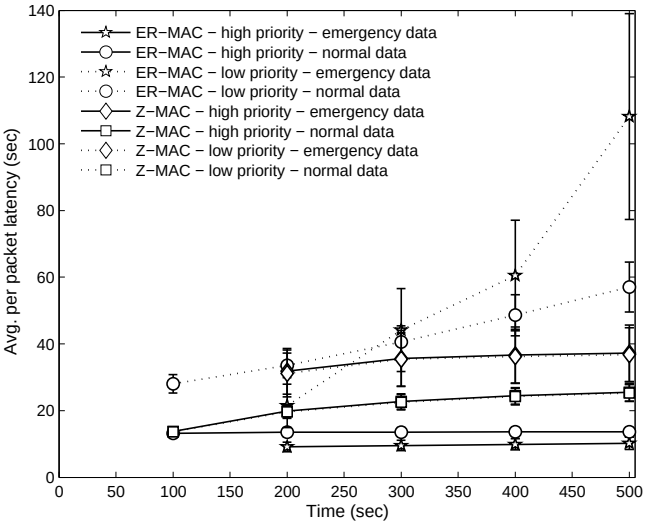

Figure 18: Latency of ER-MAC versus Z-MAC when a cluster of nodes detects fire

ER-MAC also delivers the emergency high priority packets with the lowest latency as shown in Figure 18. This happens because emergency packets have shorter deadline than normal packets and so they are placed in the front of the queue. In our priority queue modification, the emergency packets are given the priority to be sent after one packet from each descendant has been sent.

5.3. Behaviour Under Variable Traffic Load

In this simulation, we vary the traffic load during 500-second simulations. The traffic changes every 100 seconds. It jumps from 0.1 to 0.4 packets/node/sec, then drops to 0.1 packets/node/sec, and so forth. We vary the load in order to illustrate the changes in network conditions from no-fire to in-fire, then from in-fire to no-fire, and so on. When a node generates more traffic, it changes the MAC behaviour from the normal mode to the emergency mode. When it generates less traffic, it changes back to the normal mode. Figure 19, 20 and 21 show the comparison of ER-MAC against Z-MAC when the traffic changes over time in terms of average energy consumption per node, packet delivery ratio and average per packet latency, respectively.

Overall, ER-MAC outperforms Z-MAC because it is more energy-efficient and its high priority packets have better delivery ratio and latency compared to Z-MAC's. The zigzag behaviour in the delivery ratio plot is caused by changes in the traffic rate every 100 seconds. When sensor nodes increase the rate from 0.1 to 0.4 packets/node/sec, more data packets are generated and since the MAC behaviour is changed to the emergency mode, more protocol overhead incurred for contention in TDMA slots. Therefore, the delivery ratio during this period drops. In Figure 20 and 21, the delivery ratio and latency of Z-MAC's high and low priority packets overlap because Z-MAC only uses one queue and sends the high and low priority packets one after another.

5.4. Behaviour When Topology Changes

We want to show that ER-MAC is topology adaptive by simulating networks while sensor nodes are failing. In the simulation, we increase the number of dead nodes from one to five and calculate the average energy consumption and time needed to reconfigure the network. The energy consumption to reconfigure the network is the amount of energy spent by orphan nodes to find their new parents and to announce new schedules in contention slots. The network reconnectivity latency is calculated from the time a node

