Distributed Node Scheduling with Adjustable Weight Factor for Ad-hoc Networks

Wonseok Lee, Taehong Kim and Taejoon Kim *

School of Information and Communication Engineering, Chungbuk National University, Chungju 28644, Korea; wonseoklee@cbnu.ac.kr (W.L.); taehongkim@cbnu.ac.kr (T.K.)
* Correspondence: ktjcc@chungbuk.ac.kr

Received: 31 July 2020; Accepted: 5 September 2020; Published: 7 September 2020

Abstract: In this paper, a novel distributed scheduling scheme for an ad-hoc network is proposed. Specifically, the throughput and the delay of packets with different importance are flexibly adjusted by quantifying the importance as weight factors. In this scheme, each node is equipped with two queues, one for packets with high importance and the other for packets with low importance. The proposed scheduling scheme consists of two procedures: intra-node slot reallocation and inter-node reallocation. In the intra-node slot reallocation, self-fairness is adopted as a key metric, which is a composite of the quantified weight factors and traffic loads. This intra-node slot reallocation improves the throughput and the delay performance. Subsequently, through an inter-node reallocation algorithm adopted from LocalVoting (slot exchange among queues having the same importance), the fairness of traffics with the same importance is enhanced. Thorough simulations were conducted under various traffic load and weight factor settings. The simulation results show that the proposed algorithm can adjust packet delivery performance according to a predefined weight factor. Moreover, compared with conventional algorithms, the proposed algorithm achieves better performance in throughput and delay. The low average delay while attaining the high throughput ensures the excellent performance of the proposed algorithm.

Keywords: distributed scheduling; weight factor; ad-hoc network; fairness

1. Introduction

An ad-hoc network, as a group of wireless mobile nodes, can be implemented in various forms, including wireless mesh networks, wireless sensor networks, mobile ad-hoc networks, and vehicle ad-hoc networks [1,2]. Ad-hoc networks can provide flexible communication even when it is not possible to install new infrastructure or use existing infrastructure due to geographical and cost restrictions [3]. Ad-hoc networks have the advantage of node communication with other nodes without a base station. Moreover, they also have the features of self-forming and self-healing. Accordingly, they are adopted in various applications, such as battlefield situations, where topology changes frequently, disaster relief, environmental monitoring, smart space, medical systems, and robot exploration [4–8].

Unlike mobile communication networks, which allow centralized resource scheduling, an ad-hoc network requires distributed scheduling based on the information exchanged among nodes. A major problem with distributed node scheduling is packet collisions among nodes if resources are not efficiently distributed, which can lead to significant throughput degradation [9]. Considering these characteristics, supporting quality of service (QoS) through distributed scheduling is a very challenging task. QoS support for high- and low-priority data is essential in various applications. For instance, on a battlefield, a commander’s orders must be delivered as soon as possible.
In addition, for environmental monitoring, it is necessary to send emergency disaster information, such as an earthquake alert, to a destination node with very high priority [10].

The nodes of an ad-hoc network consume a lot of energy in sensing data and processing high-priority packet. However, in many situations, it is difficult to replace or recharge the battery of the nodes. Accordingly, it is important to increase energy efficiency and to enhance overall network lifetime through clustering, transmission power control, and efficient network information exchange [11–16]. Fairness and load balancing among nodes also have a great influence on the battery lifetime and the connectivity of the entire network. However, low fairness among nodes due to inefficient resource allocation causes increased packet collisions and packet retransmission to some nodes, and these detrimental effects reduce the battery lifetime. Meanwhile, some other nodes will be allocated an unnecessarily much amount of resources, resulting in severe inefficiency for the entire network. Hence, resource allocation for an ad-hoc network is a very important and challenging issue.

Fairness measurements can be categorized into qualitative and quantitative methods, depending on whether the fairness can be quantified. Qualitative methods cannot quantify fairness to an actual value, but they can judge whether a resource allocation algorithm achieves a fair allocation. Maximum-minimum fairness [17,18] and proportional fairness [19] are qualitative methods. Maximum-minimum fairness aims to achieve a max-min state, where the resources allocated to a node can no longer be increased without reducing the resources allocated to neighboring nodes. Proportional fair scheduling maximizes the log utility of the whole network by preferentially scheduling nodes with the highest ratios of currently achievable rates to long-term throughput. Measuring the fairness of an entire network is also an important issue. Jain’s fairness index [20] is a quantitative fairness measurement method, however, it cannot measure the fairness of nodes to which a weight factor is assigned.

In this paper, a distributed scheduling algorithm, which takes weight factors and traffic load into account, is proposed. In the proposed algorithm, self-fairness [21] is adopted for resource reallocation. Increment of self-fairness means that resources are fairly allocated to nodes proportionally to the weight of each node. Therefore, even in the distributed scheduling which supports packets with different importance, if the slot allocation for each node is adjusted to the direction of increasing self-fairness, the overall performance of the network can be significantly increased. Moreover, the proposed algorithm adjusts throughput and delay based on the assigned weight factor rather than an absolute distinction between high-priority packets and low-priority packets.

The contribution of this work is summarized as follows:

- A novel distributed scheduling scheme for an ad-hoc network is proposed, where both the load-balancing among neighboring nodes and the preferential processing for high importance packets are considered.
- An intra-node slot reallocation algorithm is proposed. Each node is equipped with multiple queues, and this algorithm re-arranges the slot allocation between the queues inside a node. Moreover, this algorithm enables a flexible adjustment of throughput and delay, reflecting assigned weight factors.
- Self-fairness for packets with unequal importance is introduced. This metric incorporates both the weight factor and traffic load. The metric plays an important role in achieving a fairness among the packets with the same weight factor and in supporting service differentiation among packets with different weight factors. It is validated that the proposed scheduling scheme substantially increases the performance of the network.
- It is confirmed that the proposed node scheduling outperforms the absolute priority-based scheduling scheme in terms of delay and throughput. This result is supported by thorough simulation studies accommodating various operation scenarios.

The remainder of this paper is organized as follows: Section 2 describes the various distributed resource allocation medium access control (MAC) protocols proposed in the literature. Section 3 describes
the proposed algorithm. In Section 4, the performance of the proposed algorithm is analyzed based on an extensive simulation study, and, finally, Section 5 presents some observational conclusions.

