Research Article

An Energy Efficient Dynamic Address Encoding Scheme for Wireless Sensor Network

Liuxiao Zheng, Kewang Zhang, and Deyun Zhang

Department of Computer Science, Xi’an Jiaotong University, No. 28 West Xianning Road, Xi’an 710049, China

Correspondence should be addressed to Kewang Zhang; zhangkw@gmail.com

Received 13 December 2013; Accepted 17 February 2014; Published 19 March 2014

Academic Editor: Shancang Li

Copyright © 2014 Liuxiao Zheng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Wireless sensor network is drawing intensive research attention and being applied in various practical scenarios. The integration of WSN in industrial manufacturing process provides low cost industrial automation systems. The benefits of WSN include rapid deployment, self-organization, and being maintenance free. However, wireless sensor nodes are battery powered; replacing or recharging battery is usually impossible after deployment, which makes energy a scarce resource for WSN. Furthermore, in industrial applications, environment information is gathered and delivered in a timely manner, which makes traditional energy saving methods such as listen-sleeping duty approach less efficient. In this work, a novel dynamic address encoding scheme based on packet arrival rate is proposed; the scheme can reduce the protocol overhead by reducing MAC and network address size and thus saves energy needed for data transmission. The implementation of the scheme is fully distributed and needs little modification of the existing applications. Simulation results show the proposed scheme is efficient and robust on both the MAC layer and the network layer.

1. Introduction

Nowadays, in the competitive industrial marketplace, companies face growing demands in increasing the efficiency of manufacturing process to achieve financial goals. With the dynamic environment and variable market challenges, automation systems are needed to improve productivity [1]. Traditionally, automation systems deployed in industrial producing process are controlled by full-capacity computer systems connected by wired communication networks, which is usually expensive by the hardware itself, and also need high costs at deployment and the following maintenance [2]. With the technique advances in wireless sensor networks (WSN), in recent years, researchers suggest integrating WSN in industrial manufacturing process to provide low cost automation industrial systems, which is known as industrial wireless sensor network (IWSN) [3].

WSN is an emerging technology which has potential usage in virtually any application that is related to information gathering, such as environment monitoring, battlefield, medical treatment, animal control, disaster investigation, and industrial plants [4]. Specifically, in industrial applications, IWSN is used to collect parameters that are vital in manufacturing process, such as vibration, temperature, pressure, and magnetic field level [5]. WSN is formed by a number of low cost sensor nodes. Each sensor node usually consists of four subsystems: a sensor, a processor with limited capacity and few embedded RAM, a low power radio transmission unit, and a battery to be the power source. The sensor nodes are deployed in the area of interest and installed on equipment with no more configurations. WSN is formed between sensor nodes in ad hoc manner with preprogrammed routing protocols. The sensed data is passed by WSN to sink node with minimal process on each node. The sink node usually sits on the edge of the autonomous network, which further passes the data to the plant personnel. In this way, action can be taken whenever unexpected parameters are observed in the manufacturing process. The benefits of WSN include low cost, rapid deployment, self-organization, and being maintenance free. However, WSN is often deployed in harsh environment, which means it is difficult to replace or recharge batteries on sensor nodes. Thus, the life time of WSN depends on the battery life time of individual sensor node. Experiments show that wireless communication consumes most power on sensor node, while power consumption on data processing is significantly less. Therefore, power consumption must be
taken into account when designing WSN network protocol [6].

The radio subsystem on sensor node usually works on four modes: transmitting, receiving, idle listening, and sleeping. While the sensor node is transmitting or receiving data, the radio subsystem works at maximum power level and consumes large amount of battery power. Idle listening refers to the mode when the sensor node is not exchanging data with other nodes, but the radio subsystem is still powered on and sensing the channel for potential data exchange. On idle listening mode, the radio subsystem is able to consume less power. But the power consumption is still of the same order of magnitude as in receiving or transmitting mode. On sleeping mode, the radio subsystem is totally turned off. Most of the current proposed energy conservation schemes for WSN concentrate on optimizing idle listening mode [7, 8]. Such schemes try to lower battery drain by prolonging the sleep period of the radio subsystem, with which the idle period between two consecutive data exchanging processes is split into listening and sleeping cycles. The longer the sleeping period is in one cycle, the less the power consumption needed would be. Nonetheless, in industrial plants, the application usually requires real-time environment information, which means small end-to-end delay is required. Thus, in industrial applications, the length of the sleeping period can not be long, which in turn limits the efficiency of such schemes.

