Sensors **2011**, *11*, 2505-2524; doi:10.3390/s110302505

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

W-MAC: A Workload-Aware MAC Protocol for Heterogeneous Convergecast in Wireless Sensor Networks

Ming Xia, Yabo Dong * and Dongming Lu

College of Computer Science and Technology, Zhejiang University, No. 38, Zhe-Da Road, Hangzhou, 310027 Zhejiang, China; E-Mails: nmlab_xiaming@zju.edu.cn (M.X.); ldm@zju.edu.cn (D.L.)

* Author to whom correspondence should be addressed; E-Mail: dongyb@zju.edu.cn; Tel.: +86-571-87952724; Fax: +86-571-87952724.

Received: 30 November 2010; in revised form: 19 January 2011 / Accepted: 14 February 2011 / Published: 28 February 2011

Abstract: The power consumption and latency of existing MAC protocols for wireless sensor networks (WSNs) are high in heterogeneous convergecast, where each sensor node generates different amounts of data in one convergecast operation. To solve this problem, we present W-MAC, a workload-aware MAC protocol for heterogeneous convergecast in WSNs. A subtree-based iterative cascading scheduling mechanism and a workload-aware time slice allocation mechanism are proposed to minimize the power consumption of nodes, while offering a low data latency. In addition, an efficient schedule adjustment mechanism is provided for adapting to data traffic variation and network topology change. Analytical and simulation results show that the proposed protocol provides a significant energy saving and latency reduction in heterogeneous convergecast, and can effectively support data aggregation to further improve the performance.

Keywords: wireless sensor network; heterogeneous convergecast; MAC protocol; TDMA

1. Introduction

A wireless sensor network (WSN) consists of a large number of low cost, low power sensor nodes that perform data sensing tasks. Convergecast is a typical communication pattern in WSNs, where sensor nodes in the network send data to the sink node periodically. Currently existing MAC protocols for convergecast in WSNs are mostly based on the assumption that each sensor node generates exactly the same amount of data at the same rate. However, in real deployment, this assumption frequently

does not hold. Sensor nodes may sense different amounts of data (e.g., they may be equipped with different types or numbers of sensors), or they may have different data reporting frequency configurations. This type of convergecast can be formulated as heterogeneous convergecast, where each sensor node generates different amounts of data in one convergecast operation. In this occasion, existing convergecast MAC protocols can not effectively adapt to variable data traffic on sensor nodes, and cause a great degradation in the overall performance of the network.

This paper presents W-MAC, a workload-aware MAC protocol for heterogeneous convergecast in WSNs. W-MAC employs a subtree-based iterative cascading scheduling mechanism, and a workload-aware time slice allocation mechanism to minimize the power consumption of nodes, while offering a low data latency. In addition, W-MAC provides an efficient schedule adjustment mechanism to adapt to data traffic variation and network topology changes. Analytical and simulation results show that W-MAC outperforms existing protocols in both power consumption and data latency, and can effectively support data aggregation to further improve the performance.

The rest of the paper is organized as follows: Section 2 outlines related work. Section 3 describes our scheduling algorithm and Section 4 provides detailed descriptions on the schedule establishment and adjustment. Section 5 presents the evaluation results and Section 6 concludes the paper.

2. Related Work

MAC protocols for WSNs mostly provide wakeup/sleep schedules for sensor nodes to reduce power consumption, and can be roughly categorized as either contention-based [1] or TDMA-based [2]. Although TDMA-based protocols frequently tend to pose a heavier burden on schedule maintenance, they do not suffer from collisions. This is in agreement with recent research showing that TDMA is preferred for the communications in WSNs [3]. There are also other MAC approaches such as CDMA and multi-channel; we will not discuss them in this paper as they typically pose higher requirements on node capability.

In convergecast, if not considering their special data flow direction from sensor nodes to sink nodes, wakeup/sleep schedules will cause the “data forwarding interruption problem” and bring about a high data latency [4]. Therefore, a number of MAC protocols specially designed for convergecast in WSNs have been proposed to alleviate the problem.

DMAC [4] gives the schedule of a node an offset that depends upon its level (the number of hops to the sink node) on the tree. However, DMAC is not collision-free, since nodes in the same level own the same slot to transmit data. To reduce collisions, DMAC requires the node to perform random backoff before trying to transmit data. If the channel is unavailable, the node must wait for 5 slots to retry. MERLIN [5] and QDMAC [6] adopt a similar scheduling rule as in DMAC. As a result, they also suffer from collisions.

In contrast to DMAC, LL-MAC [7] adopts a level-by-level scheduling scheme. It divides the data transfer period into several non-overlapping uniform divisions, and assigns each level of sensor nodes one division. It then allocates each node a set of unique slots from the division to enable collision-free data transfer. Because the data traffic in each level is different, part of the slots are wasted. At the same time, LL-MAC makes the node cache all data records from its children before relaying, thus causes a high memory usage. LL-MAC also considers the scheduling for network control packets (e.g., time

synchronization and route discovery) transfer. However, the control interval of LL-MAC is long and requires all nodes to keep awake in the whole control interval, thus consuming excessive energy.

The above protocols do not work well in a heterogeneous convergecast scenario. For DMAC, an extra-large data record must be divided into multiple segments to be transmitted in multiple slots, and the node can only perform transmission once every 5 slots. LL-MAC does not provide extra slots if the data record generated by a node is too long to be transmitted in one slot, as a result, the uniform slot assigned by LL-MAC must be larger than the longest data record and the nodes which have shorter data records cannot fully use their slots. In addition, none of the above protocols can effectively support data aggregation. DMAC will immediately forward the data once received, thus no time is spared for data aggregation. The traffic reduction brought by data aggregation will not benefit the performance of LL-MAC, since its slots allocation is only based on the number of descendants of the node.

There are some other similar works that focus on the data transmission time scheduling in convergecast. They can be divided into two categories: (1) works that aim to minimize the convergecast time or energy cost [8-14]. These methods only realize collision-free transmission in the case that each node generates the same amount of data at the same rate; (2) works that aim to maximize data aggregation efficiency. Some of them [15,16] employ the similar idea as that in DMAC, thus they also suffer from collisions. Some researchers have tried to avoid collisions in data aggregation [17-19], however, these TDMA scheduling approaches restrict the nodes to have to aggregate received packets into one packet to be transmitted in one time slot. As a result, these scheduling approaches are eventually not designed for heterogeneous convergecast, and at the same time, many aggregation methods, especially those lossless aggregation functions such as packing aggregation [20], will be not applicable.

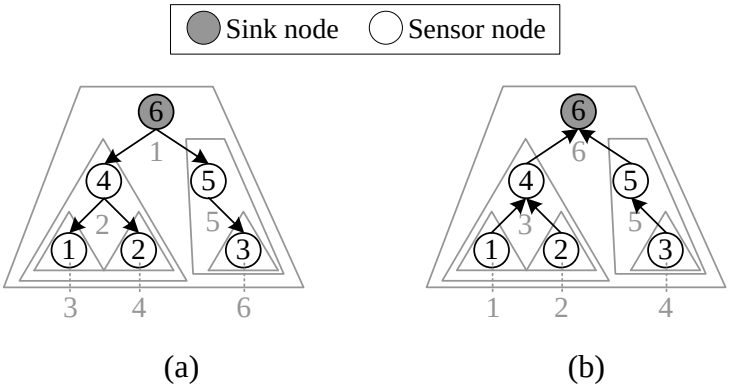
3. Scheduling Algorithm for Heterogeneous Convergecast

As mentioned before, level-by-level data transfer as in LL-MAC wastes slots and brings extra data latency. We alternatively took a subtree as the unit, and designed a collision-free iterative cascading scheduling mechanism, which makes nodes perform network control and data delivery operations subtree-by-subtree. The scheduling algorithm is divided into control interval scheduling and data interval scheduling. For the control interval scheduling, the control packet disseminations are performed from the sink node to the furthest sensor nodes (*i.e.*, nodes with the largest number of hops) subtree-by-subtree, as in Figure 1(a); for the data interval scheduling, the data packet deliveries are performed from the furthest sensor nodes to the sink node subtree-by-subtree, as in Figure 1(b). Typically, the control packet disseminations are performed first, and then the data packet deliveries will be performed. Another key technique in the scheduling algorithm is the workload-aware time slice allocation mechanism for minimizing the active time of nodes. In our scheduling, the workload of a node is essentially the communication workload of the subtree which is rooted at the node.