2. Related Works

In [22], the authors proposed a distributed randomized (DRAND) time division multiple access (TDMA) scheduling algorithm, which is a distributed version of the randomized (RAND) time slot scheduling algorithm [23]. DRAND operates in a round-by-round manner and it does not require time synchronizations on the round boundaries, resulting in energy consumption reduction. In this scheme, there are four states for each node: IDLE, REQUEST, GRANT, and RELEASE. Each node is assigned a slot that does not cause a collision within the 2-hop neighboring nodes by sending a state message to the neighboring nodes. The basic idea of the deterministic distributed TDMA (DD-TDMA) [24] is that each node collects information from its neighboring nodes to determine slot allocations. DD-TDMA is superior to DRAND in terms of running time and message complexity. This feature increases energy efficiency because DD-TDMA does not need to wait for a GRANT message, which is transmitted as a response of REQUEST message and it contains a slot allocation permission for unused slots. However, DRAND and DD-TDMA do not consider load balancing and fairness among the nodes.

Algorithms for allocating resources based on the states of networks and nodes were proposed in [25–28]. In [25], a load balancing algorithm for TDMA-based node scheduling was proposed. This scheme makes the traffic load semi-equal and improves fairness in terms of delay. In adaptive topology and load-aware scheduling (ATLAS) [26], nodes determine the amount of resources to be allocated through resource allocation (REACT) algorithms, where each node auctions and bids on time slots. Each node acts as both an auctioneer and a bidder at the same time. During each auction, an auctioneer updates an offer (maximum available capacity) and a bidder updates a claim (capacity to bid in an auction). Through this procedure, resources are allocated to the nodes in a maximum-minimum manner [17]. In [27], an algorithm consisting of two sub-algorithms was proposed. The first is a fair flow vector scheduling algorithm (FFVSA) aiming to improve fairness and optimize slot allocation by considering the active flow requirements of a network. FFVSA uses a greedy collision vector method that has less complexity than the genetic algorithm. The second is a load balanced fair flow vector scheduling algorithm (LB-FFVSA), which increases the fairness of the amount of allocated resources among nodes. In [28], the fairness among nodes was improved in terms of energy consumption through an upgraded version of DRAND. Energy-Topology (E-T) factor was adopted as a criterion for allocating time slots, and E-T-DRAND algorithm was proposed to request time slots. Instead of the randomized approach of DRAND, E-T-DRAND algorithm provides high priority to the nodes with high energy consumption and low residual energy due to the large number of neighboring nodes. E-T-DRAND balances the energy consumption among nodes and enhances scheduling efficiency. Each node determines the number of slots to be reallocated using the number of packets accumulated in the queue of its 1-hop neighboring nodes and the number of allocated slots for these nodes. The slot reallocation procedure must check whether a slot is shared by nodes within 2-hop distance. As a result, the load between nodes becomes semi-equal, and the nodal delay is reduced.

In [29–33], scheduling schemes considering priority were proposed. In [29], for the purpose of reducing delay of emergency data, energy and load balanced priority queue algorithm (ELBPQA) was proposed. In this scheme, four different priority levels are defined according to the position of a node in a network. In [30], the highest priority is given to real-time traffic, and the other priority levels are given to non-real time traffics. In order to reduce the end-to-end delay, the packets with the highest priority are processed in a preemptive manner. In [31], priority- and activity-based QoS MAC (PAQMAC) was proposed. In this scheme, the active time of traffic is dynamically allocated according to priority. Specifically, by adopting a distributed channel access scheme, the packet with high priority have reduced back-off and wait times. In [32], I-MAC protocol, which combines carrier sense multiple access (CSMA) and TDMA schemes, was proposed to increase the slot allocation for
nodes with high priority. I-MAC consists of a set-up phase and a transmission phase. The set-up phase consists of neighbor discovery, TDMA time-slot allocation using a distributed neighborhood information-based (DNIB) algorithm, local framing for reuse of time slots, and global synchronization for transmission. Nodes with high priority reduce back-off time to increase the opportunity of winning slot allocation, and nodes with the same priority compete for slot allocation. This scheme reduces the energy consumption of nodes with high priority.

In [33], a QoS-aware media access control (Q-MAC) protocol composed of both intra-node and inter-node scheduling was proposed. Intra-node scheduling determines the priority of packets arriving at the queue of a node. Priority is determined according to the importance of packets and the number of hops to a destination node. Q-MAC consists of five queues, where a queue called an instant queue transmits packets as soon as they arrive. The remaining queues transmit packets following the maximum-minimum fairness principle. Inter-node scheduling is a scheme of data transmission among nodes sharing the same channel. A power conservation MACAW (PC-MACAW) protocol based on the multiple access with collision avoidance protocol for Wireless LANs (MACAW) is applied to schedule data transmission. Q-MAC guarantees QoS through dynamic priority assignment; however, latency can be increased due to heavy computational complexity [34].

A comparative analysis of the protocols mentioned in this section is summarized in Table 1. It is largely classified into with and without prioritizations. In the load-balancing classification, “High” means the clear load-balancing by adopting max-min fairness criterion; “Medium” is an indirect load-balancing method by adjusting idle time and access time; and “Low” is the case where the load-balancing method and its effects are not clearly addressed. In the weight factor classification, “No” is strict priority without quantitative values, and PAQMAC and Q-MAC assign quantitative weight values to packets.

One of the representative fairness measurement methods is Jain’s fairness index, which is a value range (0, 1), and the closer it is to 1 the fairer it is [20]. Jain’s fairness index can measure the fairness of an entire system in a relatively simple way, but it cannot measure the fairness of nodes to which a weight factor is assigned. In [21], the authors proposed a quantitative fairness measurement method applicable to scheduling algorithms with unequal weight factors.

Table 1. Comparative analysis of related works.

| Classification | Protocol | Access Mechanism | Load-Balancing | Weight Factor | Goal |
|----------------|----------|------------------|----------------|--------------|------|
| Without Prioritization | DRAND [22] | TDMA | No | N/A | To allocate resources efficiently in ad-hoc networks |
| LocalVoting [25] | TDMA | High | N/A | To decrease average delay by making the load between neighbor nodes semi-equal |
| ATLAS [26] | TDMA | High | N/A | To adapt topology changes fast and allocate resources considering neighbor nodes |
| ELBPQA [29] | CSMA/CA | Low | No | To minimize delay of high priority packets |
| Algorithm [30] | TDMA | Low | No | To minimize end-to-end delay of high priority packets |
| With Prioritization | I-MAC [32] | CSMA + TDMA | Medium | No | To increase chance of resource allocation for high priority nodes by CSMA + TDMA |
| PAQMAC [31] | CSMA/CA | No | Quantitative | To allocate active time dynamically by considering the priority of packets |
| Q-MAC [33] | CSMA/CA | Medium | Quantitative | To increase energy efficiency while providing service differentiation |

3. Proposed Node Scheduling with Weight Factor

Instead of conventional absolute priority-based scheduling, an adjustable and flexible scheduling scheme is proposed. This scheme reallocates slots by taking the weights assigned to the queues of
nodes into account. Specifically, intra-node scheduling, which reallocates slots between the queues for high- and low-importance packets, is introduced. Then, it is followed by inter-node scheduling adopted from [25], which reallocates slots among neighboring nodes to increase the fairness measured in terms of traffic load.

