Hindawi Publishing Corporation
Journal of Sensors
Volume 2013, Article ID 142078, 14 pages
http://dx.doi.org/10.1155/2013/142078

Research Article

Energy-Efficient Adaptive Geosource Multicast Routing for Wireless Sensor Networks

Daehee Kim,[1] Sejun Song,[2] and Baek-Young Choi[1]

[1] *Department of Computer Science Electrical Engineering, School of Computing and Engineering, University of Missouri, Kansas City, MO 64110, USA*
[2] *Department of Engineering Technology Industrial Distribution, Dwight Look College of Engineering, Texas A&M University, College Station, TX 77843, USA*

Correspondence should be addressed to Baek-Young Choi; choiby@umkc.edu

Received 3 September 2012; Accepted 19 February 2013

Academic Editor: Eugenio Martinelli

Copyright © 2013 Daehee Kim 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.

We propose an energy-efficient adaptive geosource multicast routing (EAGER) for WSNs. It addresses the energy and scalability issues of previous location based stateless multicast protocols in WSNs. EAGER is a novel stateless multicast protocol that optimizes location-based and source-based multicast approaches in various ways. First, it uses the receiver's geographic location information to save the cost of building a multicast tree. The information can be obtained during the receiver's membership establishment stage without flooding. Second, it reduces packet overhead, and in turn, energy usage by encoding with a small sized node ID instead of potentially large bytes of location information and by dynamically using branch geographic information for common source routing path segments. Third, it decreases computation overhead at each forwarding node by determining the multicast routing paths at a multicast node (or rendezvous point (RP)). Our extensive simulation results validate that EAGER outperforms existing stateless multicast protocols in computation time, packet overhead, and energy consumption while maintaining the advantages of stateless protocols.

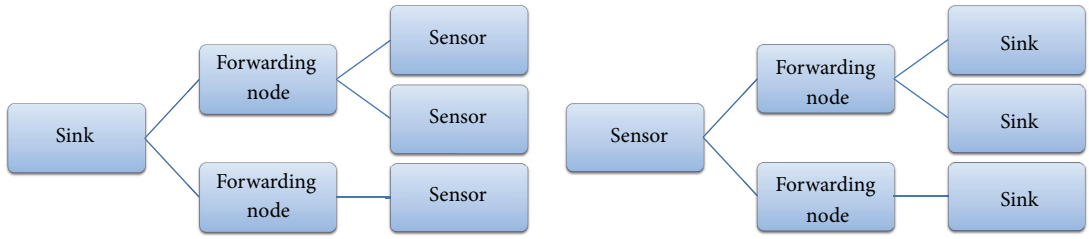
1. Introduction

Large self-organizing wireless sensor networks (WSNs) consist of a great number of sensor nodes with wireless communication and sensing capabilities. The sensor nodes can be deployed randomly close to or inside of the terrain of interest to provide cooperative wireless ad hoc network services. The sensed data and control messages are exchanged between sensor nodes and the control (sink) nodes via a multihop routing protocol. Potential applications of WSNs are numerous, and include environmental monitoring, industrial control and monitoring, and military surveillance to name a few.

Many sensor nodes have been commercially developed for various purposes (e.g., [1–6]). However, the sensor nodes have limitations such as a low capacity processor, small memory, and tiny storage as shown in Table 1, in addition to battery constraints.

Meanwhile, many WSN applications such as mission assignments, configuration updates, and phenomenon reports require one-to-many communications in nature, either from a sensor node to sink nodes or a sink node to sensor nodes. Multicast routing is an important routing service for such applications, as it provides an efficient means of distributing data to multiple recipients compared to multiple unicasts, using in-network replication. Considering both limitations of sensor nodes and the significance of multicast routing, it is critical to deliver multicast packets with a low overhead of resources such as energy, processing, memory, and storage in sensor nodes.

Multicast protocols can be classified into three categories including tree-based, source-based, and location-based multicast protocols. The tree-based multicast protocols [7–12] deliver a multicast packet relying on forwarding states maintained at nodes in a path. Its major drawbacks are control information flooding and storage for forwarding table establishment and maintenance, which produce a lot of overhead in WSNs. The source-based multicast protocols [13, 14] make a path tree at a source, and a multicast



(a) A Sink to Sensors (b) A Sensor to Sinks

Figure 1: wireless sensor network multicast application types.

packet encoded with the path tree information is propagated, requiring no states in WSN nodes. However, as the network size expands, it accrues packet size elongation due to the increased path tree information, which in turn, causes a sharp increase in the overhead of CPU processing and energy consumption. In location-based multicast protocols [15, 16], a multicast packet contains the location information of the destination nodes. It is stateless, like source-based routing, but the packet header size is proportional to the number of destinations and does not increase with the network size. However, it requires computation at every forwarding node in a path while looking for the next forwarding node, resulting in excessive processing of CPU and energy consumption.

In this paper, we present an energy-efficient adaptive geosource multicast Routing (EAGER) protocol for WSNs. EAGER is a novel stateless multicast protocol to optimize the previous location-based and source-based multicast approaches in various ways. The unique contributions of the proposed protocol are as follows: (1) it builds a common path multicast tree during the group membership establishment period. This on-demand approach reduces the location flooding overhead of the network topology maintenance on each node; (2) it decreases the computational overhead of each forwarding node such as the forwarding decision and packet decoding/encoding, with simple serialized path information; (3) it reduces the packet encoding overhead by adaptively using geographic unicast and source multicast. Geographic unicast is more efficient for long nonbranching path segments, and source multicast is desirable with branching path segments; (4) it further decreases the packet header size by using the multicast packet with a small node ID instead of potentially large position information; (5) overall, the reduced computational overhead, encoding overhead, and packet header size enable EAGER to consume less energy than location-based or source-based multicast protocols.

The rest of the paper is organized as follows. Section 2 provides a survey of existing multicast protocols for WSNs. Section 3 describes the proposed EAGER scheme and its algorithms. Section 4 presents extensive evaluations of EAGER and its comparison with location-based or source-based multicast protocols in various scenarios. Section 5 offers the conclusions and future work.

2. Related Work

A large number of studies (e.g., [17–19]) have been conducted in the field of multicast applications in WSNs. Those applications operate in a multicast communication mode either from a sensor node to sink nodes or a sink node to sensor nodes, as depicted in Figure 1(a). Configuration updates [17] or mission assignments [18] are the examples of a sensor to sinks multicast. Sensor to multiple sinks [19] scenarios are common for monitoring applications that require reliability.

Table 1: Capacities of sensor nodes.

Categories	Name	CPU (MHz)	Memory	Storage
Sensor nodes	Egs	96	52 KB	256 KB
	Micaz	8	4 KB	128 KB
	TelosB	8	10 KB	48 KB
	Tmote sky	8	10 KB	48 KB
	IMote2	400	32 MB	32 MB
	SunSPOT	400	1 MB	8 MB

Multicast protocols in wireless networks can be classified into three categories including tree-based, source-based, and location-based multicast protocols. The examples of tree-based multicast algorithms include adaptive demand-driven multicast routing protocol (ADMR) [8], on-demand multicast routing protocol (ODMRP) [9], multicast ad hoc on demand distance vector routing (MAODV) [10], progressively adapted sub-tree in dynamic meshs (PAST-DM) [7], ad hoc multicast routing protocol utilizing Increasing id-numberS (AMRIS) [11], and ad hoc multicast routing protocol (AMRoute) [12]. They have been developed for traditional wireless ad hoc networks, and have evolved in support of WSNs. However, those traditional multicast routing techniques are designed as control centric approaches, and focus on solving the mobility issues under the assumption of enough processing and local storage capacity on each node. They maintain a forwarding table on each node through a multicast routing tree for each group. The distributed group forwarding states should be updated via periodic control flooding messages, consuming significant energy. Due to the resource limitations on sensor nodes, they cannot be directly used for WSNs.