The workload W of a node is defined as [WO, WR, WT]. WO is the time required for the control packet from the node to reach all its descendants, WR is the time required for the node to collect data from all its descendants and WT is the time required for the node to transmit all data generated by its

descendants and itself. *WO* is referred to as control workload; *WR* and *WT* are referred to as data workload.

Figure 1. Scheduling sequence in W-MAC: **(a)** control packet delivery and **(b)** data packet delivery.



(a) (b)

The scheduling is based on the workloads collected from the children, and the scheduling process only involves the node and its parent. The parent node first allocates a time pool *B* to each child according to the child's workload *W*. The time pool *B* is defined as $[B^S, B^L]$, in which B^S is the start time of *B*, and B^L is the length of *B*. The time pools are categorized as: control time pool B_C, the time pool for control packet disseminations, and data time pool B_D, the time pool for data packet deliveries. Then the node will calculate and acquire its time slices from the time pool obtained from the parent. The time slices can be categorized as:

- *RX*: The time slice for receiving packet. *RX* is defined as $[RX^S, RX^L]$, in which RX^S is the start time of *RX* and RX^L is the length of *RX*. *RX* can be further categorized as *RX* for control interval (RX_C) and *RX* for data interval (RX_D).
- *TX*: The time slice for transmitting packet. *TX* is defined as $[TX^S, TX^L]$, in which TX^S is the start time of *TX* and TX^L is the length of *TX*. *TX* can be further categorized as *TX* for control interval (TX_C) and *TX* for data interval (TX_D).

Therefore, the active time of a node in one working cycle can be represented as $RX_C \cup TX_C \cup \{RX_D\} \cup TX_D$. One node may have multiple RX_Ds, but will have, and only have, one each of the other types of time slices. The node will be kept in sleep in the rest of the time. We will then elaborate on the time pool allocation and time slice calculation algorithms in W-MAC. Other variables used are listed below:

- C_i: The set of direct children of the node u_i.
- T_C: The time required for transmitting one control packet.

3.1. Control Interval Scheduling

The control workload *WO* of a node is determined by the number of descendants of the node. For the node u_i, its control workload WO_i can be calculated from all its children's control workloads by using Equation (1):

$$WO_i = \sum_{u_j \in C_i} WO_j + T_C \tag{1}$$

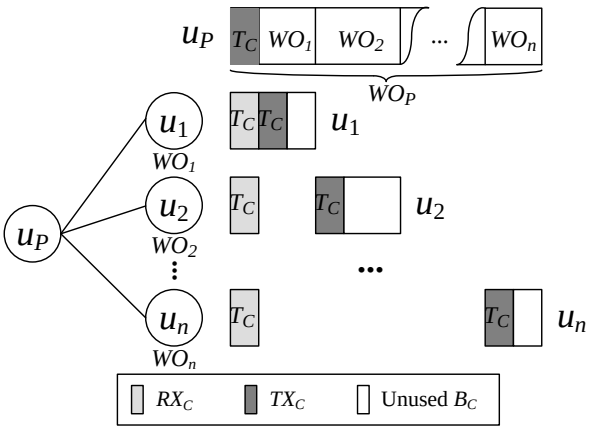
After getting all children's control workloads, the parent can calculate the control time pool allocations for its children. For the node u_i, the control time pool for the child node u_j can be calculated using Equation (2), in which B_{Ck}^L indicates the length of the kth node's control time pool whose allocation is before the node u_i:

$$\begin{cases} B_{Cj}^S = B_{Ci}^S + T_C + \sum_{k=1}^{j-1} B_{Ck}^L, \forall u_j \in C_i \\ B_{Cj}^L = WO_j \end{cases} \tag{2}$$

The node will calculate its RX_C and TX_C locally after obtaining the control time pool allocation from the parent. For the node u_i, RX_{Ci}^S is equal to TX_C^S of its parent, and RX_{Ci}^L is equal to T_C. TX_{Ci}^S is equal to B_{Ci}^S, and TX_{Ci}^L is also equal to T_C.

An example of the control interval scheduling process is shown in Figure 2. Let the number of children of the node u_P be n, and the control workload of the child node $u_j (1 \le j \le n)$ be WO_j. Then WO_P is $\sum_{j=1}^{n} WO_j + T_C$. In scheduling, T_C amount of time at the beginning of the control time pool of the node u_P is reserved for u_P to transmit control packet, and the rest of the time is allocated to its children. It can be observed that the active time of a node in the control interval is a constant, and achieves the minimum value $2T_C$.

Figure 2. Scheduling for the control interval.



3.2. Data Interval Scheduling

The data workload *WR* and *WT* of a node is determined by the number of descendants of the node, the amount of data generated by the node and the node's descendants, and the data aggregation rate on the node. For the node u_i, its data workload WR_i and WT_i can be calculated from all its children's data workloads by using Equation (3), in which WTS_i represents the time required for transmitting data generated by the node u_i itself, and R_i represents the data aggregation rate on u_i:

$$\begin{aligned} WR_i &= \sum_{u_j \in C_i} (WR_j + WT_j) \\ WT_i &= (\sum_{u_j \in C_i} WT_j + WTS_i) R_i \end{aligned} \tag{3}$$

After getting all children's data workloads, the parent can calculate the data time pool allocations for its children. For the node u_i, the data time pool for the child node u_j can be calculated using Equation (4), in which B_{Dk}^L indicates the length of the kth node's data time pool whose allocation is before the node u_i:

$$\begin{cases} B_{Dj}^S = B_{Di}^S + \sum_{k=1}^{j-1} B_{Dk}^L, \forall u_j \in C_i \\ B_{Dj}^L = WR_j + WT_j \end{cases} \tag{4}$$

The node will calculate its RX_D and TX_D locally after obtaining the data time pool allocation from the parent. For the node u_i, RX_{Dij} (the time slice for u_i to receive data from the child node u_j) can be calculated using Equation (5), and TX_{Di} can be calculated using Equation (6):

$$\begin{cases} RX_{Dij}^S = B_{Di}^S + \sum_{k=1}^{j-1} B_{Dk}^L + WR_j, \forall u_j \in C_i \\ RX_{Dij}^L = WT_j \end{cases} \tag{5}$$

$$\begin{cases} TX_{Di}^S = B_{Di}^S + WR_i \\ TX_{Di}^L = WT_i \end{cases} \tag{6}$$

An example of the data interval scheduling process is shown in Figure 3. Let the number of children of the node u_P be n, and the data workload of the child node $u_j (1 \le j \le n)$ be WR_j and WT_j. Then WR_P is:

$$\sum_{j=1}^{n} (WR_j + WT_j),$$

and WT_P is $(\sum_{j=1}^{n} WT_j + WTS_P) R_P$.

In scheduling, WT_P amount of time at the end of the data time pool of the node u_P is reserved for u_P to transmit data packet, and the rest of the time is allocated to its children. It can be observed that the active time of a node in the data interval achieves the minimum value required by the node's workload, and there is no slots waste problem.

It should be noted that concurrent transmission is not used in our scheduling. The rationale behind this decision is that collision-free concurrent transmission requires the nodes to know the interference relationship between each other, but the overhead of detecting and maintaining the interference relationship is too high in traffic and topology variable heterogeneous convergecast networks. As a result, we chose to let each node own its unique transmission time slice to completely avoid collisions while keeping the scheduling algorithm light-weight.