The proposed algorithm consists of three steps: (1) free time slot allocation, which is a process of allocating the initialized slots (unallocated empty slots) to packets; (2) the intra-node slot reallocation algorithm, which exchanges slots between the queues of a node with different importance values using self-fairness; and (3) the inter-node slot reallocation among 1-hop neighbors using a load balancing algorithm (slot exchange between queues with the same importance). The procedure of this algorithm is depicted in Figure 1.

![Figure 1. Intra-node slot reallocation and inter-node slot reallocation of the proposed scheduling algorithm.](image)

All the nodes have two types of queues for storing packets of different importance. \(Q^H\) and \(Q^L\) are queues for high- and low-importance packets, respectively. \(Q^A\), \(A \in \{H, L\}\) represent \(Q^H\) or \(Q^L\) according to the indicator \(A\), respectively. In the following, \(A\) is used as an indicator representing importance. The number of slots required to transmit all the packets at \(Q^A\) of node \(i\) at frame time \(t\) is represented by \(q^{(A,i)}\), and the number of slots assigned to \(Q^A\) of node \(i\) at frame time \(t\) for packet transmission is represented by \(p^{(A,i)}\). Assuming that the packet and the slot sizes are the same, \(q^{(A,i)}\) is equal to the number of packets in \(Q^A\). \(p^{(A,i)}/q^{(A,i)}\) is the inverse load of \(Q^A\) and expressed as \(\psi^{(A,i)} = p^{(A,i)}/q^{(A,i)}\).

Free time slot allocation requires REQUEST and RELEASE messages exchanges, as in DRAND. The number of packets to be transmitted by node \(i\) is \(q^{(H,i)} + q^{(L,i)}\), and node \(i\) with \(q^{(H,i)} + q^{(L,i)} > 0\) can be allocated slots that are not reserved by the nodes within 2-hop distance. Note that the nodes within 2-hop distance cannot reuse time slot to avoid packet collisions and this reuse can be prevented by slot reallocation between 1-hop nodes. Node \(i\) allocates as many as \(p^{(H,i)}\) slots to \(Q^H\) and increases \(p^{(H,i)}\) by the number of allocated slots. If \(q^{(H,i)} = p^{(H,i)}\), \(Q^H\) does not need to be allocated more slots; accordingly, the slots are allocated to \(Q^L\), and \(p^{(L,i)}\) is increased. Afterwards, \(p^{(H,i)}\) and \(p^{(L,i)}\) are reallocated through the intra-node slot reallocation algorithm. If both \(Q^H\) and \(Q^L\) are allocated as many as \(q^{(H,i)}\) and \(q^{(L,i)}\), no more slots are assigned.
In the intra-node slot reallocation, a self-fairness index is used to reallocate packets between $Q^H$ and $Q^L$ of each node. Self-fairness is a measure of how fairly an amount of "resources" is assigned to a particular node by considering the weight assigned to that node. In this measurement, the resource can be bandwidth, time slots, etc. The proposed algorithm uses inverse load $X_i^{(A,j)}$ as a resource for self-fairness measurement.

In the proposed algorithm, self-fairness applies to two different queues of each node. Hence, each node has two self-fairness values for its two queues ($Q^H$ and $Q^L$). The self-fairness value for $Q^A$ of node $i$ is denoted by $F_i^{(A,j)}$ and defined as it is presented in Equations (1)–(3) [21]:

$$F_i^{(A,j)} = \frac{\log \left( \frac{\phi_i^{(A,j)}}{r_i^{(A,j)} \log A} \right)}{\log(r_i^{(A,j)} / L_i^{(A,j)})}, A \in \{H, L\}$$

$$r_i^{(A,j)} = \frac{X_i^{(A,j)}}{\sum_{k \in N_i} X_k^{(H,j)} + \sum_{k \in N_i^{(1)}} X_k^{(L,j)}}$$

$$L_i^{(A,j)} = \sum_{k \in N_i} p_k^{(H,j)} + p_k^{(L,j)}$$

where $\phi_i^{(A,j)}$ is the ratio of resources allocated to $Q^A$ at node $i$ to the sum of resource allocated to $Q^H$ and $Q^L$ at 1-hop neighboring nodes, $N_i$ is a set of 1-hop neighbor nodes of node $i$, $r_i^{(A,j)}$ is the weight assigned to $Q^A$ of node $i$, and $r_i^{(A,j)}$ is the sum of the weights of 1-hop neighboring nodes. When the weight is high, more slots are allocated to increase the inverse-load, resulting in a fairer resource allocation. By setting $r_i^{(H,j)} > r_i^{(L,j)}$, more important packets are allocated more slots than less important packets. Accordingly, $F_i^{(A,j)}$ is a quantitative value for $Q^A$ of node $i$, indicating whether the load of $Q^A$ is high or low considering the weight assigned. Therefore, it is used as an index to compare the fairness of slot allocation with unequal weight factor.

When $F_i^{(A,j)} = 1$, the allocation is in the fairest state. When the amount of slots allocated is small compared to the assigned weight factor, $F_i^{(A,j)} > 1$ can be satisfied because $\phi_i^{(A,j)} \in [0, 1]$. In this case, it is necessary to gain more slots from the other queue. In the opposite case, if too many slots are allocated, $F_i^{(A,j)} < 1$ can be satisfied, and $Q^A$ must release its own slots. When a slot is gained, $p_i^{(A,j)}$ and $\phi_i^{(A,j)}$ will increase, resulting in a decrement of $F_i^{(A,j)}$. In contrast, when a slot is released, $F_i^{(A,j)}$ increases. The intra-node slots reallocation algorithm adjusts $F_i^{(H,j)}$ and $F_i^{(L,j)}$ to be as close to 1 as possible, which improves the self-fairness. Specifically, when $F_i^{(H,j)} > F_i^{(L,j)}$, the slots allocated to $Q^L$ are released to $Q^H$, and vice versa when $F_i^{(H,j)} < F_i^{(L,j)}$. The algorithm for $F_i^{(H,j)} > F_i^{(L,j)}$ is detailed in Algorithm 1.