A Source-based multicast protocol, such as dynamic source multicast (DSM), has been proposed to perform centralized membership management on a multicast root instead of distributed state maintenance. A root or a multicast source builds a multicast tree using the locally maintained network topology information and encodes the tree information into the packet header. Forwarding nodes relay the packet according to the tree path information carried in the packet

Table 2: Classification of WSN multicast protocols.

	Tree-based routing	Location-based routing	Source-based routing
Strengths	Small data packet overhead	No distributed routing control overhead, less path encoding and decoding overhead than source-based	No distributed routing control overhead, relatively smaller forwarding computation than location-based
Weaknesses	Stateful, large control overhead (flooding), large memory/storage	Large packet size (location information of destinations), large forwarding computation	Large packet size (path), path encoding and decoding overhead
Examples	MAODV, ADMR, ODMRP, AM-Route, AMRIS	PBM, LGT, GMR, DDM, RDG	DSM

header. Although the distributed stateless multicast protocols are typically considered to be better for resource constrained WSNs than stateful distributed tree-based protocols, for large-scale networks, those stateless multicast protocols suffer substantial energy consumption due to the packet encoding and decoding operations at a source node and forwarding nodes, respectively.

Several location-based multicast protocols such as Location Guided Tree construction algorithms (LGT) [15], differential destination route driven gossip (RDG) [20], differential destination multicast (DDM) [21], geographic multicast routing (GMR) [16], and position-based multicast routing protocol (PBM) [14] have been proposed. These protocols compute the next forwarding node at each node on the path, based on location information of destinations rather than path information. Therefore, these have less path encoding and decoding overhead than source-based multicast protocols. However, these protocols still require large packet sizes for the destinations' location information and large forwarding computation at all the path nodes. In order to address the issue of scalability for a large number of destinations, hierarchical rendezvous point multicast protocol (HRPM) [22], hierarchical geographic multicast routing (HGMR) [23], hierarchical differential destination multicast (HDDM) [24], and scalable position-based multicast (SPBM) [25] have been proposed.

Our work, EAGER, is unique. It adaptively uses location-based unicast and source-based multicast approaches in order to reduce the computational overhead of forwarding. It also minimizes packet header overhead using enhanced state encoding capability, as well as tree construction overhead using on-demand path information-based tree construction. Table 2 summarizes strengths, weaknesses, and examples of classified multicast protocols.

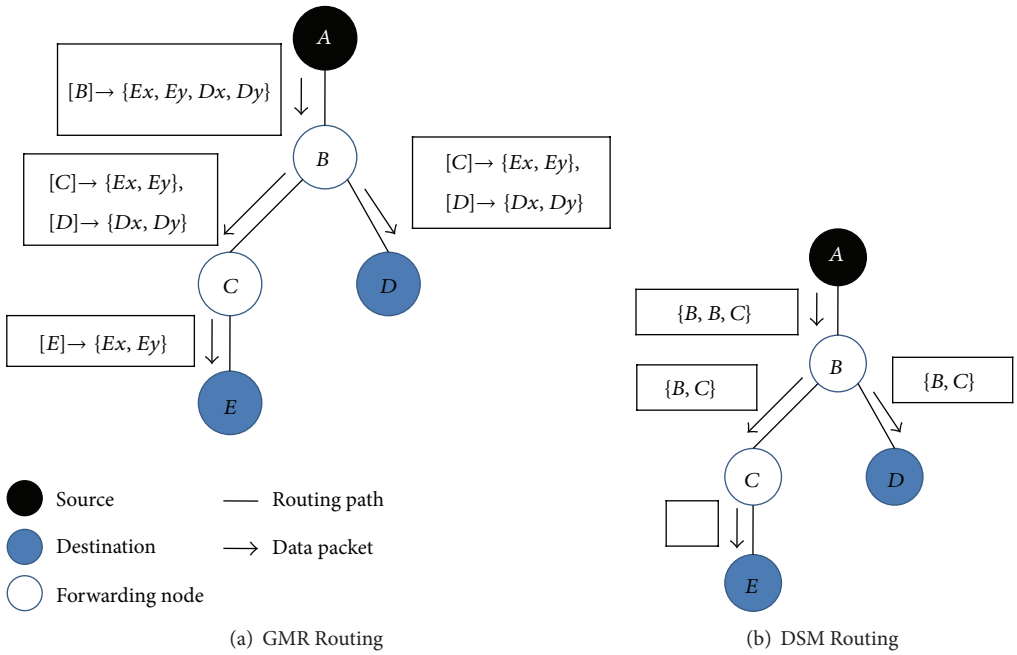
3. Energy-Efficient Adaptive Geosource Multicast Routing

In this section, we first give a brief background on GMR and DSM that are representative examples of location-based multicast routing and source-based multicast routing, respectively, as EAGER optimizes their advantages adaptively. We then describe the detailed EAGER protocol for the following three main operations: (1) multicast tree construction, (2) routing path encoding, and (3) packet forwarding method.

We have chosen GMR and DSM as the best representative schemes for large size multicasting. Inherently, LGT [15] and DDM [21] have been designed for small group multicast and are not scalable to large sized networks. GMR has shown that PBM [14] needs larger computation time and number of data transmissions than GMR. With the same packet size per transmission, the larger number of data transmissions result, in larger total packet sizes.

3.1. Background. GMR assumes to have the entire multicast membership information at the multicast root node like other stateless protocols. However, instead of building a multicast tree for all destinations, the multicast root node selects forwarding nodes among its neighbors according to the cost and progress ratio to the destinations. Hence, the packet header only carries selected forwarding neighbor IDs and a list of the destinations for each forwarding node. Figure 2(a) illustrates how GMR routing works. A source (A) broadcasts a packet that has a neighbor id (B) and $x - y$ coordinates of destinations $\{Ex, Ey, Dx, Dy\}$. Upon receiving the message packet, each selected forwarding node calculates the next forwarding nodes for the given destinations among its neighbors. That is, node B calculates a neighbor id for each path. In this example, a node C is selected as a neighbor for a destination E, and a node D is chosen for a destination D, respectively. Subsequently, the node B broadcasts the packet with chosen neighbors and the coordinates of the destinations. The multicast packet is eventually propagated to destinations using the next forwarding neighbor calculation on each forwarding node. Each forwarding node performs approximately $O(min(neighbors, destinations)^3)$ calculations to select forwarding neighbors.

DSM assumes that each node has the entire network topology using periodic location flooding information. The root node locally computes a Steiner tree for the multicast group. For example, the tree at Figure 2(b) is a Steiner tree that the root node A creates. The packet header carries encoded multicast tree path information (node IDs) using the Prüfer sequence [26]. Node IDs in the Prüfer sequence [26] represent interior nodes in paths (not leaf-nodes). Upon receiving the message packet, each child node, which is inside the Prüfer sequence, decodes the sequence, creates a Steiner subtree, and encodes a new Prüfer sequence to the packet header. For example, a node B receives $\{B, B, C\}$ sequence and knows it is a forwarding node because B is in the sequence.



(a) GMR Routing (b) DSM Routing

Figure 2: GMR versus DSM routing.

The node B decodes the sequence and creates a Steiner subtree without a node A. After that, node B creates and broadcasts the $\{B, C\}$ sequence. The packet is relayed until it reaches leaf nodes. When node D receives the packet, D is not in a sequence. Therefore, D knows that D is a leaf node and stops forwarding the packet. The complexity of tree encoding is $O((pathLength)^2)$ on each child node.

3.2. EAGER Algorithms and Operations. EAGER protocol consists of algorithms for multicast tree construction, routing path encoding, and a packet forwarding method. We next describe each of them in details.