Power consumption of the radio subsystem on data exchanging totally depends on the packet length. As the packet payload data is usually compressed before transmission, protocol overhead can be optimized by making the packet header compact. Since packets in WSN are usually very small, reducing even a few bytes for each packet would save considerable power in the long run.

Schurgers et al. [9] studied the function of MAC address in WSN and proposed a variable length MAC address encoding scheme, with which traditional fixed length MAC addresses are replaced by variable length local addresses which are coded in the Huffman encoding. According to their study, the proposed scheme achieved a MAC layer overhead reduction by a factor of three. However, there are two issues not addressed in their work: firstly, addresses are acquired in the order of node activation, which may result in underoptimal situation when the traffic load is distributed in the area unevenly; secondly, the optimization was performed on MAC layer only, which means, in multihop cases, network scale unique node IDs are still needed for packet to be routed correctly to sink node and for sink node to distinguish packets from different source nodes, and these IDs may still contribute to large portion of the packet protocol overhead.

In this work, a novel dynamic address encoding scheme based on packet arrival rate is proposed by the inspiration of Schurgers’ work. Unlike the existing address encoding scheme that tries to totally replace traditional fixed size address encodings, our approach keeps the existing address architecture and translates the old address to variable length address on the fly when needed. There are three major improvements made in the work. First, addresses are acquired when needed at data exchanging. The address space is further reduced, which is only required to hold all the nodes currently transmitting rather than all active nodes. Next, an address modification mechanism is provided for sensor node to adaptively adjust its address size according to the arrival packet rate compared to others. This allows smaller address to be used on busier nodes, which optimizes the average address size in the whole network. Last, in multihop cases, the node ID in packet header needed for routing is translated to link ID maintained locally on each node along the route path. The link ID size is optimized according to the packet rate on the specified route path by the dynamic Huffman encoding algorithm. The proposed scheme is implemented in a totally distributed way that no central control is needed. It is also implemented in a compatible way as sublayers for MAC layer and network layer, respectively; thus, modifications are not needed for the existing protocols or applications.

2. Address Uniqueness and Dynamic Address Encoding Based on Packet Rate

In wireless networks, each node usually has addresses equipped on two layers: MAC layer and network layer. In a single spectrum network, MAC address is used to distinguish nodes from each other in the shared wireless channel, so a packet can be handled correctly on both the sending node and the receiving node. On the other hand, network address guarantees that a packet is correctly routed from the source node to the destination in multihop cases. Figure 1 shows a topology of a wireless network. A solid line between two nodes indicates they are within radio range of each other.

2.1. Uniqueness of MAC Address and Its Encoding. MAC addresses on nodes need to be unique to function properly. For every packet transmitted in the network, there must be at least one pair of MAC addresses, so the packet can reach the receiver properly and a confirmation message can be sent back to the sender. Traditionally, MAC addresses are encoded in fixed length, either at manufacture time or deployment time. Reference [9] proposed a dynamic MAC address encoding scheme based on the study of wireless network with both bidirectional and unidirectional communications. Since in most practical MAC protocols ARQ (automatic repeat request) [10] is needed for packet correctness, all communications in wireless networks can be considered bidirectional, and the uniqueness condition of MAC address from [9] is actually simple and can be given as Lemma 1.
Lemma 1. A valid MAC address assignment is that, for any two nodes in a network, address distinction must be guaranteed if the two nodes (1) are within radio range of each other or (2) have at least one common neighbor.

As it is shown in Figure 1, node J and node K are within range of each other, so they must have different MAC addresses. Nodes A, B, and C are not in the range of node E or F; however, since node D is their common neighbor, all six nodes must have different addresses. On the other hand, node G can reuse an address that is being used by node A, B, or C without any ambiguity, since MAC layer communication is always only one hop long. In the proposed scheme, MAC address size is reduced, since address length is only related to the node density with respect to the radio range. Contrastively, addresses configured at manufacture time would be typically 48 bits, and address determined at deployment time would have a length of \( \log_2 N \), where \( N \) is the total number of nodes in the whole network. However, because in the proposed scheme address is obtained at activation time, the addresses generated may be under optimal from the whole network's point of view, and further improvements can be applied.