Because the subtree-based iterative cascading scheduling mechanism makes the node transmit data after receiving all data from children, the data aggregation scheme can achieve the highest accuracy and efficiency. However, as in LL-MAC, this attribute may lead to buffer overflow because sensor nodes are typically equipped with limited memory space. In order to alleviate this problem, we can

break a single convergecast operation into multiple rounds, and in each round, we transmit only part of the data records. To support multi-rounds convergecast, we extend the data workload of the node u_i to a $m \times 2$ matrix ($[WR_{i,1},WT_{i,1}]$, $[WR_{i,2},WT_{i,2}]$,…$[WR_{i,m},WT_{i,m}]$), where m indicates the number of rounds. Then, we will have m different schedules for one data interval according to the data workload of each round, and the data time pools of the sink node u_r can be represented as:

$$[0, WR_{r,1} + WT_{r,1}], [WR_{r,1} + WT_{r,1}, WR_{r,2} + WT_{r,2}], ..., [\sum_{k=1}^{m-1} WR_{r,k} + WT_{r,k}, WR_{r,m} + WT_{r,m}]$$

if we take the start time of the data interval as 0, as shown in Figure 4.

Figure 3. Scheduling for the data interval.

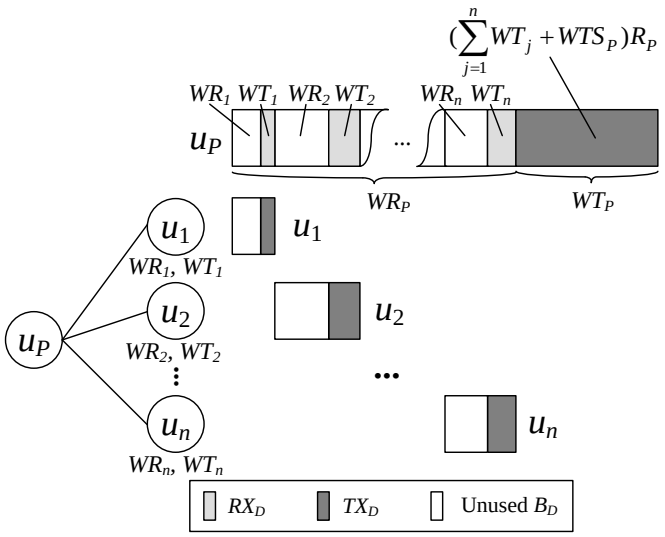
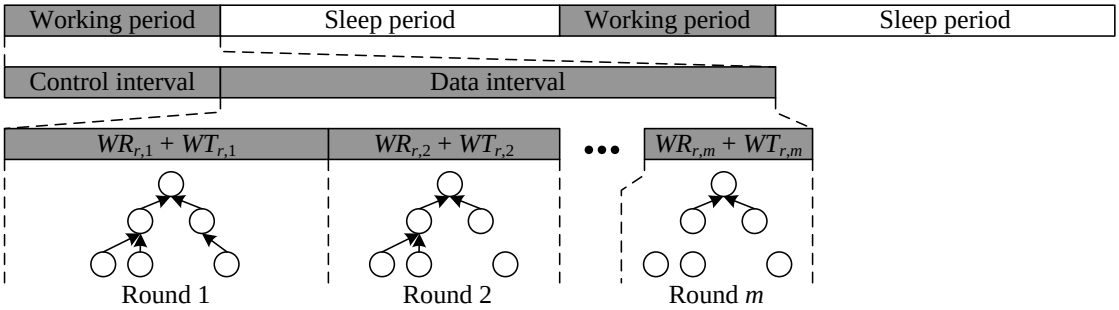


Figure 4. Scheduling of multi-rounds convergecast.



4. Schedule Establishment and Adjustment

4.1. Schedule Establishment

W-MAC makes two assumptions in establishing the schedule: (1) each sensor node knows its parent node; (2) each sensor node knows the amount of data it generates.

Therefore, there are two preliminary actions before establishing the schedule:

- Performing route discovery, then the sensor node can get the information about its parent from the routing layer.

- Acquiring the amount of data generated by the node itself from the application layer.

These two preliminaries do not have any special requirement to the routing and application layer, and thus our MAC scheduling can be directly applied to existing sensor networks. The schedule establishment can be divided into two steps: workload collection and time pool allocation.

4.1.1. Workload Collection

To initialize the workload collection, the sink node broadcasts a "workload collecting" message in the network. Sensor nodes that receive this message will report workloads to their parents. In order to achieve accurate workload calculation, the children must finish reporting before their parent. Therefore we adopted a mechanism similar to "cascading timeouts" [15] to arrange the workload reporting time of nodes, in which the node with larger hop number will report its workload earlier.

4.1.2. Time Pool Allocation

When the workload collection is finished, the sink node initializes the time pool allocation. Each node calculates its time slices, and allocates time pools to its children according to Figure 5. The overhead of the schedule establishment is quite low because: (1) in the workload collection, each node in the network (including the sink node and sensor node) only stores its children's workloads, and reports its workload to the parent (if exists); (2) in the time pool allocation, each node only receives the time pool allocation message from its parent (the sink node can generate its time pool), and notifies its children their time pool allocations. All time slices are calculated locally on the nodes. Therefore, the overhead is almost equally distributed to each node in the network.

Figure 5. Time pool allocation and time slice calculation.

if (u_i is sink){
  B_{Ci} = [0, WO_i];
  B_{Di} = [WO_i, $WR_{i,1}+WT_{i,1}$],⋯,[$WO_i + \sum_{k=1}^{m-1} WR_{i,k} + WT_{i,k}, WR_{i,m} + WT_{i,m}$];
}
else{
  Wait for B_{Ci} and B_{Di};
}

Calculate RX_{Ci}, TX_{Ci}, TX_{Di};

foreach (u_j in children list of u_i) {
  Calculate B_{Cj}, B_{Dj}, RX_{Dij};
  Notify u_j B_{Cj}, B_{Dj} with TX_{Ci};
}

4.2. Schedule Adjustment

Parameters of a WSN system may vary during run time, and this variability will greatly affect the efficiency of the schedule. Variability of the system can be categorized as: (1) data traffic variation. For instance, when the model, number or configuration of the sensors equipped on the node changes,

the workload of that node will vary; (2) network topology changes. The unstable nature of wireless communication makes the topology of a WSN prone to frequent changes. Obviously, node insertion or removal will affect the workload of the parent.

If there is no schedule adjustment mechanism, then we have to reestablish the schedule once the workload of a node or a set of nodes changes. Because the energy consumption of schedule reestablishment is relatively high, the protocol will not be able to keep the node working on an energy-efficient manner. We will then proceed to discuss our schedule adjustment mechanism to ensure the efficiency of our protocol in data traffic and network topology variable scenarios.

4.2.1. Data Traffic Variation

The data traffic variation will only affect the data interval scheduling. In W-MAC, when the data traffic of a node varies, the node will stamp its data packet with a "data traffic varies" mark, which contains $\triangle WT$ (the change of WT). Parent node that receives the data packet with this mark will recalculate and record its $\triangle WR$ (the change of WR) and $\triangle WT$. Then the parent node will stamp its data packet with a "data traffic varies" mark which contains both $\triangle WR$ and $\triangle WT$.