3.2.1. Multicast Tree Construction. The existing source multicast routing protocols assume that every node maintains the entire network topology information using location flooding. Each multicast root node constructs a multicast tree for the given destinations using the network topology. However, the periodic location flooding is expensive as it consumes much of the energy, especially for large networks. To save the cost of building a multicast tree, in EAGER, each multicast root node (or rendezvous point (RP)) obtains the path information to the destination during the multicast membership establishment stage instead of the location flooding. Each join request message carries its path information toward the multicast root node along with its location information. For example, when a member node joins by using geographic unicast, each intermediate node in the path adds its location information to the join packet. The multicast root eventually receives reverse geographic shortest path information to the destination. A multicast tree is created on-demand using the path information in $O(no.paths)$. According to the path information obtained by each join message, the multicast root further optimizes the multicast tree identifying the common path segments among the destinations.

The scheme works as follows. First a destination node, dst, sends a join message, joinMsg, toward the source, src. The join message is relayed to the next forwarding node among the neighbor nodes that is geographically close to src. As illustrated in Algorithm 1, an intermediate node maintains a temporary multicast state table named MState with the information of [dstSeg, hopCnt]. dstSeg is a list of destination node IDs, and hopCnt is the longest hop count from the intermediate node to the destinations in the list. When a joinMsg arrives at an intermediate node, if any dstSeg in the MState table contains the same dst ID and the existing hopCnt is larger than the new join message, the join message will be dropped. For example, in Figure 3, if the destination node j has already joined the multicast group, the intermediate node d will maintain the destination nodes both i and j in the dstSeg.

When the join message from the destination node i arrives in the intermediate node d, d will stop sending the join message as the MState table contains i, and the hopCnt is greater than the hop count from the join message. If the hopCnt is less than the new join message's hop count, the dstSeg and hopCnt will be updated with the new path information and hop count, respectively. If the dst ID is new, a new entry will be added in the MState table. The intermediate node adds the location information to the join message and forwards the join message to the next forwarding node toward the src node. The MState table will be maintained temporarily during the membership establishment stage. The MState table on an intermediate node helps the src node to make compressed path information to the destinations, but is not used for data packet forwarding.

```
(1) // MState: multicast state table, contains dst and hopCnt pairs
(2) if joinMsg(dst) ∈ MState(dstSeg) then
(3)   if hopCnt < MState(dstSeg).hopCnt then
(4)     MState(dstSeg) = dst
(5)     MState(dstSeg).hopCnt = hopCnt
(6)     Add the location information to joinMsg
(7)     Send the joinMsg to the next hop toward src
(8)   end if
(9) else
(10)   Add [dstSeg, hopCnt] to Mstate
(11)   Add the location information to joinMsg
(12)   Send the joinMsg to the next hop toward src
(13) end if
```

Algorithm 1: Group management at intermediate nodes.

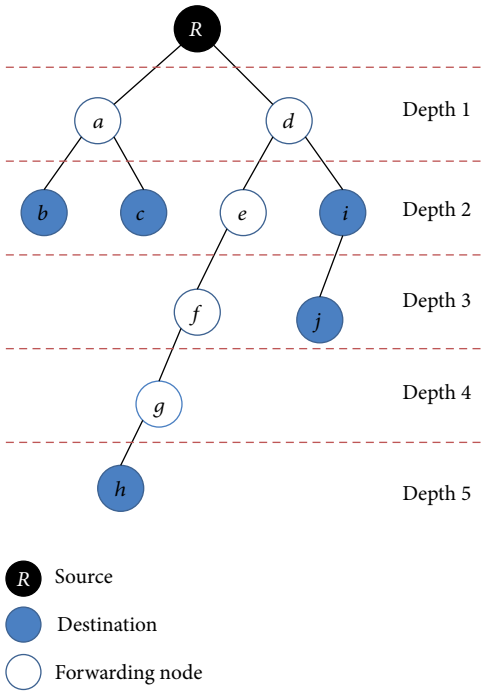


Figure 3: Construction of multicast tree.

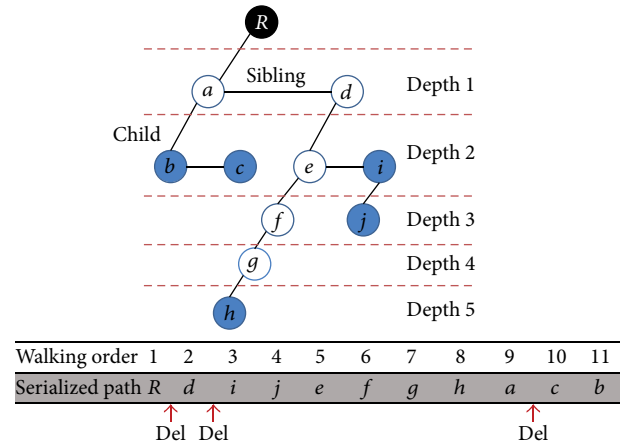


Figure 4: LCRS binary tree and a serialized path sequence.

Once a joinMsg is received by the src node, it constructs a multicast tree starting with a path segment that is used by the highest number of destinations, until it includes all the destinations. For example, in Figure 3, when the src node R has received join messages from h and j in sequence, the src node can identify a common path segment of $\{R, d\}$ from the paths $\{R, d, e, f, g, h\}$ and $\{R, d, i, j\}$.

3.2.2. Routing Path Encoding. In stateless source-based multicast routing protocols, the multicast root node encodes the multicast tree into the packet header using tree structure algorithms such as the Prüfer sequence algorithm. The encoded multicast tree information will be decoded on each intermediate node and re-encoded for the subtree entries before sending the packet. For example, the Prüfer sequence algorithm takes $O(pathLength)$ packet header size. However, the complexity of multicast tree encoding and decoding is $O((pathLength)^2)$ on each intermediate node. To avoid the expensive encoding and decoding overhead on each intermediate node, EAGER serializes the subtree path information using an LCRS (left child right sibling) binary tree [27]. As the tree serialization requires additional delimiters, it may result in a slightly bigger packet size than the other source multicast routing protocols. For example, compared with the pure source multicast scenarios of the spanning tree with n number of nodes, the encoding ratio $(n-2)$ of the Prüfer sequence algorithm used in DSM can be slightly better than our LCRS-based serialized path encoding algorithm ($n-1$ + number of branch delimiters). However, EAGER is designed to have less computation overhead on each intermediate node. The computation complexity of EAGER is $O(1)$ while the Prüfer sequence algorithm has $O(n^2)$.

Algorithms 2, 3, and 4 show how serialization algorithms work. First, the encoding algorithm translates the original n-*ary* multicast tree to an LCRS (left child and right sibling) binary tree. Starting from the multicast root node, the left

```
Input: source node
Output: path
  (1) path = ∅
  (2) PathSerializing(a source node, path)
```

Algorithm 2: Routing path encoding.

```
Input: node
Output: path
  (1) if node == Null then
  (2)     return
  (3) end if
  (4) if node → right then
  (5)     PathSerializing(node → right)
  (6) end if
  (7) if IsBranch(node) then
  (8)     // if a node has more than two children
  (9)     if ID(node) ≠ ID(source node) then
  (10)        add(path, ID(node))
  (11)    end if
  (12)    add(path, Delimiter)
  (13)    PathSerializing(node → left)
  (14) else
  (15)    if IsLongPath(node) then
  (16)        // if there are more than three subsequent children
  (17)        // that is, child - grandchild - grandgrandchild ...
  (18)        TmpNode = PathSerializingForUnicast(node)
  (19)        add(path, XCoordinate(TmpNode))
  (20)        'add(path, YCoordinate(TmpNode))
  (21)        node = TmpNode
  (22)    else
  (23)        add(path, ID(node))
  (24)    end if
  (25)    PathSerializing(node → left)
  (26) end if
```

Algorithm 3: Path serializing.

most child of a node becomes the left child of the new binary tree, and the right most sibling becomes the right child of the new binary tree. For example, the original tree in Figure 3 becomes the new LCRS binary tree in Figure 4. Second, serialized path information is created by walking along the LCRS binary tree in the order of "sibling first, then child node." As illustrated in Figure 4, the serialized path $\{d, i, j, e, f, g, h, a, c, b\}$ is created by walking through the LCRS binary tree with a few additional delimiters. EAGER uses a fixed size information block to encode the state information. The serialized path is presented as consecutive information blocks. The information block can be used as a node ID, location coordinates, or a delimiter. Figure 7 shows a 2-byte delimiter format example. A delimiter can be distinguished from other information blocks by setting 1 in the most significant bit of an information block. Each delimiter has two 7-bit offsets. The node ID block can be identified by setting 00 in the left 2 bits. With a 2 byte information block, the maximum number of node IDs can be about 16 K (use only 14 bits). Branch delimiters are inserted next to the original tree branches' serialized path to indicate the original tree's sibling relationship. That is, the subtree information for each sibling node is separated by the branch delimiters.