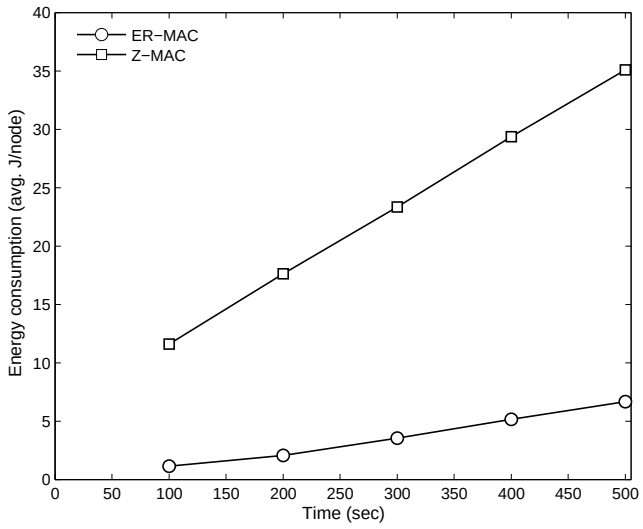

Figure 19: Energy consumption of ER-MAC versus Z-MAC under variable traffic load

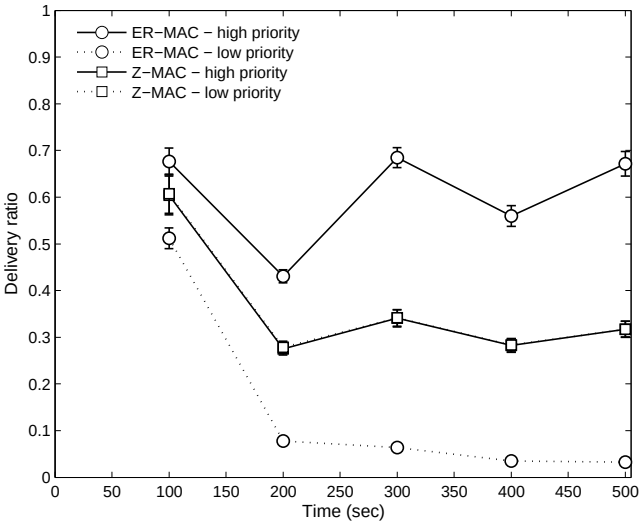

Figure 20: Delivery ratio of ER-MAC versus Z-MAC under variable traffic load

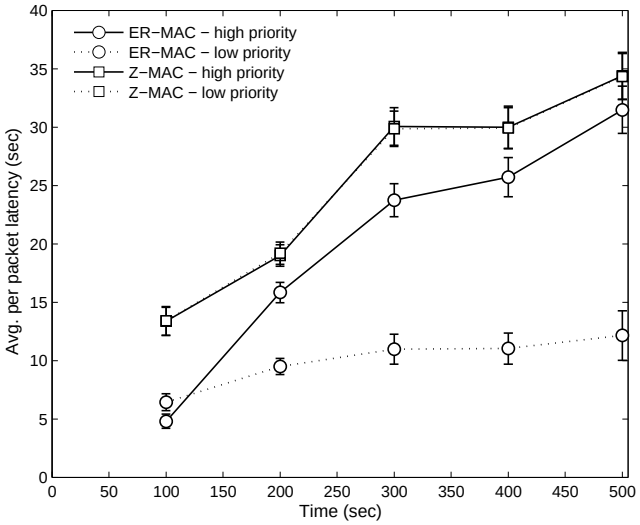

Figure 21: Latency of ER-MAC versus Z-MAC under variable traffic load

knows that its parent is dead until it uses its new TDMA schedules. In this simulation, a node is considered dead if after two data gathering cycles, its parent and children do not receive any packets from it. These simulation results are depicted in Figure 22. The amount of energy spent by a node to find a new parent is very small, i.e. less than 0.000125% of its initial energy. The reconnectivity latency slightly increases when more nodes die because the path length of an orphan node to the sink may be lengthened when it finds a new parent.

We also simulate the situation where several nodes die simultaneously. The simulation results are depicted in Figure 23, where we increase the number of dead nodes from 5 to 20. The energy consumption and latency to reconfigure the network decrease when the number of dead nodes goes over 15 because the network gets partitioned as the number of failed node increases. Hence, we only measure the energy expenditure and time to reconfigure the network from the remaining nodes that are still connected to the sink.