WSN is deployed in the area of interest to gather information. To save energy on sensor node, it is usually configured to upload the parameter to the sink node only if the parameter is changing, which means, in any given time period that is of limited length, only a subset of the whole nodes would be transmitting. Also, in industrial applications, multiple environment parameters would be monitored, and different sensors generate different amount of data in the same time period, which makes MAC addresses on each node of different importance, in other words, of different weight. Consider the six nodes A to F in Figure 1; suppose, in some particular time period, packets transmitted on all nodes are counted as 500 on A, 100 on B, 10 on C, 610 on D, 0 on E, and 610 on F. Lemma 1 indicates that all six nodes should have unique addresses. We denote the total bits length in all packets that present MAC addresses to be \( L \); then \( L = \sum_i l_i n_i \), where \( l_i \) is the address length for node \( i \) and \( n_i \) is the packet count transmitted on node \( i \). For dynamic address encoding with the same length for these six addresses, each address will take 3 bits, and the total address length in all packets would be \( L = 3 \times (500 + 100 + 10 + 610 + 610) = 5490 \) bits. However, because each node transmits different amount of packets, which makes each address of different importance in all packets. The more the packets transmitted by a node are, the more important its address becomes. Because node E transmitted no packet, its address should not be encoded in the time period. Thus, the optimal address length can be determined by a Huffman tree generated based on the packet count. The Huffman tree generated for this example is shown in Figure 2, with weight being the packet count which the node has transmitted.

The final encoding lengths for addresses are 2 bits for A, D, and F and 3 bits for B and C. So the total address length would be \( L = 2 \times 500 + 3 \times 100 + 3 \times 10 + 2 \times 610 + 2 \times 610 = 5770 \) bits, which results in more than 30% reduction compared to the former scheme. A comparison between different address encoding schemes for this example is given in Table 1.

Although little difference seems to be shown between the last three schemes, the second scheme will not scale well in larger networks. In a network with 10000 nodes, the scheme needs at least 14-bit-long address, while the address length with the latter two will not change much if the node density stays the same.

Dynamic address encoding based on packet arrival rate is supposed to have better efficiency when node density is high and traffic is distributed unevenly. This can be explained by information theory [11]. Denote the address found in one packet transmitted by node in the specified area to be a discrete random variable \( X \); then the probability \( P_i \) is defined as

\[
P_i = P \{ X = i \} = P \{ \text{address of node } i \text{ found on the packet} \}
\]

(1) \[
= \frac{\text{packets count node } i \text{ transmitted}}{\text{packets count all nodes transmitted}}
\]

The information content of the address on one node is \( I_i = -\log_2 P_i \), which means if the address appears more in packets, it will carry less information (smaller information content) and should be encoded with shorter code. And the entropy of node address in this area is given by

\[
H(X) = E(I(X)) = \sum_i P_i \cdot I_i = -\sum_i P_i \cdot \log_2 P_i.
\]

(2)

This is also the average MAC address length in the area. Since entropy indicates the unpredictability of information, if the packet arrival rate is distributed unevenly, then the addresses inspected on packets are less unpredictable, which result in smaller entropy of node address and thus shorter average address size.

2.2. Uniqueness of Network Address and Its Encoding. It is enough for MAC addresses to be unique locally only in a few hops, but this does not hold true for network addresses. In multihop wireless networks, the destination network address in packet header is used to identify the final destination node.
Table 1: Comparison of different address encoding schemes.

| Scheme name            | Average size | Implementation complexity | Scalability |
|------------------------|--------------|---------------------------|-------------|
| Global unique          | 48 bits (typical) | Configuration-free         | Good        |
| Network unique         | 4 bits       | Easy                       | Bad         |
| Dynamic                | 3 bits       | Complex                    | Good        |
| Dynamic on packet rate | 2.06 bits    | Complex                    | Good        |

And the source network address is for the final destination to identify the original source node. As in the example of Figure 1, consider a route path from nodes A to K, with the intermediate nodes being D, F, G, and J. The presence of address of K in the packet ensures each intermediate node handles the packet correctly, so even if the packet is delivered from node D to F, the receiver will pass it on to node G for its final destination node K other than deal with it any other way. Also, the presence of source address informs node K that the packet is actually from node A, even if it is delivered from node J. Because both the source node and the destination node could be any node in the whole network, the network address size must be long enough to hold all the nodes in the network. For example, with a network of 10000 nodes, the network address should be at least 14 bits long, and for each packet the network address pair will cause a 28-bit overhead.

The function of network addresses can also be explained as end-to-end links on network layer. For any stable routing algorithm, given a source and destination nodes pair, it should be able to produce a unique route path for packet delivery. So any route path in the network can be uniquely represented by the source and destination pair, with the route path being viewed as a virtual link between the two nodes and the addresses in packets considered the link ID. Since packet delivery could happen between any two nodes in the network, the size of this link ID should be large enough to present all possible links in the whole network. Also, in many networks, upload and download may take different paths (e.g., Token Ring [12]); thus, the link should be considered directional. From a little mathematics knowledge, we know that, in a network with N nodes, the possible directional link count is \(N \times (N - 1)\), so the link ID size \(L\) would be at least \(L = \log_2 N(N - 1)\) bits long. If \(N\) is big, we have \(L = 2\log_2 N\), which is twice the size of the network address.