Furthermore, EAGER optimizes the encoded packet size adaptively using branch geographic information for common source routing path segments as can be seen at line 15 to 21 of Algorithm 3. It identifies long nonbranchingrouting path segments and uses the branch locations for the source routing information instead of many node IDs along the path. A long nonbranchingpath segment is identified during the serialized path creation if a node has more than three subsequent children; that is, child-grandchild-great grandchild. The serialized path is minimized using the location information (i.e., x and y coordinates) instead of putting the entire node IDs in the path. As we see in Algorithm 4, if a forwarding node finds the location information (delimiter value is 01) in the serialized path, it uses a geographic unicast toward the next branching node. Although it requires each forwarding node to run a geographic unicast algorithm,

```
Input: node
Output: a branch or a destination node
(1) if IsBranch(node) then
(2)     return node
(3) end if
(4) // a leaf node
(5) if node → left == Null then
(6)     return node
(7) end if
(8) PathSerializingForUnicast(node → left)
```

Algorithm 4: Path serializing for unicast.

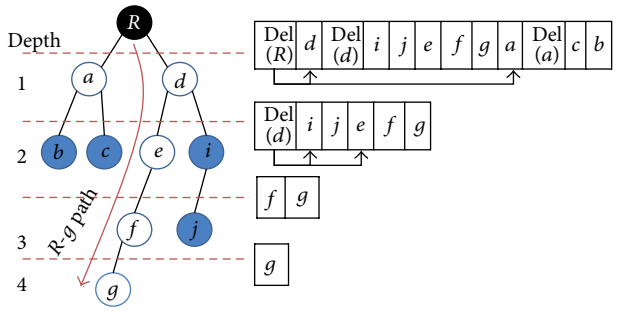
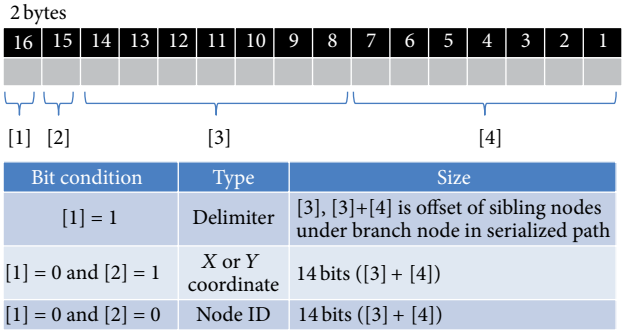


Figure 5: Packet decoding/forwarding on the short path nodes.



Bit condition	Type	Size
[1] = 1	Delimiter	[3], [3]+[4] is offset of sibling nodes under branch node in serialized path
[1] = 0 and [2] = 1	X or Y coordinate	14 bits ([3] + [4])
[1] = 0 and [2] = 0	Node ID	14 bits ([3] + [4])

Figure 7: 2 Bytes information block example.

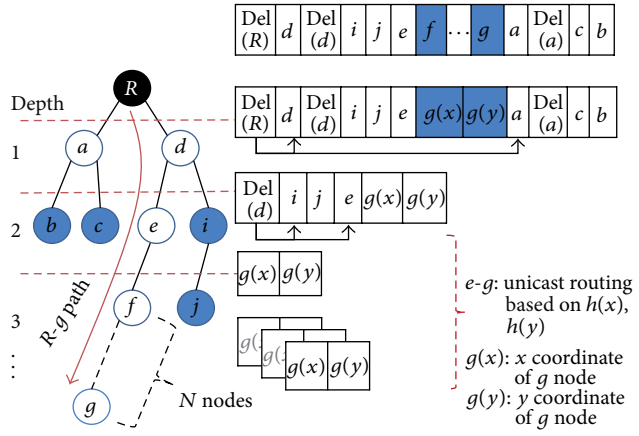


Figure 6: Packet decoding/forwarding with long path optimization.

the computation complexity is minimal. We name *long-path optimization* as the technique to reduce packet size by using x and y coordinates for a long nonbranchingpath.

3.2.3. Packet Forwarding. When a multicast packet is received by a forwarding node, the node selects the serialized path for its own subtree according to the branch delimiter information. Figure 5 illustrates how to utilize the serialized path information along the R to g forwarding track on the short path nodes. For example, node d can make a new serialized path for its subtree by checking the offset from the first delimiter (R). *Del(R)* points node d and node a, which are forwarding nodes. Node d recognizes that it is a forwarding node with *Del(R)*. By using line 3 to 7 of Algorithm 5, node d extracts the path of the next subtree, {*Del*(d), i, j, e, f, g}, and broadcasts with the path. In this case, there is no additional calculation overhead except the simple packet truncation for the next subtree, and computation overhead at each forwarding node.

However, in case of a long non-branch path, all node IDs in the long path should be contained into serialized paths. In order to reduce the size overhead of node IDs due to the long path, we use *long-path optimization*. As shown in Figure 6, there is a long (n nodes) nonbranchingpath segment between node f and g. In this case, a multicast packet is delivered to node e by using the same way as Figure 5, but node e sends the packet to node g by the geographic unicast routing because of the existence of the long and non-branch path (f to g). The path segment can be represented by only 2 information blocks; thus, the information is reduced by $2(n - 2)$ bytes. It also results in a great saving of packet size along the path. The algorithms for the forwarding operation are shown in Algorithms 5 and 6.

4. Performance Evaluation

In this section, we evaluate the performance of EAGER and compare it with the performance of GMR and DSM. We implemented EAGER using an NS2 (v2.35) simulator. We used a grid network topology. Most evaluations were

```
Input: serialized path, node
(1) if FirstBit(path[0]) == 1 then
(2)    // the first byte in path is a delimiter
(3)    if ID(node) ∈ {IDs pointed by the delimiter} then
(4)       pos1 = indexInPath(node) + 1
(5)       pos2 = indexInPath(next node pointed by delimiter after me) − 1
(6)       path = {path[pos1], . . ., path[pos2]}
(7)       Forward(path)
(8)    end if
(9) else
(10)    if FirstTwoBits(path[0]) == 01 then
(11)       // the first byte in path is x coordinate
(12)       // path [0], path [1] are X and y coordinates
(13)          of a branch or a destination
(14)       if path[2] ≠ Null then
(15)          path = {path[2], . . ., path[length(path) − 1]}
(16)       end if
(17)       Forward(path)
(18)    else if FirstTwoBits(path[0]) == 00 then
(19)       // the first byte in path is an ID
(20)       path = {path[1], . . ., path[length(path) − 1]}
(21)       Forward(path)
(22)    end if
(23) end if
```

Algorithm 5: Packet forwarding at each node.

```
Input: path
(1) if FirstTwoBits(path[0]) == 01 then
(2)    nextHop = GetNextHop(path[0], path[1])
(3)    Unicast packet(with path) to nextHop
(4) else
(5)    Broadcast packet(with path)
(6) end if
```