5.5. Protocol Overhead

In Figure 24, we compare the percentage of ER-MAC's overhead traffic to data traffic in three network situations, i.e. no-fire, in-fire, and when a cluster of five nodes detects fire as described in Section 5.2. In these three scenarios,

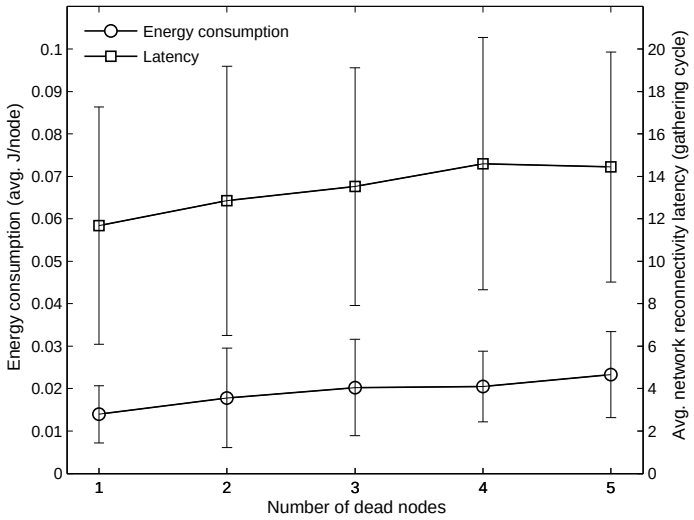

Figure 22: Energy consumption and latency of ER-MAC for network reconnectivity when some nodes die gradually

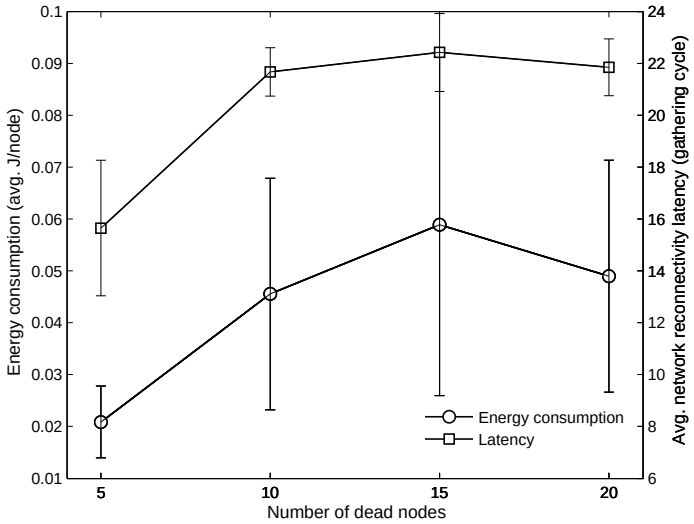

Figure 23: Energy consumption and latency of ER-MAC for network reconnectivity when some nodes die simultaneously

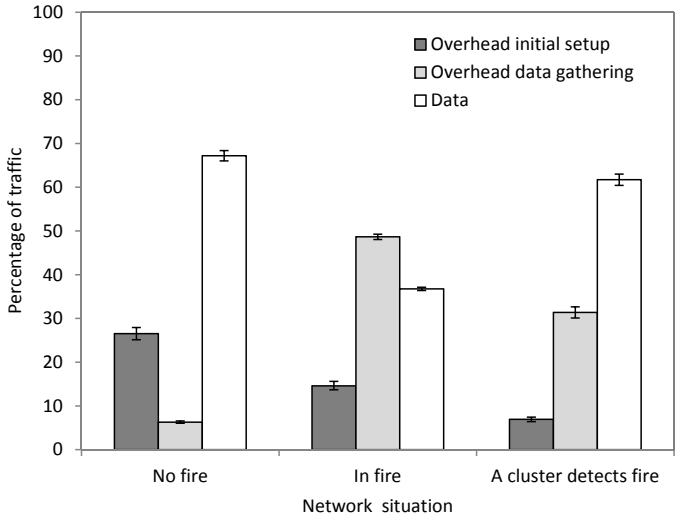

Figure 24: ER-MAC's protocol overhead traffic percentage

the no-fire nodes and in-fire nodes generate a constant 0.1 packets/node/sec and 0.2 packets/node/sec traffic rate, respectively.