In Algorithm 1, $\hat{F}_i^{(H,j)}$ and $\hat{F}_i^{(L,j)}$ are the expected self-fairness values calculated assuming that slots are reallocated. It is assume that $Q^H$ gains a slot from $Q^L$, hence, $\hat{F}_i^{(H,j)}$ is calculated by increasing $p_i^{(H,j)}$ by 1, and $\hat{F}_i^{(L,j)}$ is calculated by decreasing $p_i^{(L,j)}$ by 1. The updated $p_i^{(H,j)}$ and $p_i^{(L,j)}$ are transmitted to its 1-hop neighboring nodes at the end of each frame. Accordingly, during slot exchange at frame time $t$, $\varphi$ is calculated using only the locally updated $p_i^{(H,j)}$ and $p_i^{(L,j)}$ by intra-node slot exchange. In the next frame, the self-fairness is updated through information exchanges among neighboring nodes. When $p_i^{(L,j)} = 1$ and $Q^L$ releases a slot, $p_i^{(L,j)}$ will be 0. This makes $\phi_i^{(L,j)} = 0$, and $\hat{F}_i^{(L,j)}$ becomes infinity. To prevent this, a minimum default value above 0 is assigned to $p_i^{(L,j)}$ under this situation.
Algorithm 1. Increasing $Q^H$ slot allocation

1: for all node $i$ do
2:   if $q_i^{(H,j)}=0$ then
3:      Calculate $F_i^{(H,j)}$
4:   end if
5:   if $q_i^{(L,j)}=0$ then
6:      Calculate $F_i^{(L,j)}$
7:   end if
8:   if $F_i^{(H,j)} > F_i^{(L,j)}$ then
9:      if $p_i^{(L,j)} > 0$ then
10:     $p_i^{(H,j)} ← p_i^{(H,j)} + 1$
11:     $F_i^{(L,j)} ← F_i^{(L,j)} - 1$
12:     $F_i^j ← \sqrt{(1 - F_i^{(H,j)})^2 + (1 - F_i^{(L,j)})^2}$
13:     $F_i^j ← \sqrt{(1 - F_i^{(H,j)})^2 + (1 - F_i^{(L,j)})^2}$
14:   end if
15:   while $F_i^j > F_i^j$ do
16:     $p_i^{(H,j)} ← p_i^{(H,j)} + 1$
17:     $F_i^{(L,j)} ← F_i^{(L,j)} - 1$
18:     if $p_i^{(L,j)} > 0$ then
19:        $F_i^j ← F_i^j$
20:        $F_i^j ← \sqrt{(1 - F_i^{(H,j)})^2 + (1 - F_i^{(L,j)})^2}$
21:     else break;
22:   end if
23: end while
24: end if
25: end if

At every frame, slots are reallocated until self-fairness can no longer be improved. Note that the fairness index $1$ is the fairest state. Consequently, the Euclidean distance between the fairest status $F_i^{(H,j)} = F_i^{(L,j)} = 1$ and a current $(p_i^{(H,j)}, F_i^{(L,j)})$ combination is introduced as a metric representing a target fairness, as it is presented in Equation (4):

$$ F_i^j = \sqrt{(1 - F_i^{(H,j)})^2 + (1 - F_i^{(L,j)})^2} $$ (4)

Now, the expected Euclidean distance $F_i^j$ from the expected fairness $(p_i^{(H,j)}, F_i^{(L,j)})$ is compared with the current Euclidean distance $F_i^j$ from $(p_i^{(H,j)}, F_i^{(L,j)})$. If $F_i^j < F_i^j$, $Q^H$ gains a slot from $Q^L$, and $p_i^{(H,j)}$ and $F_i^{(L,j)}$ are updated. Because slot reallocation is an intra-node process, collisions with 2-hop neighboring nodes need not be considered.

When $F_i^{(H,j)} < F_i^{(L,j)}$, the slot reallocation algorithm is very similar to Algorithm 1, and $p_i^{(H,j)}$ and $F_i^{(L,j)}$ are calculated with $p_i^{(H,j)} - 1$ and $F_i^{(L,j)} + 1$, respectively. However, instead of $p_i^{(L,j)} > 0$ in lines 9 and 18 of Algorithm 1, $p_i^{(H,j)} > 1$ is used as a slot release condition. This prevents $p_i^{(H,j)}$ from being zero by releasing all slots to $Q^L$ to improve the fairness when $q_i^{(H,j)} < q_i^{(L,j)}$. That is, $p_i^{(H,j)} \geq 1$ is guaranteed in any situation.

After the intra-node slot reallocation algorithm, the inter-node slots reallocation [25] follows. At this time, the slot exchange does not consider the weights of $Q^H$ and $Q^L$ any more because these exchanges take place among the queues with the same importance. Node $i$’s $Q^H$ computes
\( u_i^{(A,i)} \) to determine how many slots to reallocate with a 1-hop neighboring node as it is presented in Equation (5) [25]:

\[
 u_i^{(A,i)} = \left[ \frac{q_i^{(A,k)} \cdot \sum_{k \in N_i} p_i^{(A,k)}}{\sum_{k \in N_i} q_i^{(A,k)}} \right] - p_i^{(A,k)} \tag{5}
\]

If \( u_i^{(A,i)} > 0 \), slots are gained from the 1-hop neighboring node. If \( u_i^{(A,i)} < 0 \), slots are released to the 1-hop neighboring node. The number of reallocated slots is determined by \( \min\{u_i^{(A,i)}, u_i^{(A,i)} - u_i^{(A,k)}, p_i^{(A,k)}\} \). This increases the equality of the inverse-load of the same importance among node \( i \) and its 1-hop neighboring nodes. These processes are performed for all nodes in a node-by-node manner. The same intra-node and inter-node slot reallocations are repeated in the next frame.

4. Performance Evaluation

A network simulator [35] implemented in Java was used for performance analysis of the proposed algorithm. No isolated nodes are assumed, i.e., all the nodes have at least a single 1-hop neighbor node. Accordingly, in establishing a connection, any two nodes can be connected with each other through multi-hop links. The connections are established using arbitrarily chosen pairs of a source node and a destination node, and high- and low-importance connections generate high- and low-importance packets, respectively. In the following, high- and low-importance packets are denoted by \( Pkt_H \) and \( Pkt_L \), respectively.