The possible link count is tremendously big in large networks in theory. Luckily, the count of the activated ones would be smaller than, actually not even close to, the whole population in practical environments. In real world networks, each node would be exchanging information with a very limited amount of other nodes; thus, only a little portion of the possible links would be activated, which means the link ID can be much smaller than the theoretical one. Actually, if a link can be identified on each hop along the route path, then packets of the link can be delivered correctly to the destination, which means, with links distributed sparsely in the network, spatial reuse could be further applied to reduce the link ID size. An example is illustrated in Figure 3.

Rewriting the route path names by their source and destination node labels, we get AH, HA, BH, HB, CH, HC, CI, and IC. Although the possible count of links is 72 which needs a 7-bit-long link ID to represent, the activated ones need only a 3-bit ID to be distinguished from each other on the whole network scope. Furthermore, because the 8 route paths only partially overlap with each other, each link can be identified on every node by a local link ID with respect to the node it passes by. Consider the nodes along the route path AH; the link count on each node is 1 for A, 4 for D, 6 for F, and 3 for H (note that incoming links which have reached their destination do not need to be counted in). Then the link ID length on each node would be 1 bit on A, 2 on D, 3 on F, and 2 on H. Assume the link AH is marked as link 0 on every node along the path; the packet can be delivered as is shown in Figure 3, which is described as follows:

(i) node A sends the packet to D by marking it with link 0 on A, which is denoted by [A: 0];
(ii) node D receives the packet and translates [A: 0] to the full link ID as AH;
(iii) node D maps packet that belongs to link AH to link 0 locally and sends it to node F as [D: 0];
(iv) node F receives [D: 0], translates it, and passes it on to node H with [F: 0] with the same process as it is done by node D in 2 and 3;
(v) the packet marked as [F: 0] reached the destination node H and is being processed as a packet that belongs to AH.

As is illustrated in this process, on each hop, the local link ID with respect to the MAC address of the last node is translated to the full link ID (the source-destination network address pair) and encoded again with the local link ID with respect to the MAC address of the current node and passed
on. Because the MAC address does not bring any ambiguity as long as it follows Lemma 1 on each node and local link ID is unique, the translation on each hop will get a global unique link ID successfully, which guarantees the packet is delivered to the correct destination. The translation is accomplished with the help of a link map table (LMT). An example of LMT on node D is shown in Table 2.

The PR (packet rate) entry in LMT is discussed later. The LMTs show that this approach actually trades memory space and CPU time off transmission efficiency. Because data transmission needs more energy, this behavior is intended.

By employing the local link ID approach, we could achieve a much shorter network layer identifier than the original network addresses. The bigger a network is, the more efficiency this approach provides. The link ID could further be optimized by packet arrival rate to each link. As it is discussed for MAC address encoding, that an address appears more in packets would carry less information by itself. If more packets are delivered along a link, the link should have a shorter link ID compared to others. The situation can be explained by entropy in information theory. Since the explanation is similar to the one with MAC addresses, it is not repeated here.

In WSN, attribute based routing could be applied to eliminate the needs for destination address on network layer. However, in IWSN, when an unexpected event happens, the application usually needs to get detailed information about the specific plant or even machine where it happens, which means the use of source address is inevitable. Thus, the methodology presented here is still applicable. A Huffman link ID encoding scheme based on packet arrival rates is implemented in the next section.

3. Implementation of Dynamic Address Encoding Based on Packet Rate

In theory, addresses generated by the Huffman tree based on packet rate are the most efficient. However, in practical environments, implementation of such schemes must be chosen to compromise among efficiency, control complexity, and algorithm stability. In this section, an efficient address encoding scheme for both the MAC layer and network layer is designed and implemented based on the principles from the last section. To avoid modifications on the existing applications and protocols as much as possible, the address encoding process is implemented as two sublayers which perform address translation (AT) for MAC (MAC-AT) and network (NET-AT) layer, respectively. In the design, traditional network-wide node IDs are assumed to be equipped on nodes at either deployment time or manufacture time. Figure 4 shows the system architecture.

3.1. Implementation of MAC-AT. MAC-AT is designed as a sublayer under MAC layer. It encodes MAC addresses on outgoing packets with local addresses before it goes to physical layer on the fly and translates back local addresses on incoming packets to MAC addresses before MAC layer processing. The algorithm is fully distributed and consists of three major functions: (1) address assignment; (2) conflict resolution; and (3) address adjustment based on packet rate.