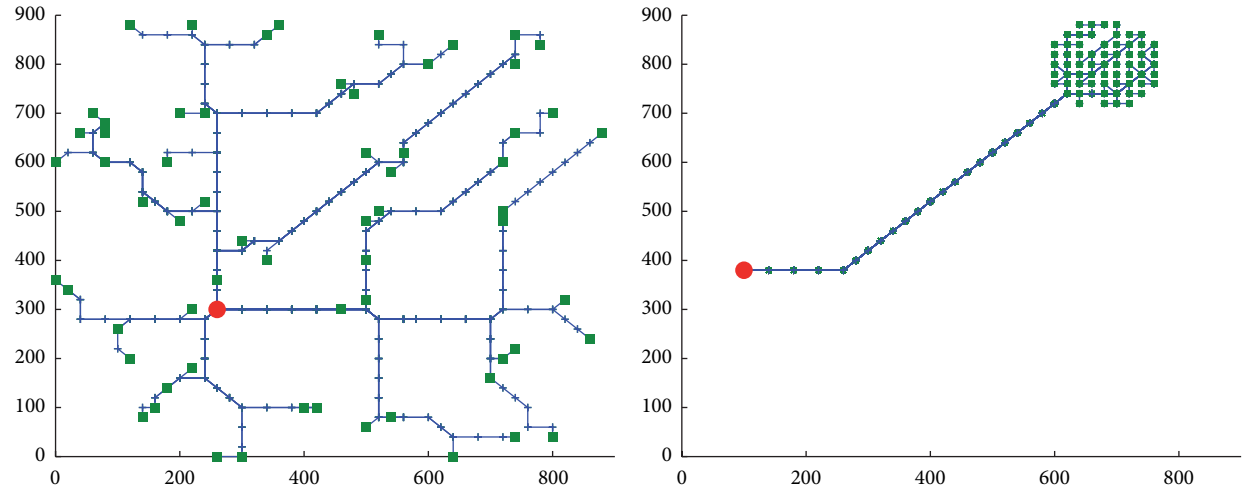
Algorithm 6: Forward(path): Decision of forwarding mechanism.

performed in the network with 2025 nodes unless the network size is not mentioned in this section. The number of neighbor nodes in the communication range is set up to 12 unless specified differently. We assume that there is no packet loss, and the size of the location coordinates of a node is 2 times bigger than the size of the node identifier. The evaluation metrics used were total packet overhead, average computation time, and consumed energy. *Total packet overhead* is the sum of all the multicast packets delivered from a multicast root node to all the destination nodes along a multicast path. *Average computation time* is the average time taken by each forwarding node on the multicast path for neighbor selection and packet re-encoding. *Consumed energy* is the total energy used by the nodes in the multicast path to perform transmission, reception, and computation. Consumed energy is computed by multiplying duration for transmission, reception, and computation by the power consumption (Watt). The power consumption (Watt) ratio of computation, transmission, and reception is shown in Table 3 that corresponds to the cc2420 [28] and ATMega128L [29] specifications.

Table 3: Power consumption ratio.

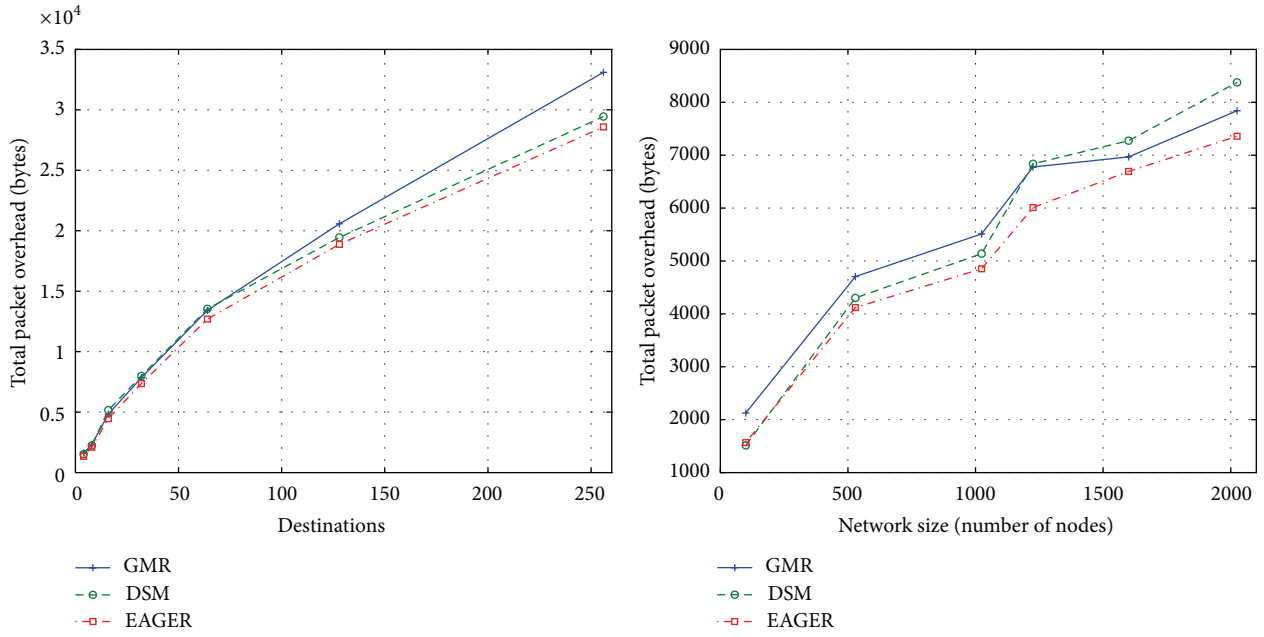
Operations	Consumed power (Watt)
Computation	0.0000459
Transmission	0.0001
Reception	0.000132

As for the placement of destinations, we used both random and clustered destinations. Many studies [30–34] have shown that clustered destinations are common for the group communication applications. Clustered WSNs were used to achieve efficient energy usage, a long network lifetime, and high network coverage. To evaluate clustered destinations, we used various configurable parameters, including the number of clusters, the number of nodes in a cluster, the distance between a source and a cluster, and the radius of a cluster.



(a) Random 64 destinations (b) Clustered 64 destinations (1 cluster)

Figure 8: Example of random and clustered destinations.

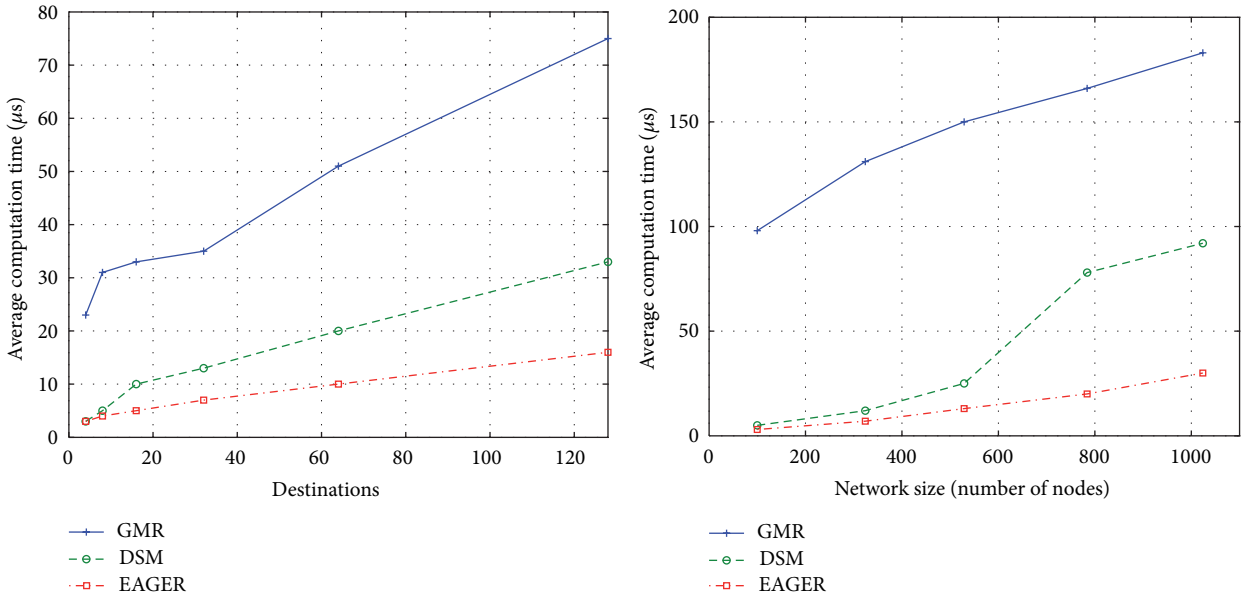


(a) Varied Number of Destinations (b) Varied Network Sizes

Figure 9: Total packet overhead comparison with random destinations.