For the performance analysis, the throughput, delay, and fairness are measured by changing the connection creation ratio (between \( Pkt_H \) and \( Pkt_L \)) and the weight factor setting. Then, the proposed algorithm is compared with the absolute priority-based algorithm in which \( Pkt_H \) preempts time slots when allocating free time slots. Note that the absolute priority algorithm adopts only the inter-node slot reallocation algorithm, not the intra-node slot one.

The generation ratios of high- and low-importance connections are denoted by \( \alpha, 1 - \alpha \in [0,1] \). The weight factor setting in \( Q_A \) is denoted by \( r^A \). Assuming that \( Q^H \) and \( Q^L \) of all nodes have the same weight settings as \( r^H \) and \( r^L \), respectively, the node index \( i \) can be dropped from the weight factors. The weight factors are set as: \( r^H, r^L \in [0,10] \) and \( r^H + r^L = 10 \).

The performance of the proposed scheme was measured in two scenarios. Table 2 lists the parameters setting for each scenario. In the first scenario, a fixed number of connections are created at the starting epoch of the simulation, the packets of the connections are generated at fixed time intervals, and the number of packets generated for each connection is the same. In the second scenario, connections are created based on Poisson processes. Unlike the first scenario, the number of packets generated per connection follows a Poisson distribution. The arrival rate \( \lambda \) determines the connection creation interval. The duration of each connection follows an exponential distribution of parameter \( \mu \), which determines the number of packets generated in each connection. The packets are generated at a fixed interval, as in the first scenario. Each connection is closed if all the packets arrive at its destination node. Because the connections are continually generated, in the second scenario, the simulation duration is specified at the beginning of the simulation. For both scenarios, the final measurement is the average over 1000 independent simulations.
Table 2. Simulation parameters.

| Parameter                  | Value                  |
|----------------------------|------------------------|
| Number of nodes            | 30                     |
| Transmission range         | 5 units                |
| Topology size              | 50 × 50 units          |
| Frame length               | 30 time-units          |
| Packet generation interval | 5 time-units           |
| Number of connections $N_c$| 50–500                 |
| Number of packets per connection $N_p$ | 50                  |
| Connection duration $\mu$ | $10^3$ time-units      |
| Arrival rate $\lambda$    | $10^{-3}$–$10^{-1}$ time-units$^{-1}$ |
| Simulation time $T$        | 3000/150,000 time-units |

In the first scenario, the performance of the proposed algorithm was analyzed with the increasing total number of connections and the various settings of the weight factor and $\alpha$. The total number of created connections is the sum of the high- and low-importance connections. Throughput, packet delivery ratio, 1-hop delay, and fairness are measured and compared with those of absolute priority-based scheduling. Throughput refers to the number of all packets arriving at a destination node during the simulation. However, in the first scenario, since the number of generated connections is determined at the beginning of the simulation, the throughput measured when all packets arrive at a destination node will be simply the product of $N_c$ (number of connections) and $N_p$ (number of generated packets per connection). Therefore, throughput is measured not at the end of the simulation but at a predefined time $T$, which is large enough for the transmission of packets in the network to be in a steady state. The packet delivery ratio means the proportion of received packets to the packets sent. The 1-hop delay is measured as the average of ((the time when a packet is dequeued) minus (the time when a packet is enqueued)). The results of the absolute priority-based algorithm are marked as Preempt.$Pkt^H$ and Preempt.$Pkt^L$.

Figures 2–6 show the results of the first scenario. Figure 2 depicts the throughputs with the increasing total number of connections, various weight factors, and $\alpha = 0.3$. When the number of connections is small, most packets are delivered to the destination nodes until the predefined time $T$ because the network is not heavily loaded. For this reason, in Figure 2a,b, when the number of connections is 50, the throughput of $Pkt^H$ is lower than that of $Pkt^L$ because the number of $Pkt^H$ is lower than $Pkt^L$. In most cases, if the number of connections increases, the throughput of $Pkt^H$ is higher than that of $Pkt^L$. However, in Figure 2b, when the weight factors are $r^H = 7$ and $r^L = 3$, the throughput of $Pkt^L$ is higher than that of $Pkt^H$, even when the number of connections increases. Note that the proposed algorithm considers not only the weight factors but the traffic load as well; hence, even when $r^L < r^H$, the throughput of $Pkt^L$ is higher than that of $Pkt^H$ in the entire range of $N_c$. The service differentiation between $Pkt^H$ and $Pkt^L$ is more evidently shown in Figure 2c,d. As shown in these figures, over all the range of the number of connections, the packet delivery ratio of $Pkt^H$ is higher than $Pkt^L$. Specifically, Figure 2b with $r^H = 7$, $r^L = 3$ can be compared with Figure 2d with $r^H = 7$, $r^L = 3$. In this case, Figure 2b shows that the throughput of $Pkt^L$ is higher than $Pkt^H$. However, Figure 2d shows that the packet delivery ratio of $Pkt^H$ is still twice as high as that of $Pkt^L$. This result clearly shows that the proposed scheme preferentially processes packets reflecting the weight factors. When the absolute priority-based algorithm is applied, as the number of $Pkt^H$ to be transmitted increases owing to the increment of the number of connections, the opportunity for $Pkt^L$ slot allocation decreases, resulting in a further decrease in the throughput of $Pkt^L$. 

In Figure 3, throughputs are measured when $r_H \alpha = r_L (1 - \alpha)$ is satisfied under the condition of increasing number of connections. Figure 3 shows the characteristics of the proposed algorithm by considering both the weight factor and traffic load. When $r_H \alpha = r_L (1 - \alpha)$ is satisfied, it is confirmed that the throughputs of $Pkt^H$ and $Pkt^L$ have similar values and converge to a single value, as shown in Figure 3.

As shown in Figures 2 and 3, the sums of the throughputs of $Pkt^H$ and $Pkt^L$ are similar when $N_c$ is the same, even though $\alpha$ and the weight factors are different. This is because even when the number of allocated slots of $Pkt^H$ and $Pkt^L$ are changed by $\alpha$ and the weight factors during the process of reallocation, the number of allocated slots in the entire network does not change. Therefore, there is
a tradeoff between the throughputs of $Pkt^H$ and $Pkt^L$ depending on the weight factors. From Figures 2 and 3, it is confirmed that an appropriate weight factor setting is necessary to adjust the throughputs of $Pkt^H$ and $Pkt^L$ for various network situations with different $\alpha$.