3.1.1. Address Assignment. Theoretically, the Huffman tree generation algorithm produces the most efficient address encoding. Nevertheless, it is not feasible to implement such algorithm on MAC layer in WSN for three reasons. First, the Huffman tree can not be easily maintained in a fully distributed way. To the best of the authors’ knowledge, there is no dynamic parallel algorithm to generate and maintain a Huffman tree efficiently. Furthermore, the system needs to be robust that it recovers fast upon node failures, which makes the approach that uses one special node to maintain the tree also unacceptable. Next, the wireless media are unreliable and error prone; plus a listen-sleeping duty cycle may be applied on MAC layer to save node energy, which makes it difficult for node to get full knowledge of all the nodes around. Without the full knowledge of the network, the Huffman tree algorithm may produce inefficient address code. Last, the algorithm should be stable in short time periods; if not, frequently exchanged control packets may cause more protocol overhead that neutralize the effect of short addresses by itself. Consequently, the addresses are defined as variable length prefix-free code on four different zones, as is shown in Table 3.

The zones are defined with addresses in each zone twice the length of those in the last zone but have much more addresses to allocate. The address assignment process works as follows.

(i) After a node is activated, it overhears packets transfer nearby and generates an address allocation and usage table (AAUT). The address usage statistic does not need to be accurate. So if there is a listen-sleeping duty applied on the MAC layer, it just needs to overhear packets at the active listening periods. In the example
of Figure 1, the AAUT on node E may look like what is shown in Table 4.

For each data packet it hears, the packet rate in the AAUT is updated for the corresponding address. If a control packet is heard to change the address, which will be described later, the AAUT is also updated with the new address mapping information. In case a node is activated late compared to others that it misses control packets, the actual node ID in the table is optional.

(ii) Before a node is able to send out packets, it needs to allocate an address as small as possible from the address zones. The allocation is accomplished by checking the AAUT for an address that is not in use. The address it picked should be confirmed before actually being used.

(iii) The address picked by the node is confirmed by sending out an ADDR_REQ packet to the intended receiver for data packets. On receiving the ADDR_REQ packet, the receiver first waits for a small time period and then sends back an ADDR_RES packet. Because hidden terminals may cause some allocations recognizable only by the receiver, it could approve or deny the allocation in the ADDR_RES packet. If the allocation is denied, the receiver can also embed a hint for further allocation in the ADDR_RES packet, which is the smallest address available for allocation according to its own AAUT. If the allocation is considered to be OK by the receiver, this request-approve handshake should be repeated ADDR_CONFIRM_COUNT times to guarantee that no neighbor of the sender or receiver picked the same address. The bigger the ADDR_CONFIRM_COUNT is, the less possible it is for addresses to conflict, but more latency would be caused on the first data transmission.

(iv) If any node hears the ADDR_REQ packet finds a conflict, it should deny the allocation by sending an ADDR_RES, which should also embed a hint for further allocation. If any node hears the ADDR_RES that tries to approve the allocation but not the ADDR_REQ packet that finds a conflict, it should deny the allocation by sending an ADDR_RES to the receiver. In this case, the receiver would deny the allocation in the following request-approve handshake.

(v) If the allocation is denied, the node goes back to step c; repack and address according to the hint in the deny packet and proceed. An address allocation is successful if it is not denied in all request-approve handshakes, and the data transmission can begin.

### 3.1.2. Conflict Resolution

Even the address assignment process tries to eliminate conflicts; due to the broadcast nature of wireless networks, it cannot be totally avoided. So there must be a way to resolve conflicts when that happens.

In the normal data exchanging, all addresses are encoded as small addresses, and there is no extra information about the original address; thus, if a conflict happens, it may only be discovered by higher layer or, even worse, bring confusion to higher layer. In this work, we develop a methodology to discover address conflict by CRC in the MAC packet. Namely, in the packets transmitted out, although the MAC addresses are replaced by encoded small addresses, the CRC will be kept as the original one. Since the CRC is checked after the MAC-AT translated address back to the full MAC addresses, which means packets submitted to MAC layer are fully structured MAC packets, the CRC check would fail if a packet is received erroneously because of address conflict. To prevent false alarms, we oblige a threshold ERROR_CONFLICT_COUNT on the error count before initiating an address conflict resolution process.

When a node makes sure that there are address conflicts, it starts a conflict resolution process, by sending out a CONFLICT_RESOLVE control packet to the node which continuously sends out packets with errors. This packet contains information of the small address in question, its usage statistics on this node, and the full address the node thinks it maps to. Upon receiving the CONFLICT_RESOLVE packet, the receiver checks whether an address conflict actually happens. If it does, then the node which is using the address less frequently gives up the address and starts the address assignment process as described in the last section. If they both have the same usage statistics, the node that found the conflict gives up the address.