4.1. Random Destinations. We evaluated packet processing overhead including total packet overhead and average computation time and then quantified the consumed energy with random destinations for EAGER, GMR, and DSM. We randomly selected destinations as well as a multicast root node. Figure 8 shows examples of random and clustered destinations with 64 destinations. A solid circle, solid squares, crosses, and lines represent a source, destinations, forwarding nodes, and routing paths, respectively.

The *total packet overhead* for the different number of destinations is shown in Figure 9(a). The numbers of destinations are varied with 4, 8, 16, 32, 64, 128, and 256 nodes. While the number of destinations increases, EAGER and DSM use less packets than GMR. Since GMR encodes the packet header with the destination locations, the packet header size becomes larger as the number of destinations increases. For example, if the number of destinations is less than 64, the total packet overhead of GMR is smaller than



(a) Varied Number of Destinations

(b) Varied Network Sizes

Figure 10: Average computation time comparison with random destinations.

that of DSM. However, if the number of destinations becomes greater than 64, GMR has a bigger packet overhead than other protocols. The result also shows that EAGER uses slightly smaller packet sizes than DSM, as EAGER can reduce the packet sizes adaptively using branch geographic information for common source routing path segments.

Next, we examined the total packet overhead for the different network sizes in Figure 9(b). The network size ranges from 100 to 2025 nodes while the number of destinations is fixed with 30 nodes. The result also shows that the total packet overhead of EAGER is always smaller than that of GMR and DSM. Because DSM encodes the packet with the multicast tree path, DSM's packet size becomes bigger as the larger network size increases. It also illustrates that DSM is more sensitive to the network size than EAGER as well as GMR. EAGER has a smaller total packet overhead and is less sensitive to the network size than other protocols, as it has a larger chance of having longer and nonbranchingpaths for the large network.

As network size increases to more than 2025, we expect that EAGER has the lowest total packet overhead and DSM has the highest one. However, the network size at which DSM has more total packet overhead than GMR changes depending on the number of destinations. Specifically, the more numbers of destinations are used, the larger network size is needed so that DSM has more total packet overhead than GMR.

The average computation time is compared for the different number of destinations in Figure 10(a). The numbers of destinations are from 4 to 128 in the network with 2025 nodes. GMR requires the most computation time compared to other protocols, resulting in a high CPU overhead. This is because GMR calculates the next forwarding neighbors on each forwarding node and the algorithm complexity increases according to the number of destinations. Meanwhile, in DSM and EAGER, the multicast path information is calculated and encoded by the multicast root node, leading to a lower average computation time. However, for the large number of destinations, the computation time of DSM is higher than that of EAGER due to the encoding and decoding overhead of the forwarding nodes.

We show the average computation time for the varied network sizes in Figure 10(b). The network size is varied from 100 to 1024, while the number of destinations is set with 30% of the network size. The results display that GMR spends a much higher computation time than other protocols, but the time difference is bounded and not proportional to the increment of the network size. Both DSM and EAGER spend minimal computation time and have little dependency on the network size. However, for the larger network size, the computation time of DSM becomes much higher than that of EAGER because the encoding and decoding overhead increase proportionally to the number of nodes on the multicast routing path.

Figure 11 shows the energy consumption for the varied number of destinations. The number of destinations increases from 32 to 256 for the network of 2025 nodes. The result shows that EAGER consumes the least energy. It is because it has a smaller total packet overhead and a lower computation time than the other two protocols. The result also shows that EAGER becomes more energy-efficient than other protocols as the number of destinations increases. DSM shows worse energy efficiency than GMR for the small number of destinations. However, DSM has better energy consumption than GMR when the number of destinations increases. It is also observed that the energy consumption conforms more to

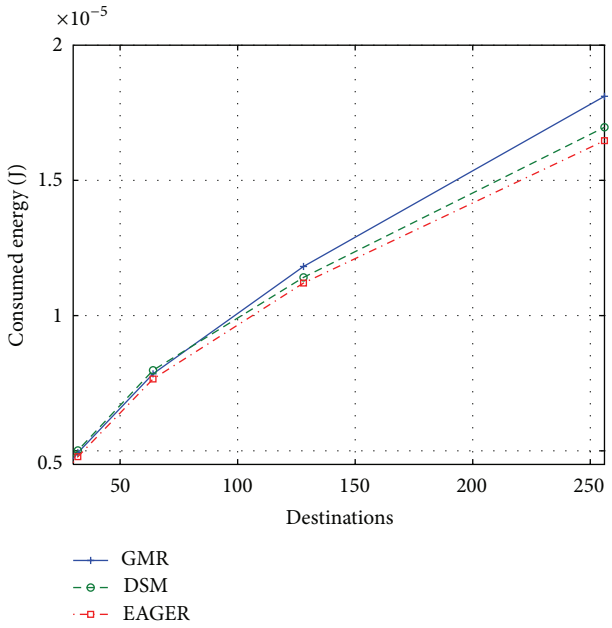


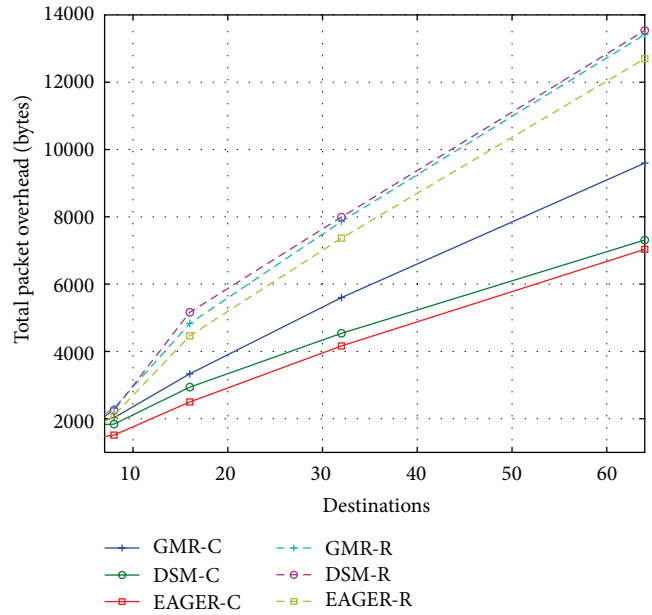
Figure 11: Consumed energy comparison with random destinations.

the total packet overhead than the computation time, as the relative energy consumption for communication is set much higher than the one for computation as in Table 3.