The process repeats until the data packet with the "data traffic varies" mark reaches the sink. Then the sink will perform schedule adjustment in the next control interval. The sink node will first adjust its time pool allocation according to Equation (7), in which m indicates the number of rounds to complete convergecast, and $B_{Dr,k}$ indicates the data time pool allocation of the sink node u_r for the kth round of convergecast:

$$\begin{cases} B_{Dr,k}^{S}{}' = B_{Dr,k}^{S} + \sum_{l=1}^{k-1} (\Delta WR_{r,l} + \Delta WT_{r,l}) \\ B_{Dr,k}^{L}{}' = B_{Dr,k}^{L} + \Delta WR_{r,k} + \Delta WT_{r,k} \end{cases}, \quad \mathbf{1} \le k \le m \tag{7}$$

Then, the schedule adjustment operation will be triggered from the sink node to leaf sensor nodes. Let the number of children of the node u_P be n (u_P's children $\{u_j \mid 1 \le j \le n\}$ are ordered by the sequence in which they appear in the time pool allocation). If the data workload of the child node $u_t (1 \le t \le n)$ varies, u_P will adjust the data time pool allocation according to equation (8), and notify the children in sending control packet:

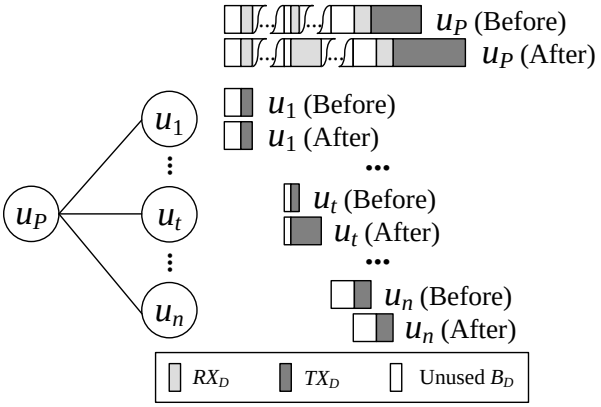
$$\begin{cases} B_{Dj,k}^{S}{}' = B_{Dj,k}^{S} + (B_{Dp,k}^{S}{}' - B_{Dp,k}^{S}) \\ B_{Dj,k}^{L}{}' = B_{Dj,k}^{L} \end{cases}, \quad if \quad j < t$$

$$\begin{cases} B_{Dj,k}^{S}{}' = B_{Dj,k}^{S} + (B_{Dp,k}^{S}{}' - B_{Dp,k}^{S}) \\ B_{Dj,k}^{L}{}' = B_{Dj,k}^{L} + \Delta WR_{t,k} + \Delta WT_{t,k} \end{cases}, \quad if \quad j = t \tag{8}$$

$$\begin{cases} B_{Dj,k}^{S}{}' = B_{Dj,k}^{S} + (B_{Dp,k}^{S}{}' - B_{Dp,k}^{S}) + \Delta WR_{t,k} + \Delta WT_{t,k} \\ B_{Dj,k}^{L}{}' = B_{Dj,k}^{L} \end{cases}, \quad if \quad j > t$$

The schedule adjustment process of a single round of convergecast is shown in Figure 6.

Figure 6. Schedule adjustment for data traffic variation.



The node will recalculate its time slices, and further adjust the data time pool allocations of its children when received the data time pool allocation adjustment notification from the parent.

4.2.2. Network Topology Change

The network topology change will affect both the control and data interval scheduling. The network topology changes include: node insertion, node removal and node changing parent.

A. Node Insertion

Because the scheduling in W-MAC eliminates the idle listening, it is quite hard to detect the node insertion event. Therefore we appended a very short "child admission" time to the end of TX_C. After receiving a control packet, the new node will return a "node insertion" message, which contains its workload information, to the node that broadcasted the control packet. The node that receives the "node insertion" message will stamp its data packet with a "node insertion" mark, which contains the changes of workloads ($\triangle WO$, $\triangle WR$ and $\triangle WT$). The following adjustment process is quite similar to that for data traffic variation, and both the control and data interval scheduling will be adjusted. The new node will keep on listening on the channel after sending the "node insertion" message in this control interval. If a better parent node (mostly determined by routing metrics) is detected, the node changing parent operation will be triggered.

B. Node Removal

If a node does not receive any data from one child for several consecutive working cycles, it will stamp its data packet with a "node removal" mark, which contains the changes of workloads ($\triangle WO$, $\triangle WR$ and $\triangle WT$). The following adjustment process is similar to that for data traffic variation, and will also adjust both the control and data interval scheduling.

C. Node Changing Parent

When a node wishes to change parent, it sends a "node insertion" message in the "child admission" time of the new parent, and proactively sends data packet with the "node removal" mark to the old

parent. By this proactive "node removal" notification, we can finish the node changing parent operation within one working cycle as two notification operations are done in one control and data interval, thus the schedule adjustment will not interrupt the data delivery.

The overhead of the proposed schedule adjustment mechanism in W-MAC is very low because: (1) nodes will proactively report workload changes caused by data traffic variation or network topology change, thus no periodical detection is required; (2) when there is a workload change detected, the schedule adjustment process will be triggered, thus schedule reestablishment is never required; (3) most of the adjustment messages are piggybacked on control or data packets, thus the communication overhead brought by the schedule adjustment is minimized.

5. Evaluation

In this section, we first present the analysis result of performance of W-MAC, and compare it with LL-MAC to prove that W-MAC can outperform LL-MAC. After that, we test W-MAC on NS2 simulator, and compare it with DMAC and LL-MAC to verify that W-MAC successfully achieves its design goals.

The metrics used in evaluation are listed below:

- Power consumption. For DMAC, the energy will only be consumed in data packet transfers; but for LL-MAC and W-MAC, the energy will be consumed in both the data and control packet transfers.
- End-to-end latency. The time required for the data from the furthest sensor node to reach the sink node. This term will be referred to as latency unless otherwise stated.
- Global latency. For DMAC, this term only includes the length of the data interval; but for LL-MAC and W-MAC, it includes the lengths of both the control and data interval. The global latency determines the maximum data sampling rate supported by the protocol.

5.1. Mathematical Analysis

In the discussion below, the data aggregation rate R is set to 1 (*i.e.*, no data aggregation is employed) since LL-MAC does not support data aggregation. At the same time, we consider the case that each data interval contains only one round of convergecast for simplicity. We denote the number of sensor nodes in the network as N, and the data collection cycle (or working cycle) as T_P.

5.1.1. Power Consumption

The power consumption of a sensor node spent on communication (P) can be calculated using Equation (9):

$$P = \frac{P_{RX}T_{RX} + P_{TX}T_{TX} + P_{sleep}(T_P - T_{RX} - T_{TX})}{T_P} \tag{9}$$

In Equation (9), P_{RX}, P_{TX} and P_{sleep} are the power consumption of a sensor node in receiving data, transmitting data, and sleeping, respectively; T_{RX} and T_{TX} are the time spent on receiving and transmitting data. Obviously, we can minimize P by minimizing T_{RX} and T_{TX}.

For W-MAC, the time spent on receiving data ($T_{RX(W\text{-}MAC)}$) and the time spent on transmitting data ($T_{TX(W\text{-}MAC)}$) can be calculated using Equation (10), in which T_A is the length of "child admission" time:

$$\begin{aligned} T_{RX(W-MAC)} &= T_C + 2T_A + \sum_{j\in O_i} WTS_j \\ T_{TX(W-MAC)} &= T_C + \sum_{j\in O_i} WTS_j + WTS_i \end{aligned} \tag{10}$$

For LL-MAC, the time spent on receiving data ($T_{RX(LL\text{-}MAC)}$) and the time spent on transmitting data ($T_{TX(LL\text{-}MAC)}$) can be calculated using Equation (11):

$$\begin{aligned} T_{RX(LL-MAC)} &= (3N+2)T_C + K[O_i]WTS_{\max} \\ T_{TX(LL-MAC)} &= T_C + (K[O_i]+1)WTS_{\max} \end{aligned} \tag{11}$$

In Equation (11), O_i is the set of descendants of the node u_i. $K[O_i]$ is the cardinal of O_i. $WTS_{\max}$ is the length of the uniform slot assigned by LL-MAC, since the length of time slot in LL-MAC depends on the data generation rate of the node that generates the maximum amount of data.