### 3.1.3. Address Adjustment Based on Packet Rate

Address is assigned before the node transmits the first data packet, so the nodes that start transmissions earlier may have shorter addresses. As nodes that have longer addresses may transmit

| Zone | Binary code | Address space | Code length |
|------|-------------|---------------|-------------|
| S    | 0xxx        | 8             | 4 bits (3 bits) |
| M    | 10xx xxxx   | 64            | 8 bits (6 bits) |
| L    | 110x xxxx xxxx xxxx | 8192 | 16 bits (13 bits) |
| XL   | 1110 xxxx xxxx xxxx xxxx xxxx xxxx xxxx | 268,435,456 | 32 bits (28 bits) |

Table 3: MAC address defined for dynamic encoding.

| Node | Address | PR |
|------|---------|----|
| N/A  | 0001    | 100 |
| B    | 0010    | 1000 |
| N/A  | 0011    | 50  |
| D    | 0100    | 500  |

Table 4: AAUT on node E.
more packets afterwards, this may be under optimal in the long run. Thus, addresses should be adjusted by the packet arrival rate. This is achieved with the help of AAUT that was used in the address assignment process.

After the address assignment is accomplished, the node keeps the AAUT and continues to update it on any overheard packets. The packet rate of the node itself is also tracked by inserting the address allocation information in the AAUT after the address assignment and updating it for each packet it sends out or receives. If the address allocation was done in section M, L, or XL, the node checks whether it can get a boost on each packet rate update for itself. Namely, if the node is in section M, it compares its packet rate with the smallest packet rate in section S. If it is in section L, it compares with the smallest one in section M and so on. When the node finds another one which has less packet rate but occupies a smaller address, it could initiate an address exchanging process. For the address exchange, the node sends out an ADDR_EXCHANGE_REQ to the receiver, with its own packet rate embedded. Because the packet rate of the other node may be inaccurate, the comparison is performed with a SCALE_FACTOR. The receiver approves or denies the exchange request by an ADDR_EXCHANGE_RES packet. If the request is to be denied, the actual packet rate should be embedded in the response. Upon receiving the response which denies the exchange, the sender should update the packet rate in the AAUT of this node and adjust its SCALE_FACTOR value accordingly. The pseudo code of the address exchange process is shown in Algorithm 1.

3.2. Implementation of NET-AT. NET-AT works similarly to MAC-AT, except that it is a sublayer between the MAC layer and the network layer. NET-AT encodes network addresses in packets to local link IDs as they leave network layer to MAC layer and translates link IDs back to network addresses on receiving them from MAC layer. Since link IDs are generated with respect to the MAC address, maintained locally only on this node, and used only on the node that maintains it and the node on the next hop (recall that links are directional), a dynamic Huffman tree encoding algorithm is adopted to generate them; the detailed algorithm is described as follows.

(i) When a node wants to send packets to another node which is far away, it needs to create a multihop virtual link first. The link is created by a LINK_CREATE control packet sent out by the source node, which contains the full source address and the destination address.

(ii) On creating LINK_CREATE packet, the source node inserts a new record of the new link to LMT with PR set to 0. A new link ID is allocated by adding one to the biggest current ID and saved to the OUT entry. Because it is the source node, the IN entry is set to N/A. The newly allocated link ID is set in LINK_CREATE packet, and it is sent out to the next hop.

(iii) For each LINK_CREATE received by intermediate node, it creates a new link record in LMT. The node first allocates a new local link ID in the same way the source nodes do and saves it to OUT in the record; the rest of the information needed by the record is extracted from the LINK_CREATE packet. On the destination node, the OUT entry in the record is set to N/A, and other entries are filled the same way as the other nodes. PR in the record is also set to 0 for each node.

(iv) After the link is created and a link ID is allocated, the NET-AT sublayer is brought up to work. It translates packets between network layer and MAC layer to achieve minimal size network addresses. For each packet it processes, the NET-AT updates its PR entry of the record of the link in the LMT, which the packet belongs to. PR is used as the weight value to apply the dynamic Huffman encoding for producing optimal link IDs.

(v) If the packet rate updating results in a link ID change from the Huffman tree, the link is marked as changed locally. When the next packet of the link arrives, before transmitting the packet, the NET-AT sends out an UPDATE_LINK control packet to update LMT information on the next hop of this link. To minimize the unnecessary protocol overhead, if no packet arrives at the changed link, the link information is actually not updated.