Figure 4 shows 1-hop delay with various weight factors and $\alpha$ with the increasing total number of connections. Similar to in Figures 2 and 3, when the number of connections is small, all the generated packets can be delivered to destination nodes, resulting in nearly no difference in the delay between $Pkt^H$ and $Pkt^L$. However, as the number of connections increases, the delays of both $Pkt^H$ and $Pkt^L$ increase, and the delay difference between $Pkt^H$ and $Pkt^L$ becomes conspicuous. Compared to the absolute priority-based algorithm, the delay gap between $Pkt^H$ and $Pkt^L$ of the proposed algorithm is relatively small. In the case of $r^H = 7$ and $r^L = 3$ shown in Figure 4a, when $N_c$ is 500, the delay of $Pkt^L$ is twice that of $Pkt^H$. On the other hand, the delay of Preemp. $Pkt^L$ is more than 6 times the delay of Preemp.$Pkt^H$. The delay of $Pkt^H$ increases compared to Preemp.$Pkt^H$, but the delay of $Pkt^L$ decreases much more than Preemp.$Pkt^L$. In particular, when $r^H = 9, r^L = 1$, and $N_c = 500$ in Figure 4b, the delay of $Pkt^H$ increases by approximately 500 time slots compared to Preemp.$Pkt^H$, but the delay of $Pkt^L$ decreases by approximately 3000 time slots compared to Preemp.$Pkt^L$, and it is a noticeable improvement. The average sum delay of $Pkt^H$ and $Pkt^L$ is reduced by 20% compared to that of Preemp.$Pkt^H$ and Preemp.$Pkt^L$. This means that, compared to the absolute priority-based algorithm, the proposed algorithm achieves the higher performance. Moreover, the proposed algorithm can achieve the same delay performance with Preemp.$Pkt^H$ by throttling $Pkt^L$, i.e., with $r^H = 10$ and $r^L = 0$. When $\alpha = 0.5$, the number of $Pkt^H$ to be transmitted increases and the delay of $Pkt^H$, at the same $N_c$, increases compared to the case of $\alpha = 0.3$. In the whole range of $N_c$, the delay of $Pkt^H$ in Figure 4b is higher than that of $Pkt^H$ in Figure 4a. In addition, $Pkt^H$'s delay when $r^H = 7$ in Figure 4a and that when $r^H = 9$ in Figure 4b are similar.

![Figure 4](image-url)

**Figure 4.** Delay comparison between priority-based algorithms $Pkt^H$ and $Pkt^L$: (a) 1-hop delay with $\alpha = 0.3$; (b) 1-hop delay with $\alpha = 0.5$. 
In Figures 2 and 4, for $Pkt^H$, the higher $r^H$ is, the better the performances of throughput and delay are. The decrement in $r^L$ due to the increased $r^H$ leads to the worse performance of throughput and delay of $Pkt^L$. The larger the difference between the values of $r^H$ and $r^L$, the larger the performance gap between the throughput and delay of $Pkt^H$ and $Pkt^L$. This confirms that $Pkt^H$ and $Pkt^L$ are flexibly adjusted based on the values of the weight factor in various network situations.

In Figure 5, the proposed scheduling scheme is compared with DRAND, LocalVoting, and Q-MAC. Q-MAC was developed for CSMA/CA and the packets with high weight value had a relatively high probability of accessing channel. For comparison, Q-MAC was modified to be applicable to TDMA. Specifically, the slots of Q-MAC are initialized according to the weight values, and the inter-node reallocation of LocalVoting is followed. As shown in Figure 5a, the delay of $Pkt^H$ is better than both DRAND and LocalVoting, and slightly worse than Q-MAC with $Pkt^H$. Even $Pkt^L$ shows the better performance than DRAND and slightly worse than LocalVoting. Specifically, the delay of DRAND is twice longer than $Pkt^L$ and four times longer than $Pkt^H$. LocalVoting shows the performance better than DRAND through the neighbor-aware load balancing. However, the proposed scheme of $Pkt^H$ still outperforms LocalVoting. The delay of $Pkt^H$ is 1.8 times smaller than LocalVoting. In Figure 5b, the average delay of the proposed scheme shows the best performance. Q-MAC and LocalVoting show the similar performance with each other. In Figure 5c, the throughput of the proposed scheme with $Pkt^H$ is lower than Q-MAC with $Pkt^H$. However, the throughput of the proposed scheme with $Pkt^L$ is higher than Q-MAC with $Pkt^L$. Note that the throughput of LocalVoting in Figure 5c is the sum of its $Pkt^H$ and $Pkt^L$. In Figure 5d, the proposed scheme achieves the highest throughput. In Figure 5b,d, it is ensured that the proposed scheme possesses the excellent performance in slot allocation because it achieves the highest throughput and the lowest delay.

Moreover, Figure 5a,c show that the service differentiation of the proposed scheme is enabled compared with other schemes. These are the major contributions of the proposed scheme.
Figure 6 compares Jain’s fairness \([20]\) of \(Pkt^H\) and \(Pkt^L\) with and without the proposed algorithm. In this figure, in terms of \(\Gamma^{(A,j)}\), Jain’s fairness index shows how fairly resources are allocated among the queues of the same importance. \(\Gamma^{(A,j)}\) is the ratio of the accumulative number of packets transmitted from a queue to the number of accumulated packets in a queue until \(T\), which can be expressed as Equation (6). Similar to the throughput measurement, at the end of the simulation, all packet delivery is completed; accordingly, Jain’s fairness is calculated at time \(T\).

\[
\Gamma^{(A,j)} = \frac{\sum_{i=0}^{T} P_i^{(A,j)}}{\sum_{i=0}^{T} Q_i^{(A,j)}}, \ A \in \{H,L\}
\]  

(6)

In this analysis, \(\alpha = 0.3\) and \(r^H = 7, r^L = 3\) are considered. When the number of connections is small, the fairness index is high regardless of the adoption of the proposed algorithm because the \(\Gamma^{(A,j)}\) of most nodes becomes close to 1. For the absolute priority-based algorithm, as the number of connections increases, only a few nodes are allocated slots for Preempt.\(Pkt^L\). Since most nodes cannot transmit Preempt.\(Pkt^L\), the fairness of Preempt.\(Pkt^L\) is very low. In contrast, when the intra-node slot reallocation of the proposed algorithm is adopted, time slots proportional to \(r^i\) are allocated to \(Q^i\), and this results in an increase in the fairness index. As a result, the fairness performance of \(Pkt^L\) is significantly increased compared to that of \(Pkt^H\) when the intra-node slot exchange algorithm is applied.