The difference between the power consumption of LL-MAC and W-MAC (ΔP) can be calculated using Equation (12), in which $P_{LL\text{-}MAC}$ is the power consumption of LL-MAC, and $P_{W\text{-}MAC}$ is the power consumption of W-MAC. The proof is described in Appendix.

$$\begin{aligned} \Delta P = P_{LL-MAC} - P_{W-MAC} = \frac{1}{T_P}((P_{TX} - P_{sleep})(\sum_{j\in O_i}(WTS_{\max} - WTS_j) + WTS_{\max} - WTS_i) + \\ (P_{RX} - P_{sleep})((3N+1)T_C - 2T_A + \sum_{j\in O_i}(WTS_{\max} - WTS_j))) \end{aligned} \tag{12}$$

Because the "node insertion" message is small, T_A is frequently shorter than T_C. If we assume that T_A is equal to T_C, then we have:

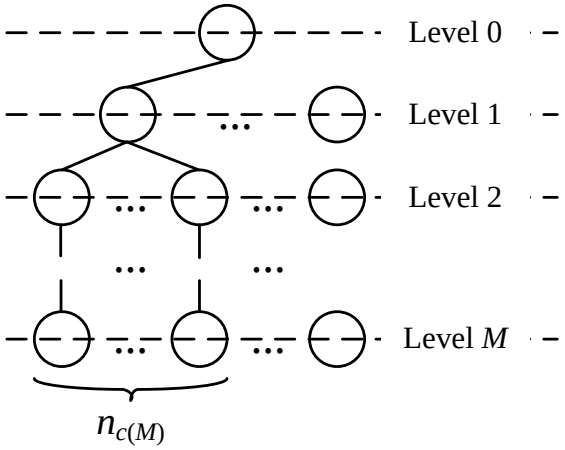
$$\begin{aligned} \Delta P &= \frac{1}{T_P}((P_{TX} - P_{sleep})(\sum_{j\in O_i}(WTS_{\max} - WTS_j) + WTS_{\max} - WTS_i) + \\ &\quad (P_{RX} - P_{sleep})((3N-1)T_C + \sum_{j\in O_i}(WTS_{\max} - WTS_j))) \\ &\geq \frac{1}{T_P}((P_{TX} - P_{sleep})(\sum_{j\in O_i}(WTS_{\max} - WTS_{\max}) + WTS_{\max} - WTS_{\max}) + \\ &\quad (P_{RX} - P_{sleep})((3N-1)T_C + \sum_{j\in O_i}(WTS_{\max} - WTS_{\max}))) \\ &= (P_{RX} - P_{sleep})(3N-1)\frac{T_C}{T_P} \end{aligned} \tag{13}$$

Obviously, ΔP is always a positive number, *i.e.*, W-MAC is more energy efficient than LL-MAC. We can also see that ΔP will grow when there is more significant difference between the maximum and average data generation rate of nodes, which shows the supreme energy efficiency of the scheduling of W-MAC in heterogeneous convergecast scenario. At the same time, ΔP will also be larger when the number of sensor nodes in the network increases, *i.e.*, W-MAC can reserve more energy in large scale network.

5.1.2. Latency

Given a general *M*-level data collection network (as shown in Figure 7), we select the node which is at the level *M*, and will be the first one to transmit data in the leftmost subtree, to calculate the end-to-end data transmission latency. We denote the selected node as u_{M0}, the number of sensor nodes at the level *m* of the subtree which u_{M0} belongs to as $n_{c(m)}$, and the number of sensor nodes at the level *m* of the whole network as $n_{(m)}$.

Figure 7. General data collection tree.



The latency of W-MAC ($D_{W\text{-}MAC}$) can be calculated using Equation (14):

$$D_{W-MAC} = (\sum_{i=1}^{M}(i-1)\sum_{j=1}^{n_{c(i)}} WTS_j) + WTS_{M0} \tag{14}$$

For LL-MAC, its latency ($D_{LL\text{-}MAC}$) can be calculated using Equation (15) according to its level-by-level scheduling behavior:

$$D_{LL-MAC} = ((M-1)N+1)WTS_{\max} \tag{15}$$

The difference between the latency of LL-MAC and W-MAC (ΔD) is:

$$\begin{aligned} \Delta D &= D_{LL-MAC} - D_{W-MAC} = \\ &\sum_{i=1}^{M}((M-1)n_{(i)}WTS_{\max} - (i-1)\sum_{j=1}^{n_{c(i)}} WTS_j) + WTS_{\max} - WTS_{M0} \\ &\geq \sum_{i=1}^{M}(M-i)n_{(i)}WTS_{\max} \end{aligned} \tag{16}$$

The proof is described in Appendix. ΔD is always a positive number. Similar to the power consumption analysis results, we can see that the latency performance of W-MAC will be much better than that of LL-MAC in large scale heterogeneous convergecast network.

5.1.3. Global Latency

The global latency of W-MAC ($GD_{W\text{-}MAC}$) can be calculated using Equation (17):

$$GD_{W-MAC} = (N+1)(T_C + T_A) + \sum_{i=1}^{M} i \sum_{j=1}^{n_{(i)}} WTS_j \tag{17}$$

The global latency of LL-MAC ($GD_{LL\text{-}MAC}$) can be calculated using Equation (18):

$$GD_{LL-MAC} = (3N+3)T_C + MNWTS_{\max} \tag{18}$$

The difference between $GD_{LL\text{-}MAC}$ and $GD_{W\text{-}MAC}$ (ΔGD) is:

$$\begin{aligned} \Delta GD &= GD_{LL-MAC} - GD_{W-MAC} = \\ &(2N+2)T_C - (N+1)T_A + \sum_{i=1}^{M} (Mn_{(i)} WTS_{\max} - i \sum_{j=1}^{n_{(i)}} WTS_j) \end{aligned} \tag{19}$$

Similarly, if we let T_A=T_C, then Equation (19) can be simplified as:

$$\begin{aligned} \Delta GD &= (N+1)T_C + \sum_{i=1}^{M} (Mn_{(i)} WTS_{\max} - i \sum_{j=1}^{n_{(i)}} WTS_j) \\ &\geq (N+1)T_C + \sum_{i=1}^{M} (Mn_{(i)} WTS_{\max} - i \sum_{j=1}^{n_{(i)}} WTS_{\max}) \\ &= (N+1)T_C + \sum_{i=1}^{M} (M-i) n_{(i)} WTS_{\max} \end{aligned} \tag{20}$$

ΔGD is always a positive number. Again, the global latency of W-MAC will be much smaller than that of LL-MAC in large scale heterogeneous convergecast network. We will next examine the performance of W-MAC in the NS2 simulator, and verify the results of the theoretical analysis.

5.2. Simulation Results

In our simulation, we use a time-driven data collection network to test the performance of the proposed protocol. In the network, the data collection cycle is set to 1 minute, and the data generation rate of each sensor node is a uniformly distributed random value within the range of 32–512 bytes/minute. Node parameters are set to the typical values of the Crossbow MICAz mote, as shown in Table 1. In LL-MAC and W-MAC, the convergecast in the data interval is broken into 10 rounds to alleviate the buffer usage. We first set the data aggregation rate R of W-MAC to 1 in protocols comparison for fair competition, and will later show the performance improvement achieved by combining W-MAC and data aggregation.

Table 1. Node Parameters.

Parameter	**Value**
RX power	83.1 mW
TX power	66 mW
Sleep power	0.048 mW
Data rate	250 kbps

5.2.1. Protocol Comparison

Figure 8(a) shows the power consumption simulation results. The average power consumption of W-MAC is only 8% of that of DMAC, and 48% of that of LL-MAC. The excessive power consumption of DMAC mainly comes from the fact that: (1) random backoff increases the length of slot. Meanwhile, the node has to wait for 5 slots before retrying when the channel is unavailable; (2) extra large data record must be divided into multiple segments to be transmitted in multiple slots. The power consumption of LL-MAC is higher than that of W-MAC due to: (1) relatively higher energy consumption in the control interval. Figure 9(a) gives a comparison between the energy consumption of LL-MAC and W-MAC in one control interval. It can be observed that the energy consumption of LL-MAC rises quickly when the scale of the network increases, but the energy consumption of W-MAC is independent of the scale of the network and always keeps at an extremely low level; (2) idle listening caused by uniform slots allocation.