ER-MAC's overhead during the initial setup includes the messages exchanged for topology discovery and TDMA slot assignment. During the data gathering period, ER-MAC's protocol overhead for no-fire nodes is only caused by *SYNCHRONISATION* messages, which are sent once each data gathering cycle by every node with child(ren). This amount of overhead is fixed regardless of the traffic rate and bounded by the number of nodes. Besides *SYNCHRONISATION* messages, the amount of overhead for in-fire nodes is caused by *FIRE* or *FALSE_ALARM*, *SLOT_REQUEST*, and *SLOT_ACKNOWLEDGEMENT*. Since the other four types of messages are generated only by in-fire nodes, the amount of the overhead depends on those nodes. *FIRE* or *FALSE_ALARM* messages are sent once every data gathering cycle in the contention slot. However, *SLOT_REQUEST* and *SLOT_ACKNOWLEDGEMENT* messages might be sent by several nodes in every TDMA slot for possible contention.

6. Conclusion

In this paper, we present ER-MAC, a hybrid MAC protocol for emergency response WSNs with flexibility to adapt well to traffic and topology changes. ER-MAC schedules collision-free slots, so during the normal monitoring, nodes only wake up for their scheduled slots, but otherwise sleep to save energy. During an emergency, nodes that participate in the emergency monitoring change their MAC behaviour by allowing contention in each slot to achieve high delivery ratio and low latency, but have to sacrifice energy efficiency. ER-MAC is designed to prioritise high priority packets. It also offers a synchronised and loose slot structure, where nodes can modify their schedules locally. Our ns-2 simulation results demonstrate the scalability of ER-MAC and show that ER-MAC achieves higher delivery ratio and lower latency at low energy consumption compared to Z-MAC.

Our future work includes the incorporation of dynamic link estimation, such as using the Received Signal Strength Indicator (RSSI) [38], into ER-MAC. We will also consider network security, which is an important aspect in emergency response application, to prevent an attacker from switching the network into emergency mode as often as possible to deplete nodes' energy and flooding the network with high priority packets to fill in the queue with bogus data.

Acknowledgment

This research is funded by the Irish Higher Education Authority PRTLI-IV research program through the NEMBES project and by the Science Foundation Ireland through the CTVR project (SFI CSET 10/CE/I1853).

References

[1] W. Ye, J. Heidemann, D. Estrin, An Energy-Efficient MAC Protocol for Wireless Sensor Networks, in: Proc. 21st Ann. Joint Conf. IEEE Computer and Communications Societies (INFOCOM'02), 2002, pp. 1567–1576.

[2] T. van Dam, K. Langendoen, An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks, in: Proc. 1st Int'l Conf. Embedded Networked Sensor Systems (SenSys'03), 2003, pp. 171–180.

[3] J. Polastre, J. Hill, D. Culler, Versatile Low Power Media Access for Wireless Sensor Networks, in: Proc. 2nd Int'l Conf. Embedded Networked Sensor Systems (SenSys'04), 2004, pp. 95–107.

[4] M. Buettner, G. V. Yee, E. Anderson, R. Han, X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks, in: Proc. 4th Int'l Conf. Embedded Networked Sensor Systems (SenSys'06), 2006, pp. 307–320.

[5] P. Hurni, T. Braun, MaxMAC: A Maximally Traffic-Adaptive MAC Protocol for Wireless Sensor Networks, in: J. S. Silva, B. Krishnamachari, F. Boavida (Eds.), Proc. 7th European Conf. Wireless Sensor Networks (EWSN'10), Vol. 5970 LNCS, 2010, pp. 289–305.

[6] I. Rhee, A. Warrier, M. Aia, J. Min, Z-MAC: A Hybrid MAC for Wireless Sensor Networks, in: Proc. 3rd Int'l Conf. Embedded Networked Sensor Systems (SenSys'05), 2005, pp. 90–101.

[7] G. Ahn, E. Miluzzo, A. T. Campbell, S. G. Hong, F. Cuomo, Funneling-MAC: A Localized, Sink-Oriented MAC for Boosting Fidelity in Sensor Networks, in: Proc. 4th Int'l Conf. Embedded Networked Sensor Systems (SenSys'06), 2006, pp. 293–306.