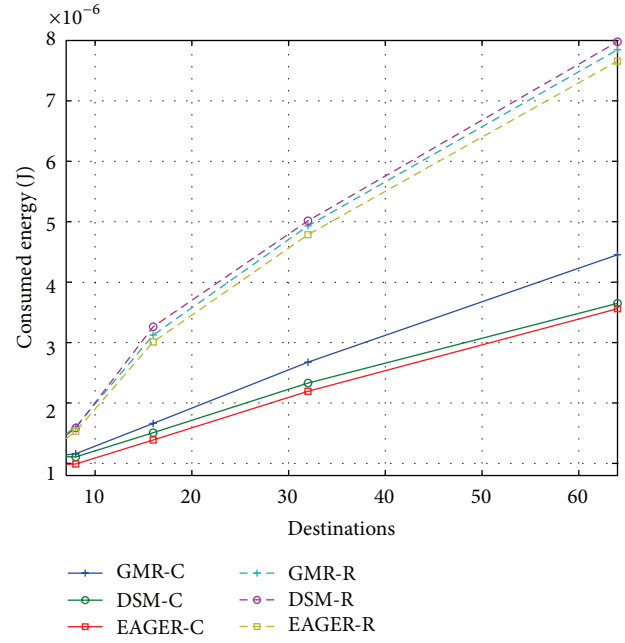
4.2. Clustered Destinations. Here we used the clustered destinations for the comparative evaluations of total packet overhead and consumed energy.

Figures 12(a) and 12(b) illustrate total packet overhead and consumed energy, respectively, with the different number of destinations from 4 to 64. *Protocol*-C means the protocol measured with clustered destinations, and *protocol*-R means the protocol measured with random destinations. We found that all protocols with clustered destinations have smaller total packet overhead and energy consumption than those with random destinations. It was observed that the gap between GMR and DSM becomes larger in clustered destinations than in random ones while the number of destinations increases. This is because the total path lengths from a source and destinations have been reduced much faster in clustered destinations than in random destinations. As EAGER enjoys the benefit of GMR, it outperforms DSM in clustered destination scenarios. It also always has a smaller total packet overhead than GMR as well, due to compact packet encoding.

We next varied the different distances between a source and a cluster head from 600 to 850 meters while fixing the number of destinations as 8 and measured total packet overhead and consumed energy in Figures 13(a) and 13(b). EAGER exhibits the least energy consumption and total packet overhead than other protocols. DSM shows lower energy consumption and less total packet overhead than GMR in the short path lengths. However, as the path lengths become longer, DSM shows higher energy consumption and



(a) Total Packet Overhead Comparison

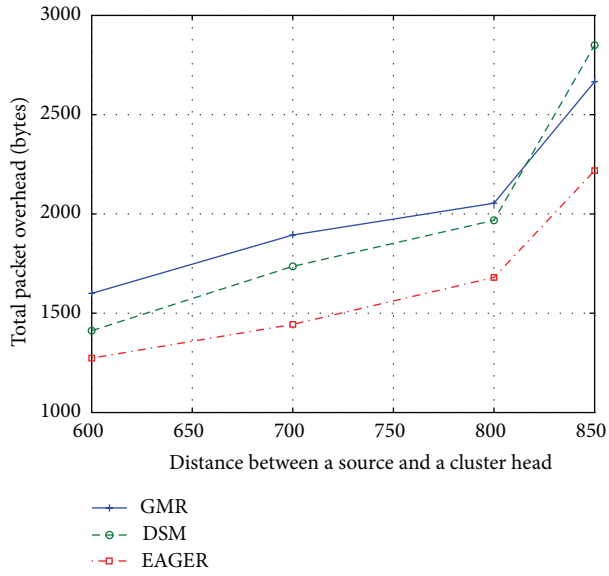


(b) Consumed Energy Comparison

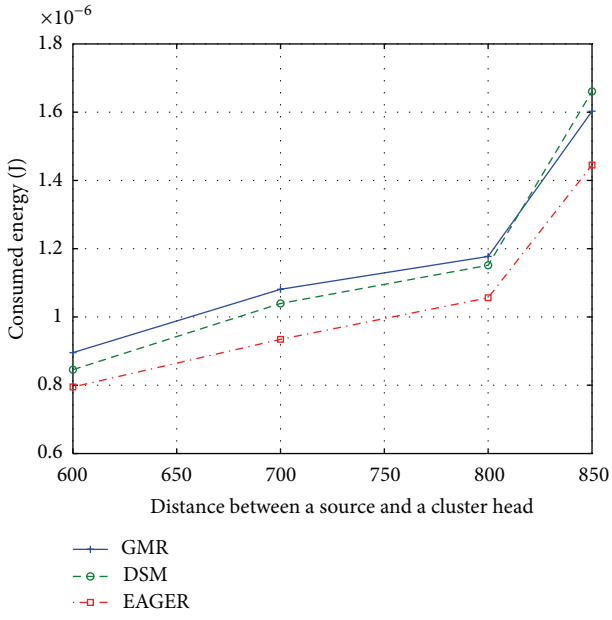
Figure 12: Total packet overhead and consumed energy comparisons with varied number of destinations, (-R: random destinations, -C: clustered destinations).

larger total packet overhead than GMR, since DSM's packet header has to include all of the path information.

Finally, we varied the number of clusters from 1 to 8 while the total number of destinations is fixed, in Figures 14(a) and 14(b). They demonstrate that the total packet overhead and energy consumption with EAGER shows less energy consumption and total packet overhead than other protocols as the number of clusters increases. DSM shows lower energy

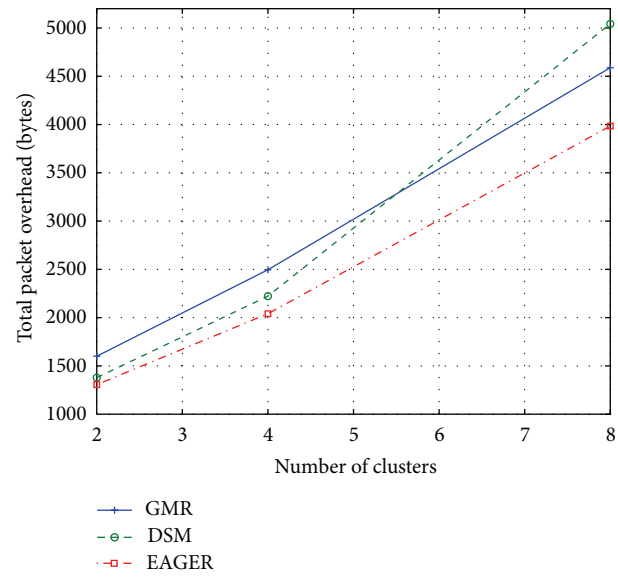


(a) Total Packet Overhead Comparison

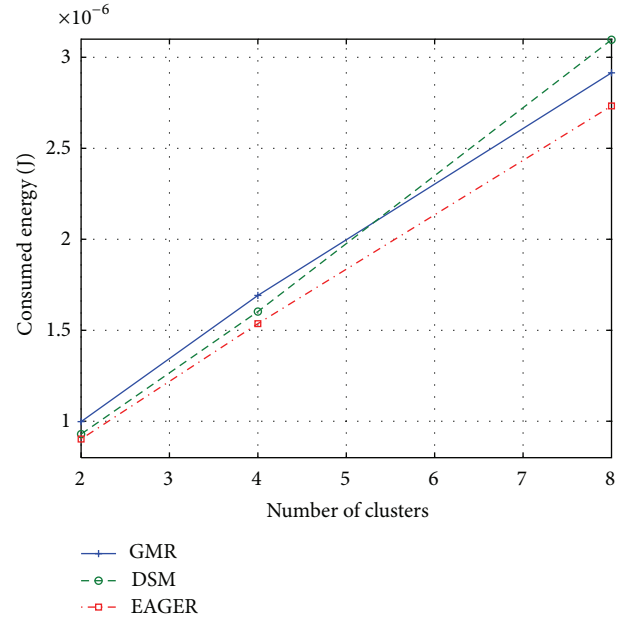


(b) Consumed Energy Comparison

Figure 13: Total packet overhead and consumed energy comparisons with varied cluster path length.



(a) Total Packet Overhead Comparison



(b) Consumed Energy Comparison

Figure 14: Total packet overhead and consumed energy comparisons with varied number of clusters of destinations.

consumption and less total packet overhead than GMR in the small number of clusters. However, as the number of clusters increases, DSM shows higher energy consumption and larger total packet overhead than GMR. This is because the path length increases as the number of clusters increases.