To simplify the manipulation of the Huffman tree, the change of PR value is normalized to be increased or decreased by one each time. This is achieved by applying a PR_CHANGE_THRESHOLD on calculating PR. Each time when the PR value changes, the increment or decrement value is compared with PR_CHANGE_THRESHOLD. If it is bigger, then the node weight is changed by one; if it is smaller, the statistic continues.

Every node has a Huffman tree to maintain link IDs. The Huffman tree must meet two principles. (1) On the same level of the tree, any node must have a weight value smaller than or equal to that of the node on its right. (2) If a node has no right node, it must have a weight value smaller than or equal to the left-most node one level up.

Every tree contains a root node and an NYT (not yet transmitted; we are just following some naming convention) node, where the NYT node always has 0 weight. Node labels are full IDs (source and destination addresses combination) of the links. On a node without a link the initial tree has only one node, which is NYT, also the root. For every PR change event the NET-AT generates, the tree is first checked to see if it already contains the link ID. If it does not, the NYT node spawns two new nodes. The node to its right is a new node containing the link ID, and the new left node is a new NYT node. If the link ID already exists in the tree, the weight is simply updated for the node. If after the update the weight of node does not meet the two principles above, the node is swapped with its neighbor node to remedy the tree. Note that a left-most node has a neighbor one level down and a right-most node has a neighbor one level up. To do this, before updating the node weight, the node should be swapped with the left-most neighbor (including down level) with the same
**Algorithm 1: Algorithm for address exchange.**

```plaintext
for Each Packet_Rate change on this node do
    if Packet_Rate > Other_Packet_Rate[i] * SCALE_FACTOR[i] then
        Send ADDR_EXCHANGE_REQ to node i;
        Receive ADDR_EXCHANGE_RES;
        if The request is approved then
            Record the address exchange;
            Use new address in the following transmissions;
        else
            Update Other_Packet_Rate[i] with information from ADDR_EXCHANGE_RES;
            SCALE_FACTOR[i] = Old_SCALE_FACTOR[i]/Old_Packet_Rate[i] * Other_Packet_Rate[i];
        end if
    end if
end for
```

**Figure 5: Generation of link ID by the Huffman tree.**

- **(a)** NYT \( W=0 \)
- **(b)** NYT \( W=0 \), L1 \( W=1 \)
- **(c)** NYT \( W=0 \), L1 \( W=2 \)
- **(d)** NYT \( W=0 \), L1 \( W=2 \), L2 \( W=1 \)
- **(e)** W = 3
- **(f)** W = 4
- **(g)** W = 5

**4. Performance Evaluation**

The performance of the proposed scheme was evaluated under the NS-3 simulator by simulation. In the experiments performed, \( N \) sensor nodes were distributed uniformly in the square field with dimension of \( L \) by \( L \). All sensor nodes in weight, if weight is to be decreased, or with the right-most neighbor (including up level) with the same weight, if weight is to be increased. Finally the update can be performed on the node.

On each update, the value must also be updated for every parent above, so every internal node has a weight value to be the sum of the two sibling nodes. Also, on each update, the tree must follow the two principles above, so node swap is performed if necessary for every node above. Thus, to avoid unnecessary energy consumption on maintaining the tree, PR_CHANGE_THRESHOLD should be reasonably big.

Assume there are four links on one node and the statistic gets five events: link 1 created, link 1 PR increased by 1, link 2 created, link 3 created, and link 4 created; the update of the Huffman tree on the node can be illustrated by Figure 5.

After the last step, the binary link IDs are 0 for link 1, 10 for link 2, 111 for link 3, and 1101 for link 4.
the network are the same. The radio equipped on the node has a range of $R$, and the MAC layer is configured to work as CSMA with ACK, like IEEE 802.11 DCF [13]. Thus, every node has a neighbor count of $\rho = (N/L^2)\pi R^2$. Because MAC address is dynamically generated with respect to the packets rates of all neighbors, the performance of MAC-AT is mainly evaluated with respect to $\rho$. However, the performance of NET-AT is evaluated with respect to both $\rho$ and $N$, because routing algorithm works network-wide.

The first set of experiments was carried out to evaluate the performance of MAC-AT. The network density is set to 10 to 50 with step 10, and the network scale is set to 1000 to 100000 with step 1000; with each combination of $\rho$ and $N$, the experiments were performed and the mean address size was measured. The result is shown in Figure 6.