[8] Z. Merhi, M. Elgamel, M. Bayoumi, EB-MAC: An Event Based Medium Access Control for Wireless Sensor Networks, in: Proc. 2009 IEEE Int'l Conf. Pervasive Computing and Communications (PerCom'09), 2009, pp. 1–6.

[9] L. Sitanayah, C. J. Sreenan, K. N. Brown, Poster Abstract: Emergency Response MAC Protocol (ER-MAC) for Wireless Sensor Networks, in: Proc. 9th ACM/IEEE Int'l Conf. Information Processing in Sensor Networks (IPSN'10), 2010, pp. 364–365.

[10] L. Sitanayah, C. J. Sreenan, K. N. Brown, ER-MAC: A Hybrid MAC Protocol for Emergency Response Wireless Sensor Networks, in: Proc. 4th Int'l Conf. Sensor Technologies and Applications (SENSORCOMM'10), 2010, pp. 244–249.

[11] S. Chen, H. Bao, X. Zeng, Y. Yang, A Fire Detecting Method based on Multi-Sensor Data Fusion, in: Proc. IEEE Int'l Conf. Systems, Man and Cybernetics (SMC'03), 2003, pp. 3775–3780.

[12] D. T. Gottuk, M. J. Peatross, R. J. Roby, C. L. Beyler, Advanced Fire Detection Using Multi-Signature Alarm Algorithms, Fire Safety Journal 37 (2002) 381–394.

[13] T. Tabirca, K. N. Brown, C. J. Sreenan, A Dynamic Model for Fire Emergency Evacuation Based on Wireless Sensor Networks, in: Proc. 8th Int'l Symp. Parallel and Distributed Computing (ISPDC'09), 2009.

[14] A. El-Hoiydi, J. D. Decotignie, WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks, in: S. E. Nikoletseas, J. D. P. Rolim (Eds.), Proc. 1st Int'l Workshop Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS'04), Vol. 3121 LNCS, 2004, pp. 18–31.

[15] H. Gong, M. Liu, Y. Mao, L. Chen, L. Xie, Traffic Adaptive MAC Protocol for Wireless Sensor Network, in: X. Lu, W. Zhao (Eds.), Proc. 3rd Int'l Conf. Computer Network and Mobile Computing (ICCNMC'05), Vol. 3619 LNCS, 2005, pp. 1134–1143.

[16] V. Rajendran, K. Obraczka, J. J. Garcia-Luna-Aceves, Energy-Efficient Collision-Free Medium Access Control for Wireless Sensor Networks, in: Proc. 1st Int'l Conf. Embedded Networked Sensor Systems (SenSys'03), 2003, pp. 181–192.

[17] V. Rajendran, J. J. Garcia-Luna-Aceves, K. Obraczka, Energy-Efficient, Application-Aware Medium Access for Sensor Networks, in: Proc. 2nd IEEE Int'l Conf. Mobile Adhoc and Sensor Systems (MASS'05), 2005.

[18] E. Egea-Lopez, J. Vales-Alonso, A. S. Martinez-Sala, J. Garcia-Haro, P. Pavon-Marino, M. V. Bueno-Delgado, A Real-Time MAC Protocol for Wireless Sensor Networks: Virtual TDMA for Sensors (VTS), in: W. Grass, B. Sick, K. Waldschmidt (Eds.), Proc. 19th Int'l Conf. Architecture of Computing Systems (ARCS'006), Vol. 3894 LNCS, 2006, pp. 382–396.

[19] T. Zheng, S. Radhakrishnan, V. Sarangan, PMAC: An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks, in: Proc. 19th IEEE Int'l Symp. Parallel and Distributed Processing (IPDPS'05), 2005, pp. 65–72.

[20] G. P. Halkes, K. G. Langendoen, Crankshaft: An Energy-Efficient MAC-Protocol for Dense Wireless Sensor Networks, in: K. Langendoen, T. Voigt (Eds.), Proc. 4th European Conf. Wireless Sensor Networks (EWSN'07), Vol. 4373 LNCS, 2007, pp. 228–244.