Figure 8(b) shows the simulation results of latency. The average latency of W-MAC is only 11% of that of DMAC, and 33% of that of LL-MAC. The reasons that W-MAC outperforms DMAC in latency are similar to those have been presented in the power consumption simulation results analysis. The latency of LL-MAC is higher than that of W-MAC due to: (1) slots waste problem brought by the level-by-level data transfer scheduling; (2) low channel utilization caused by uniform slots allocation.

Figure 8. Protocols comparison: **(a)** power consumption; **(b)** latency and **(c)** global latency.

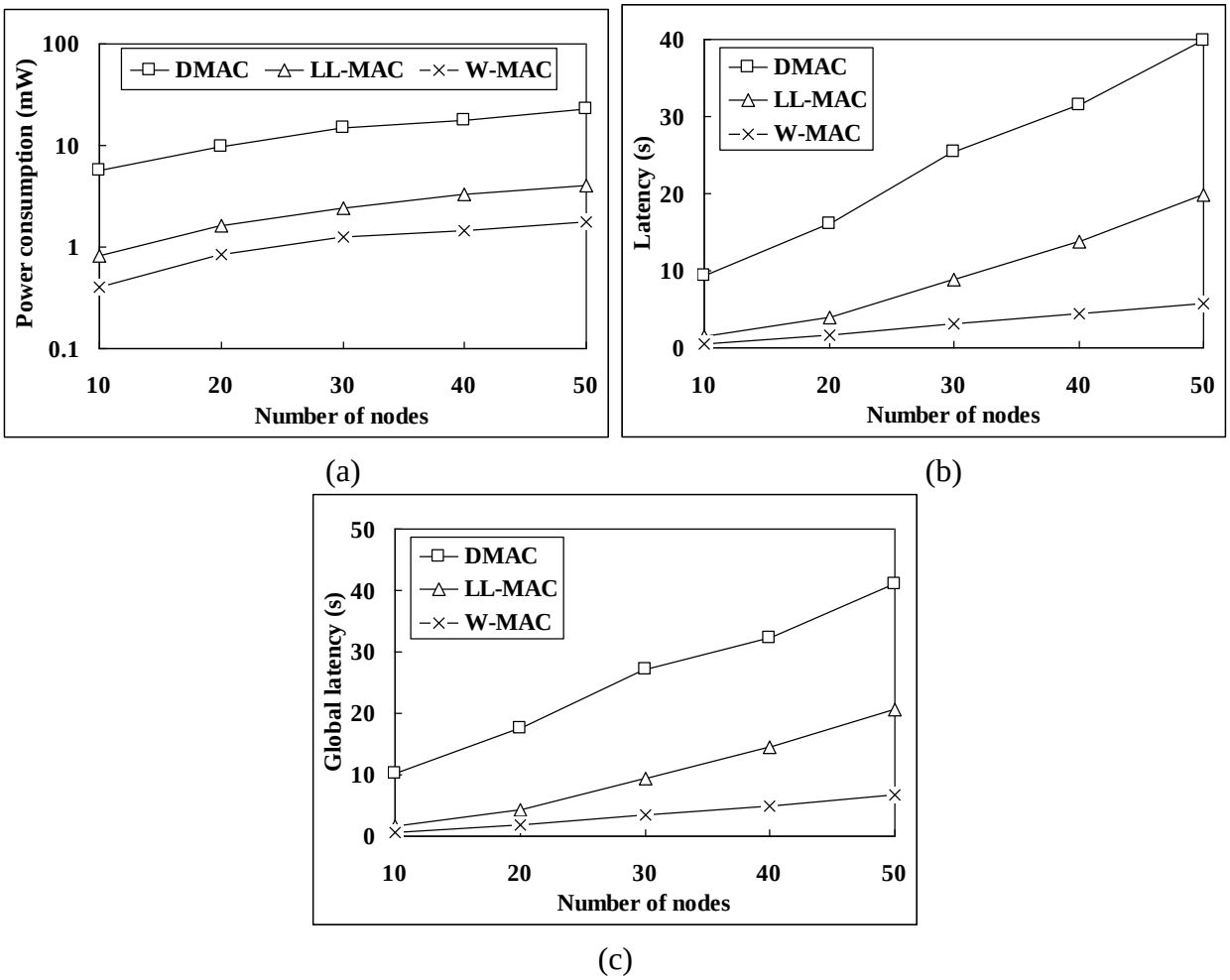
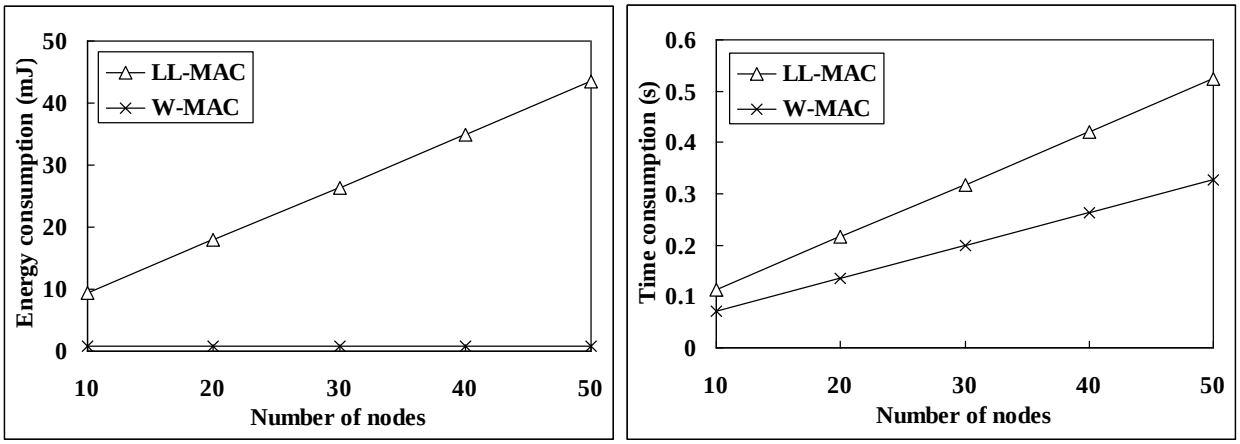


Figure 8(c) shows the simulation results of global latency. The average global latency of W-MAC is only 12% of that of DMAC, and 36% of that of LL-MAC. Besides those have been explained in the latency simulation results analysis, the relatively long control interval of LL-MAC also plays an important part in its high global latency. Figure 9(b) gives a comparison between the control interval length of LL-MAC and W-MAC. It can be observed that the control interval length of LL-MAC increases faster when the network scale increases.

The simulation results verify the correctness of the theoretical analysis. Figure 10 provides a comparison between the analysis and simulation result of ΔP (the difference between the power consumption of LL-MAC and W-MAC). We can see that the analysis result matches the simulation result, and there is only a small difference between them. The reason of this small difference is that the node is not always in TX mode in the time slice for transmitting data, but will occasionally switch to RX mode (e.g., to receive acknowledgement). Because the current in TX mode is lower than that in RX mode for the MICAz mote, the analysis result is a little bit smaller than the simulation result. For latency and global latency, the analysis and simulation results completely match with each other.

Figure 9. The control overhead of LL-MAC and W-MAC: **(a)** energy consumption and **(b)** time consumption.



(a) (b)

Figure 10. Analysis and simulation result of ΔP.