In all the various evaluation scenarios, EAGER outperformed both source-based and location-based multicast protocols, not only taking the advantage of each effectively, but also enhancing each of them with efficient encoding and forwarding operations.

5. Conclusion

We have presented a novel stateless path information-based multicast protocol, named EAGER (energy-efficient adaptive geo-source multicast routing) for WSNs. EAGER optimizes the previous location-based multicast and source multicast approaches by adaptive usage of geographic unicast and source multicast routing. It is also equipped with unique features including on-demand tree construction using path information, light-weight forwarding, and enhanced state encoding capability. Our extensive simulation results exhibit that EAGER outperforms GMR and DSM in computation

time, packet overhead, and energy consumption while maintaining the advantages of stateless protocols.

References

[1] "Imote2: High-Performance Wireless Sensor Network Node," http://docs.tinyos.net/tinywiki/index.php/IMote2.

[2] "TMote Sky," http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf.

[3] J. G. Ko, Q. Wang, T. Schmid, W. Hofer, P. Dutta, and A. Terzis, "Egs: a Cortex M3-based mote platform," in *Proceedings of the 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '10)*, pp. 1–3, IEEE, June 2010.

[4] "Micaz," http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf.

[5] "SunSPOT," http://www.sunspotworld.com/products/index.html.

[6] "TelosB," http://www.willow.co.uk/TelosB_Datasheet.pdf .

[7] C. Gui and P. Mohapatra, "Efficient overlay multicast for mobile Ad Hoc networks," in *Proceedings of the IEEE Wireless Communications and Networking (WCNC '03)*, vol. 2, pp. 1118–1123, March 2003.

[8] J. G. Jetcheva and D. B. Johnson, "Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks," in *Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '01)*, pp. 33–44, October 2001.

[9] S.-J. Lee, M. Gerla, and C. C. Chiang, "On-demand multicast routing protocol," in *Proceedings of the IEEE Wireless Communications and Networking (WCNC '99)*, September 1999.

[10] E. M. Royer and C. E. Perkins, "Multicast operation of the Ad-Hoc on-demand distance vector routing protocol," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99)*, pp. 207–218, 1999.

[11] C. W. Wu and Y. C. Tay, "AMRIS: a multicast protocol for ad hoc wireless networks," in *Proceedings of the IEEE Military Communications Conference (MILCOM '99)*, pp. 25–29, November 1999.

[12] J. Xie, R. R. Talpade, A. McAuley, and M. Liu, "AMRoute: Ad Hoc multicast routing protocol," *Mobile Networks and Applications*, vol. 7, no. 6, pp. 429–439, 2002.

[13] S. Basagni, I. Chlamtac, and V. R. Syrotiuk, "Location aware, dependable multicast for mobile ad hoc networks," *Computer Networks*, vol. 36, no. 5-6, pp. 659–670, 2001.

[14] M. Mauve, H. Fuessler, J. Widmer, and T. Lang, "Positionbased multicast routing for mobile Ad-Hoc networks," Tech. Rep. CS TR-03-004, University of Mannheim, Baden-Württemberg, Germany, 2003.

[15] K. Chen and K. Nahrstedt, "Effective location-guided tree construction algorithms for small group multicast in MANET," in *Proceedings of the 21th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, pp. 1180–1189, June 2002.

[16] X. Liu, J. A. Sanchez, P. M. Ruiz, and I. Stojmenovic, "GMR: geographic multicast routing for wireless sensor networks," in *Proceedings of the 3rd Annual IEEE Communications Society on Sensor and Ad hoc Communications and Networks (Secon '06)*, pp. 20–29, September 2006.

[17] S. Pennington, A. Waller, and T. Baugé, "RECOUP: efficient reconfiguration for wireless sensor networks," in *Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services (Mobiquitous)*, pp. 33:1–33:2, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[18] H. Rowaihy, M. P. Johnson, O. Liu, A. Bar-Noy, T. Brown, and T. La Porta, "Sensor-mission assignment in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 6, no. 4, 2010.

[19] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 34–40, 2004.

[20] P. T. Eugster, J. Luo, and J. P. Hubaux, "Route driven gossip: probabilistic reliable multicast in ad hoc networks," in *Proceedings of the 22nd Annual Joint Conference on the IEEE Computer and Communications Societies (IEEE INFOCOM '03)*, pp. 2229–2239, San Francisco, Calif, USA, April 2003.

[21] L. Ji and M. S. Corson, "Differential destination multicast-a MANET multicast routing protocol for small groups," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '01)*, pp. 1192–1201, April 2001.

[22] S. M. Das, H. Pucha, and Y. C. Hu, "Distributed hashing for scalablemulticast in wireless ad hoc networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 3, pp. 347–361, 2008.

[23] Y. C. Hu, I. Stojmenovic, D. Koutsonikolas, and S. Das, "Hierarchical geographic multicast routing for wireless sensor networks," in *Proceedings of the International Conference on Sensor Technologies and Applications (SENSORCOMM '07)*, pp. 347–354, October 2007.

[24] M. Transie, H. Fuler, J. Widmer, M. Mauve, and W. Effelsberg, "Scalable multicasting in mobile ad hoc networks," in *Proceedings of the 23th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '04)*, vol. 3, pp. 2119–2129, March 2004.

[25] M. Transie, H. Fuler, J. Widmer, M. Mauve, and W. Effelsberg, "Scalable position-based multicast for mobile Ad-Hoc networks," in *Proceedings of the International Workshop on Broad-band Wireless Multimedia: Algorithms, Architectures and Applications (Broad- Wim)*, October 2004.

[26] J. Gottlieb, B. A. Julstrom, G. R. Raidl, and F. Rothlauf, "Prüfer numbers: a poor representation of spanning trees for evolutionary search," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '01)*, pp. 343–350, 2001.

[27] NIST Dictionary of Algorithms and Data Structures, "Left child right sibling tree".

[28] "CC2420," http://focus.ti.com/lit/ds/symlink/cc2420.pdf.

[29] "ATMega128L," http://www.atmel.com/dyn/resources/prod_documents/2467s.pdf.

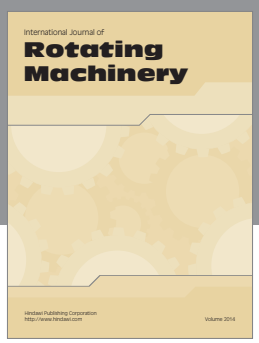
[30] W. Robertson, S. Sivakumar, N. Aslam, and W. Phillips, "A multi-criterion optimization technique for energy efficient cluster formation in wireless sensor networks," *Information Fusion*, vol. 12, no. 3, pp. 202–212, 2011.

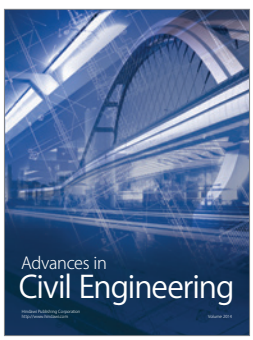
[31] N. Vlajic and D. Xia, "Wireless sensor networks: to cluster or not to cluster?" in *Proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '06)*, pp. 258–266, June 2006.

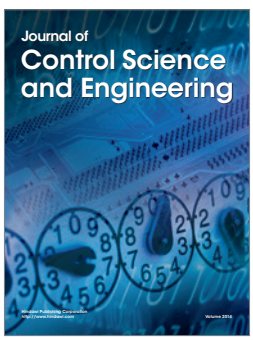
[32] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[33] N. Xu, A. Huang, T. W. Hou, and H. H. Chen, "Coverage and connectivity guaranteed topology control algorithm for cluster-based wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 12, no. 1, pp. 23–32, 2012.

[34] O. Younis and S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for Ad-Hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.







Hindawi
Submit your manuscripts at
http://www.hindawi.com