![Figure 6. Jain’s fairness comparison between the proposed algorithm and absolute priority-based scheduling.](image)

Figure 7 shows the delay and throughput performance of the second scenario with the increasing Poisson arrival rate \(\lambda\). In Figure 7a,b, because \(\alpha = 0.5\) is applied, the numbers of \(Pkt^H\) and \(Pkt^L\) are similar. Although the connection creation interval and the number of packets generated for each connection are varied, Figure 7 shows similar performances to those of the first scenario. The larger the difference between \(r^H\) and \(r^L\), the greater the performance gap between \(Pkt^H\) and \(Pkt^L\). For instance, in Figure 7a, when the arrival rate is 0.01 time – units\(^{-1}\) and the weight factors are \(r^H = 7\) and \(r^L = 3\), the \(Pkt^L\) delay is approximately 1.5 times longer than the \(Pkt^H\) delay. However, when the weight factors are \(r^H = 9\) and \(r^L = 1\), \(Pkt^L\) delay is over two times \(Pkt^H\) delay. When the arrival rate is low, the connection creation interval is long, and the number of connections created during the entire simulation is small. As shown in Figure 7a,b, when the arrival rates are as low as 0.001 and 0.002 time – units\(^{-1}\), there is only a slight difference in delay and throughput between \(Pkt^H\) and \(Pkt^L\) regardless of the weight factor setting.

Figure 7c shows the throughput when the number of \(Pkt^L\) is larger than that of \(Pkt^H\), by setting \(\alpha = 0.3\). The result of Figure 7c is very similar to that of Figure 2a when \(N_c\) ranges between 100 and 500. In particular, if \(r^H \cdot \alpha = r^L \cdot (1 - \alpha)\) is satisfied by setting \(r^H = 7, r^L = 3\), the throughputs of \(Pkt^H\) and \(Pkt^L\) converge to a constant value. However, note that \(\alpha\) is set as 0.3, i.e., 70% of the generated
packets are \( Pkt^L \) and the remaining 30% is \( Pkt^H \). Even in this asymmetric packet generation scenario, \( Pkt^H \) achieves the higher throughput than \( Pkt^L \). Accordingly, this clearly shows that the service differentiation between \( Pkt^H \) and \( Pkt^L \) is attained.

5. Conclusions

In this paper, a novel distributed node scheduling algorithm for an ad-hoc network was proposed. This scheme flexibly adjusts time slot allocations according to weight factor and traffic load. From thorough simulation studies under various environments, the performance differentiation reflecting weight factor setting was validated. It was confirmed that, as the weight of the high importance packets increases, the delay decreases and the throughput at the same time increases. Because the proposed algorithm considers both the weight factors and traffic loads, even

Figure 7. Delay and throughput with increasing Poisson arrival rates and the same weight factor setting: (a) 1-hop delay with \( \alpha = 0.5 \); (b) throughput with \( \alpha = 0.5 \); (c) throughput with \( \alpha = 0.3 \).
5. Conclusions

In this paper, a novel distributed node scheduling algorithm for an ad-hoc network was proposed. This scheme flexibly adjusts time slot allocations according to weight factor and traffic load. From thorough simulation studies under various environments, the performance differentiation reflecting weight factor setting was validated. It was confirmed that, as the weight of the high importance packets increases, the delay decreases and the throughput at the same time increases. Because the proposed algorithm considers both the weight factors and traffic loads, even the throughput and delay for the same weight factors can be adjusted separately according to the connection creation ratios with different importance. Through comparison with other distributed node scheduling algorithms, the advantages of the proposed algorithm were validated. Specifically, it supports load balancing with neighboring nodes and preferential processing of important data. In addition, compared to the conventional absolute priority-based algorithm, the proposed algorithm shows performance improvement in terms of throughput, delay, and fairness for low-importance packets. Moreover, the performance comparison with other scheduling scheme ensures the excellent performance of the proposed scheme because it achieves the highest throughput and the lowest delay. These results verify that both the service differentiation and performance improvement can be achieved through an appropriate weight factor setting.

Author Contributions: W.L. and T.K. (Taejoon Kim) conceived and designed the experiments; W.L. and T.K. (Taehong Kim) performed the network simulation; W.L. and T.K. (Taehong Kim), and T.K. (Taejoon Kim) analyzed the data; T.K. (Taejoon Kim) acquired funding; W.L. and T.K. (Taejoon Kim) wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2020R1F1A3068305).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rajaraman, R. Topology control and routing in ad hoc networks. SIGACT News 2002, 33, 60. [CrossRef]
2. Sharmila, S.; Shanthi, T. A survey on wireless ad hoc network: Issues and implementation. In Proceedings of the 2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS), Pudukkottai, India, 24–26 February 2016; pp. 1–6.
3. Wu, K.; Harms, J. Multipath routing for mobile ad hoc networks. J. Commun. Netw. 2002, 4, 48–58. [CrossRef]
4. Fossa, C.F.; Macdonald, T.G. Internetworking tactical MANETs. In Proceedings of the 2010-MILCOM 2010 Military Communications Conference, San jose, CA, USA, 31 October–3 November 2010; pp. 611–616.
5. Bekmezci, I.; Alagöz, F. A New TDMA Based Sensor Network for Military Monitoring (MIL-MON). In Proceedings of the MILCOM 2005 IEEE Military Communications Conference, Atlantic City, NJ, USA, 17–20 October 2006; pp. 1–6.
6. Alemdar, H.; Ersoy, C. Wireless sensor networks for healthcare: A survey. Comput. Netw. 2010, 54, 2688–2710. [CrossRef]
7. Zhang, C.; Zhang, M.; Su, Y.; Wang, W. Smart home design based on ZigBee wireless sensor network. In Proceedings of the 7th International Conference on Communications and Networking in China, Kunming, China, 8–10 August 2012; pp. 463–466.
8. Saleh, N.; Kassem, A.; Haidar, A.M. Energy-Efficient Architecture for Wireless Sensor Networks in Healthcare Applications. IEEE Access 2018, 6, 6478–6486. [CrossRef]
9. Chao, H.-L.; Kuo, J.-C.; Liao, W. Fair scheduling with QoS support in ad hoc networks. In Proceedings of the 27th Annual IEEE Conference on Local Computer Networks 2002 Proceedings LCN 2002 LCN-02, Tampa, FL, USA, 6–8 November 2003; pp. 502–507.
10. Kim, H.; Min, S.-G. Priority-based QoS MAC protocol for wireless sensor networks. In Proceedings of the 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS), Rome, Italy, 23–29 May 2009; pp. 1–8.
11. Ye, W.; Heidemann, J.; Estrin, D. An energy-efficient MAC protocol for wireless sensor networks. In Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, New York, NY, USA, 23–27 June 2003; Volume 3, pp. 1567–1576.