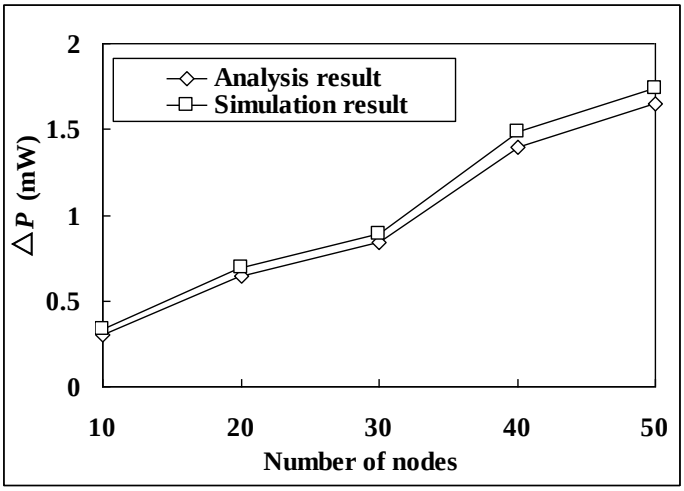
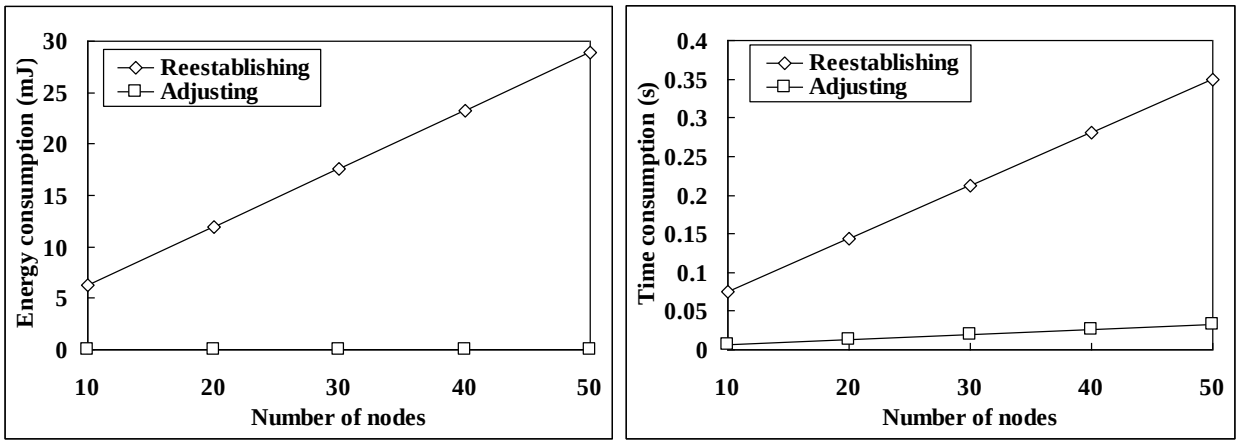


Figure 11 shows the energy and time saving brought by the proposed schedule adjustment mechanism (here we assume that the workload of every node in the network has been changed to make the energy and time consumption of the schedule adjustment mechanism maximized). The period of schedule reestablishment is relatively long, and all nodes have to be kept awake, thus causes extra high energy consumption. As a comparison, through our schedule adjusting, the adjustment messages are piggybacked in normal network control and data delivery packages, and as a result, each node almost does not need to spend additional energy on schedule adjustment, and this advantage can even be kept regardless of the network scale, as shown in Figure 11(a). At the same time, because the schedule adjustment is conducted in normal network control and data delivery operations, the overall time consumption of schedule adjustment is also much lower than that of schedule reestablishment, and increases much slower when the network scale grows, as shown in Figure 11(b).

Figure 11. Comparison between the schedule reestablishment and adjustment: **(a)** energy consumption and **(b)** time consumption.

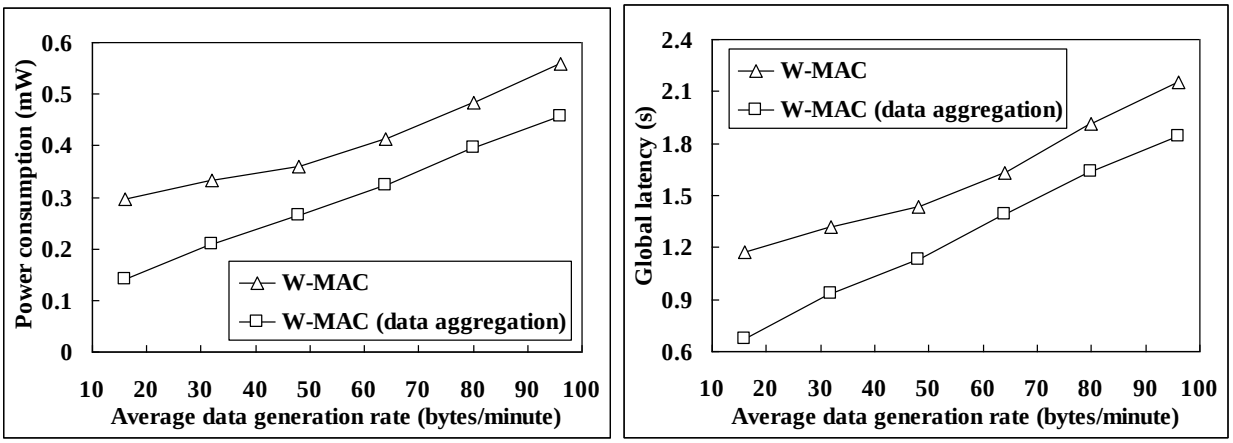


(a) (b)

5.2.2. Collaborating with Data Aggregation

Data aggregation is wildly used in WSNs for reducing data traffic. In this part, we choose packing aggregation, a typical lossless data aggregation technique which packs several non-aggregated packets into one aggregated packet without compression, for evaluation. Figure 12 gives a comparison between the performance of W-MAC with and without data aggregation under different data generation rate conditions (The number of nodes is set to 50). When data aggregation is employed, the data aggregation rate R in the workload W will be reduced, thus W-MAC will shorten the active time of nodes, and then the power consumption and data delivery latency will be lower. The power consumption and global latency of W-MAC with data aggregation can even be reduced to a half of that without data aggregation under low data generation rate condition (16 bytes/minute).

Figure 12. The performance of W-MAC with and without data aggregation: **(a)** power consumption and **(b)** global latency.



(a) (b)

6. Conclusions

This paper presents W-MAC, a workload-aware MAC protocol for heterogeneous convergecast in WSNs. W-MAC adopts a subtree-based iterative cascading scheduling mechanism, and a workload-aware time slice allocation mechanism for minimizing the power consumption of nodes while offering a low data latency. We also present a schedule adjustment mechanism for W-MAC to minimize the energy and time consumption in adapting to workload changes, thus ensure the operation efficiency of W-MAC in data traffic and network topology variable scenarios.

Through extensive theoretical analysis and simulation tests, we compared the performance of W-MAC with existing protocols, including DMAC and LL-MAC, and proved that W-MAC successfully meets the design goals. The average power consumption, latency and global latency of W-MAC are only 48%, 33% and 36% of those of the best competitor. Furthermore, W-MAC can effectively support data aggregation to further improve the performance.

Acknowledgments

This work is supported by the National Basic Research Program of China (973 Program) under grant No. 2006CB303000, and in part by the Science and Technology Program of Zhejiang Province under grant No. 2006C13104, the National Hi-Tech Research and Development Program (863 Program) of China under grant No. 2007AA01Z240.

References

1. Ye, W.; Heidemann, J.; Estrin, D. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '02*, New York, NY, USA, 23–27 June 2002; pp. 1567-1576.
2. Rajendran, V.; Obraczka, K.; Garcia-luna-aceves, J.J. Energy-efficient, collision-free medium access control for wireless sensor networks. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems, SENSYS' 03*, Los Angeles, CA, USA, 5–7 November 2003; pp. 181-192.