To compare the efficiency of different address assignment methods, six sets of experiments were carried out. For one set of experiments, addresses of sensor nodes were assigned at deployment time, which is performed before the system is brought up to run. In the other five sets of experiments, addresses were assigned using the approach proposed in this work; with each set a different node density value is configured. For each set of experiments, the mean address size is measured for the sensor network with node count from 1000 to 10000. Figure 6 shows that, in fixed size address assignment approach, the address size increases as network scale increases. However, the dynamic address size with dynamic assignment stays the same as long as node density does not change. When node density increases, the address size increases as well but is still smaller than those in fixed size cases. From the result we can get that the performance of dynamic encoding does not depend on the network scale, and it is only related to the network density.

The performance of NET-AT is also evaluated by experiments. In the experiments, the performance of NET-AT was evaluated against the change of network scale. The network density is fixed to 10, and the network scale is fixed to 10000; 90% of the nodes in the network were set to have a constant packet arrival rate of 1 packet/second. And 10% of the nodes have variable packets rate increasing from 1 packet/second to 100 packets/second. The change was performed every 10 s, with a rate increment of 1 packet per second. The $PR,CHANGE,THRESHOLD$ was set to a fixed value of 1 packet per second, so the experiment lasts for 1000 seconds. The result was measured as the mean link ID length of the 10% of nodes with variable packet arrival rate. Figure 7 shows the final result.

The results were compared between fixed length address assignment approach and dynamic assignment approach. As is shown in Figure 7, for fixed length address assignment, although traffic is distributed unevenly among nodes in the network, address length will not change. However, as traffic on the 10% of nodes with variable packet rate increases, the weight of the network address corresponds to the node increases, which result in a change of the Huffman tree maintained on the node and also a shorter address generated by the tree. Thus, as the experiment goes on, the address length drops with the increased packet rate on such nodes. The result in Figure 7 shows that the proposed approach performs well for network with unevenly distributed traffic.

5. Conclusion

In this paper, a novel dynamic address encoding scheme is proposed for IWSN. The scheme optimizes MAC addresses and network addresses based on packet arrival rate. Unlike the existing address encoding schemes that try to totally replace traditional fixed size address encodings, our approach keeps the existing address architecture and translates the old address to variable length address on the fly when needed. The proposed scheme is implemented in a totally distributed way so no central control is needed. It is also implemented in a compatible way as sublayers for MAC layer and network layer, respectively, so modifications are not needed for the existing
protocols or applications. The simulation results show the proposed scheme works with great efficiency.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is sponsored by the National Science Foundation of China (under contract no. 61103242).

References

[1] V. C. Gungor and F. C. Lambert, “A survey on communication networks for electric system automation,” Computer Networks, vol. 50, no. 7, pp. 877–897, 2006.
[2] U.S. Department of Energy, Assessment Study on Sensors and Automation in the Industries of the Future, Office of Energy and Renewable Energy Reports, 2004.
[3] T. Federico, Wireless Sensor Networks.
[4] J. Vick, B. Mukherjee, and D. Ghosal, “Wireless sensor network survey,” Computer Networks, vol. 52, no. 12, pp. 2292–2330, 2008.
[5] V. C. Gungor and G. P. Hancke, “Industrial wireless sensor networks: challenges, design principles, and technical approaches,” IEEE Transactions on Industrial Electronics, vol. 56, no. 10, pp. 4258–4265, 2009.
[6] G. Anastasi, M. Conti, M. Di Franceso, and A. Passarella, “Energy conservation in wireless sensor networks: a survey,” Ad Hoc Networks, vol. 7, no. 3, pp. 537–568, 2009.
[7] J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor networks,” in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys ’04), pp. 95–107, November 2004.
[8] W. Ye, F. Silva, and J. Heidemann, “Ultra-low duty cycle MAC with scheduled channel polling,” in Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys ’06), pp. 321–334, November 2006.
[9] C. Schurgers, G. Kulkarni, and M. B. Srivastava, “Distributed on-demand address assignment in wireless sensor networks,” IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 10, pp. 1056–1065, 2002.
[10] A. K. El-Hakeem, Automatic Repeat Request, Wiley Encyclopedia of Electrical and Electronics Engineering, 1999.
[11] C. M. Thomas and J. A. Thomas, Elements of Information Theory, John Wiley & Sons, 2012.
[12] M. Ergen, D. Lee, R. Sengupta, and P. Varaiya, “WTRP-wireless token ring protocol,” IEEE Transactions on Vehicular Technology, vol. 53, no. 6, pp. 1863–1881, 2004.
[13] IEEE802.11b/D3.0, Wireless LAN Medium Access Control(MAC) and Physical (PHY) Layer Specication: High Speed Physical Layer Extensions in the 2.4 GHz Band, 1999.