12. Carlos-Mancilla, M.; Fapojuwo, A.; Lopez-Mellado, E.; Siller, M. An efficient reconfigurable ad-hoc algorithm for multi-sink wireless sensor networks. Int. J. Distrib. Sens. Netw. 2017, 13, 1550147717733390. [CrossRef]

13. Abu Salem, A.O.; Shudifat, N. Enhanced LEACH protocol for increasing a lifetime of WSNs. Pers. Ubiquitous Comput. 2019, 23, 901–907. [CrossRef]

14. Sakthy, S.S.; Bose, S. Dynamic Model Node Scheduling Algorithm Along with OBSP Technique to Schedule the Node in the Sensitive Cluster Region in the WSN. Wirel. Pers. Commun. 2020, 114, 1–15. [CrossRef]

15. Zareei, M.; Vargas-Rosasale, C.; Hernández, R.V.; Azpilicueta, E. Efficient Transmission Power Control for Energy-harvesting Cognitive Radio Sensor Network. In Proceedings of the 2019 IEEE 30th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops), Istanbul, Turkey, 8 September 2019; pp. 1–5.

16. Zareei, M.; Vargas-Rosasale, C.; Anisi, M.H.; Musavian, L.; Villalpando-Hernandez, R.; Goudarzi, S.; Mohamed, E.M. Enhancing the Performance of Energy Harvesting Sensor Networks for Environmental Monitoring Applications. Energies 2019, 12, 2794. [CrossRef]

17. Shi, H.; Prasad, R.V.; Onur, E.; Niemegeers, I.G.M.M. Fairness in Wireless Networks: Issues, Measures and Challenges. IEEE Commun. Surv. Tutor. 2013, 16, 5–24. [CrossRef]

18. Kelly, F. Charging and rate control for elastic traffic. Eur. Trans. Telecommun. 1997, 8, 33–37. [CrossRef]

19. Jain, R.; Chiu, D.-M.; Hawe, W. A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System; Digit. Equip. Corp. Tech. Rep-301; Eastern Research Laboratory, Digital Equipment Corporation: Hudson, MA, USA, 1984; pp. 1–38.

20. Elliott, R. A measure of fairness of service for scheduling algorithms in multiuser systems. In Proceedings of the IEEE CCECE2002 Canadian Conference on Electrical and Computer Engineering Conference Proceedings (Cat No 02CH37373) CCECE-02, Winnipeg, MB, Canada, 12–15 May 2003; Volume 3, pp. 1583–1588.

21. Rhee, I.; Warrier, A.; Min, J.; Xu, L. DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad Hoc Networks. IEEE Trans. Mob. Comput. 2009, 8, 1384–1396. [CrossRef]

22. Ramanathan, S. A unified framework and algorithm for (T/F/C)DMA channel assignment in wireless networks. In Proceedings of the Proceedings of INFOCOM ’97, Kobe, Japan, 7–11 April 2002; Volume 2, pp. 900–907.

23. Wang, Y.; Henning, I. A Deterministic Distributed TDMA Scheduling Algorithm for Wireless Sensor Networks. In Proceedings of the 2007 International Conference on Wireless Communications, Networking and Mobile Computing, Shanghai, China, 21–25 September 2007; pp. 2759–2762.

24. Vergados, D.J.; Amelina, N.; Jiang, Y.; Kraljevska, K.; Granichin, O. Toward Optimal Distributed Node Scheduling in a Multihop Wireless Network Through Local Voting. IEEE Trans. Wirel. Commun. 2018, 17, 400–414. [CrossRef]

25. Lutz, J.; Colbourn, C.J.; Syrotiuk, V.R. ATLAS: Adaptive Topology- and Load-Aware Scheduling. IEEE Trans. Mob. Comput. 2013, 13, 2255–2268. [CrossRef]

26. Vergados, D.J.; Sgora, A.; Vergados, D.D.; Vouyioukas, D.; Anagnostopoulos, I. Fair TDMA scheduling in wireless multihop networks. Telecommun. Syst. 2010, 50, 181–198. [CrossRef]

27. Li, Y.; Zhang, X.; Zeng, J.; Wan, Y.; Ma, F. A Distributed TDMA Scheduling Algorithm Based on Energy-Topology Factor in Internet of Things. IEEE Access 2017, 5, 10757–10768. [CrossRef]

28. Ambigavathi, M.; Sridharan, D.; M. A. Energy efficient and load balanced priority queue algorithm for Wireless Body Area Network. Futur. Gener. Comput. Syst. 2018, 88, 586–593. [CrossRef]

29. Karim, L.; Nasser, N.; Taleb, T.; Alqallaf, A. An efficient priority packet scheduling algorithm for Wireless Sensor Network. In Proceedings of the 2012 IEEE International Conference on Communications (ICC), Ottawa, ON, Canada, 10–15 June 2012; pp. 334–338.

30. Li, X.; Chen, N.; Zhu, C.; Pei, C. Improved Efficient Priority-and-Activity-Based QoS MAC Protocol. In Proceedings of the 2013 5th International Conference on Intelligent Networking and Collaborative Systems, Xi’an, China, 9–11 September 2013; pp. 315–318.
32. Slama, I.; Jouaber, B.; Zeghlache, D. A Free Collision and Distributed Slot Assignment Algorithm for Wireless Sensor Networks. In Proceedings of the IEEE GLOBECOM 2008 IEEE Global Telecommunications Conference, New Orleans, LO, USA, 30 November–4 December 2008; pp. 1–6.

33. Liu, Y.; Elhanany, I.; Qi, H. An energy-efficient QoS-aware media access control protocol for wireless sensor networks. In Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, Washington, DC, USA, 7 November 2005; pp. 189–191.

34. Yigitel, M.A.; Incel, O.D.; Ersoy, C. QoS-aware MAC protocols for wireless sensor networks: A survey. Comput. Netw. 2011, 55, 1982–2004. [CrossRef]

35. Github. Available online: https://github.com/djvergad/local_voting (accessed on 1 August 2017).