[21] J. Kim, J. Lim, C. Pelczar, B. Jang, RRMAC: A Sensor Network MAC for Real Time and Reliable Packet Transmission, in: Proc. IEEE Int'l Symp. Consumer Electronics (ISCE'08), 2008, pp. 1–4.

[22] M. Ringwald, K. Römer, BurstMAC - An Efficient MAC Protocol for Correlated Traffic Bursts, in: Proc. 6th Int'l Conf. Networked Sensing Systems (INSS'09), 2009, pp. 1–9.

[23] K. K. Chintalapudi, i-MAC - A MAC that Learns, in: Proc. 9th ACM/IEEE Int'l Conf. Information Processing in Sensor Networks (IPSN'10), 2010, pp. 315–326.

[24] T. Melodia, M. C. Vuran, D. Pompili, The State of the Art in Cross-Layer Design for Wireless Sensor Networks, in: Proc. EuroNGI Workshops Wireless and Mobility, 2005, pp. 78–92.

[25] A. Boukerche, R. W. N. Pazzi, R. B. Araujo, Fault-Tolerant Wireless Sensor Network Routing Protocols for the Supervision of Contex-Aware Physical Environments, Journal of Parallel and Distributed Computing 66 (4) (2006) 586–599.

[26] S. Ganeriwal, R. K. M. B. Srivastava, Timing-sync Protocol for Sensor Networks, in: Proc. 1st Int'l Conf. Embedded Networked Sensor Systems (SenSys'03), 2003, pp. 138–149.

[27] J. Elson, D. Estrin, Time Synchronization for Wireless Sensor Networks, in: Proc. 15th IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS'01), 2001.

[28] S. Severi, D. Dardari, Performance Limits of Time Synchronization in Wireless Sensor Networks, in: Proc. IEEE Int'l Conf. Communications (ICC'08), 2008, pp. 2124–2128.

[29] M. Maróti, B. Kusy, G. Simon, A. Lédeczi, Robust Multi-Hop Time Synchronization in Sensor Networks, in: Proc. Int'l Conf. Wireless Networks (ICWN'04), 2004, pp. 454–460.

[30] K. Akkaya, M. F. Younis, An Energy-Aware QoS Routing Protocol for Wireless Sensor Networks, in: Proc. 23rd IEEE Intl Conf. Distributed Computing Systems (ICDCS'03), 2003, pp. 710–715.

[31] R. Kumar, R. Crepaldi, H. Rowaihy, A. F. H. III, G. Cao, M. Zorzi, T. F. L. Porta, Mitigating Performance Degradation in Congested Sensor Networks, IEEE Transactions on Mobile Computing 7 (6) (2008) 682–697.

[32] R. S. Dubey, R. Choubey, A. Dubey, Mitigating Congestion Aware Routing Protocol in Wireless Sensor Networks, Int'l Journal Engineering Science and Technology 2 (12) (2010) 7395–7400.

[33] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, T. Abdelzaher, Real-time Power-Aware Routing in Wireless Sensor Networks, in: Proc. 14th IEEE Workshop Quality of Service (IWQoS'06), 2006, pp. 83–92.

[34] The Network Simulator - ns-2, available at `http://www.isi.edu/nsnam/ns/` [30 April 2010].

[35] S. Wenming, H. Chuanhe, S. Mingkai, C. Yong, C. Zhe, Indoor Localization Scheme in Wireless Sensor Networks Using Spatial Information, in: Proc. Int'l Conf. Wireless Communications, Networking and Mobile Computing (WiCOM'06), 2006, pp. 1–5.

[36] Tmote Sky Datasheet, available at `http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf` [30 April 2010].

[37] Z-MAC: Hybrid MAC for Wireless Sensor Networks, available at `http://www4.ncsu.edu/~rhee/export/zmac/software/zmac/zmac.htm` [30 April 2010].

[38] K. Srinivasan, P. Levis, RSSI is Under Appreciated, in: Proc. 3rd Workshop Embedded Networked Sensors (EmNets'06), 2006.