3. Ahn, G.S.; Hong, S.G.; Miluzzo, E.; Campbell, A.T.; Coumo, F. Funneling-MAC: A localized, sink-oriented MAC for boosting fidelity in sensor networks. In *Proceedings the 4th ACM Conference on Embedded Networked Sensor Systems, SENSYS '06*, Boulder, CO, USA, 31 October–3 November 2006; pp. 293-306.
4. Lu, G.; Krishnamachari, B.; Raghavendra, C.S. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium, IPDPS'04*, Santa Fe, NM, USA, 26–30 April, 2004; pp. 224-231.
5. Ruzzelli, A.G.; Tynan, R.; O'hare, G. An energy-efficient and low-latency routing protocol for wireless sensor networks. In *Proceedings of 2005 Systems Communications, ICW '05*, Montreal, Canada, 14–17 August 2005; pp. 449-454.
6. Anand, A.; Sachan, S.; Kapoor, K.; Nandi, S. QDMAC: An energy efficient low latency MAC protocol for query based wireless sensor networks. In *Proceedings of 2009 International Conference of Distributed Computing and Networking, ICDCN' 09*, Hyderabad, India, 3–6 January 2009; pp. 306-317.
7. Marin, I.; Arias, J.; Arceredillo, E.; Zuloaga, A.; Losada, I.; Mabe, J. LL-MAC: A low latency MAC protocol for wireless self-organised networks. *Microprocessors Microsystems* **2008**, *32*, 197-209.
8. Choi, H.; Wang, J.; Hughes, E.A. Scheduling for information gathering on sensor network. *Wirel. Netw.* **2009**, *15*, 127-140.
9. Gandham, S.; Zhang, Y.; Huang, Q. Distributed minimal time convergecast scheduling in wireless sensor networks. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, ICDCS'06*, Lisboa, Portugal, 4–7 July 2006; pp. 50-57.
10. Ke, X.; Sun, L.M.; Wu, Z.M. Distributed scheduling for real-time convergecast in wireless sensor networks. *J. Commun.* **2007**, *28*, 44-50.
11. Macedo, M.; Grilo, A.; Nunes, M. Distributed latency-energy minimization and interference avoidance in TDMA wireless sensor networks. *Comput. Netw.* **2009**, *53*, 569-582.
12. Wu, F.J.; Tseng, Y.C. Distributed wake-up scheduling for data collection in tree-based wireless sensor networks. *IEEE Commun. Lett.* **2009**, *13*, 850-852.
13. Egren, S.C.; Varaiya, P. TDMA scheduling algorithms for wireless sensor networks. *Wirel. Netw.* **2010**, *16*, 985-997.
14. Song, W.Z.; Yuan, F.; Lahuseny, R. Time-optimum packet scheduling for many-to-one routing in wireless sensor networks. *Int. J. Parall. Distrib. Sys.* **2007**, *22*, 355-370.
15. Solis, I.; Obraczka, K. The impact of timing in data aggregation for sensor networks. In *Proceedings of 2004 IEEE International Conference on Communications, ICC'04*, Paris, France, 20–24 June 2004; pp. 3640-3645.
16. Li, H.; Yu, H.Y.; Yang, B.W.; Liu, A. Timing control for delay-constrained data aggregation in wireless sensor networks. *Int. J. Commun. Syst.* **2006**, *20*, 875-887.
17. Chen, X.J.; Hu, X.D.; Zhu, J.M. Data gathering schedule for minimal aggregation time in wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2009**, *5*, 321-337.
18. Yu, B.; Li, J.Z.; Li, Y.S. Distributed data aggregation scheduling in wireless sensor networks. In *Proceedings of the 28th IEEE International Conference on Computer Communications, INFOCOM'09*, Rio de Janeiro, Brazil, 19–25 April 2009; pp. 2159-2167.

19. Xu, X.H.; Wang, S.G.; Mao, X.F.; Tang, S.J.; Wang, S.G. A delay efficient algorithm for data aggregation in multi-hop wireless sensor networks. *IEEE Trans. Parall. Distrib. Syst.* **2010**, *22*, 163-175.
20. Chen, J.X.; Yang, Y.H.; Ma, M.D.; Ouyang, Y. Performance study of packing aggregation in wireless sensor networks. *IEICE Trans. Commun.* **2007**, *E90-B*, 160-163.

Appendix

1. The proof of Equation (12):

$$\begin{aligned}
\Delta P &= P_{LL-MAC} - P_{W-MAC} \\
&= \frac{1}{T_P}(P_{RX}((3N+2)T_C + K[O_i]WTS_{\max}) + P_{TX}(T_C + (K[O_i]+1)WTS_{\max}) + \\
&\quad P_{sleep}(T_P - (3N+3)T_C - (2K[O_i]+1)WTS_{\max}) - P_{RX}(T_C + 2T_A + \sum_{j\in O_i} WTS_j) - \\
&\quad P_{TX}(T_C + \sum_{j\in O_i} WTS_j + WTS_i) - P_{sleep}(T_P - 2(T_C + T_A) - 2\sum_{j\in O_i} WTS_j - WTS_i)) \\
&= \frac{1}{T_P}(P_{RX}((3N+1)T_C - 2T_A + \sum_{j\in O_i}(WTS_{\max} - WTS_j)) + P_{TX}(\sum_{j\in O_i}(WTS_{\max} - WTS_j) + WTS_{\max} - WTS_i) + \\
&\quad P_{sleep}(2T_A - (3N+1)T_C - 2\sum_{j\in O_i}(WTS_{\max} - WTS_j) + WTS_i - WTS_{\max})) \\
&= \frac{1}{T_P}(P_{RX}((3N+1)T_C - 2T_A + \sum_{j\in O_i}(WTS_{\max} - WTS_j)) - P_{sleep}((3N+1)T_C - 2T_A + \sum_{j\in O_i}(WTS_{\max} - WTS_j)) + \\
&\quad P_{TX}(\sum_{j\in O_i}(WTS_{\max} - WTS_j) + WTS_{\max} - WTS_i) - P_{sleep}(\sum_{j\in O_i}(WTS_{\max} - WTS_j) + WTS_{\max} - WTS_i)) \\
&= \frac{1}{T_P}((P_{TX} - P_{sleep})(\sum_{j\in O_i}(WTS_{\max} - WTS_j) + WTS_{\max} - WTS_i) + \\
&\quad (P_{RX} - P_{sleep})((3N+1)T_C - 2T_A + \sum_{j\in O_i}(WTS_{\max} - WTS_j)))
\end{aligned}$$

2. The proof of Equation (16):

$$\begin{aligned}
\Delta D &= D_{LL-MAC} - D_{W-MAC} = ((M-1)N+1)WTS_{\max} - (\sum_{i=1}^{M}(i-1)\sum_{j=1}^{n_{c(i)}} WTS_j) - WTS_{M0} \\
&= \sum_{i=1}^{M}(M-1)n_{(i)}WTS_{\max} - (\sum_{i=1}^{M}(i-1)\sum_{j=1}^{n_{c(i)}} WTS_j) + WTS_{\max} - WTS_{M0} \\
&= \sum_{i=1}^{M}((M-1)n_{(i)}WTS_{\max} - (i-1)\sum_{j=1}^{n_{c(i)}} WTS_j) + WTS_{\max} - WTS_{M0} \\
&\geq \sum_{i=1}^{M}((M-1)n_{(i)}WTS_{\max} - (i-1)\sum_{j=1}^{n_{(i)}} WTS_{\max}) + WTS_{\max} - WTS_{\max} \\
&= \sum_{i=1}^{M}((M-1)n_{(i)}WTS_{\max} - (i-1)n_{(i)}WTS_{\max}) \\
&= \sum_{i=1}^{M}(M-i)n_{(i)}WTS_{\max}
\end{aligned}$$