TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 03/09 pp475-490
Volume 16, Number 5, October 2011

Interference-Aware Multipath Routing Protocol for QoS Improvement in Event-Driven Wireless Sensor Networks

Marjan Radi**, Behnam Dezfouli, Kamalrulnizam Abu Bakar, Shukor Abd Razak, Mohammad Ali Nematbakhsh†

Faculty of Computer Science and Information Systems, Universiti Teknologi Malaysia, Johor, Malaysia;
† Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran

Abstract: The existing multipath routing protocols for wireless sensor networks demonstrate the efficacy of traffic distribution over multiple paths to fulfill the Quality of Service (QoS) requirements of different applications. However, the performance of these protocols is highly affected by the characteristics of the wireless channel and may be even inferior to the performance of single-path approaches. Specifically, when multiple adjacent paths are being used concurrently, the broadcast nature of wireless channels results in inter-path interference which significantly degrades end-to-end throughput. In this paper, we propose a Low-Interference Energy-efficient Multipath Routing protocol (LIEMRO) to improve the QoS requirements of event-driven applications. In addition, in order to optimize resource utilization over the established paths, LIEMRO employs a quality-based load balancing algorithm to regulate the amount of traffic injected into the paths. The performance gain of LIEMRO compared to the ETX-based single-path routing protocol is 85%, 80%, and 25% in terms of data delivery ratio, end-to-end throughput, and network lifetime, respectively. Furthermore, the end-to-end latency is improved more than 60%.

Key words: LIEMRO; QoS; load balancing; interference; multipath routing; event-driven

Introduction

With the advances in Micro-Electro-Mechanical Systems (MEMS), radios, and microcontrollers, wireless sensor networks have gained world-wide attention in the recent years. Not only in science and in engineering, wireless sensor networks are being used for a wide range of applications such as healthcare, battlefield surveillance, habitat monitoring, and disaster management[1]. Depending on the type of application in which a wireless sensor network is being deployed, certain quality parameters should be satisfied during the data transmission process. Accordingly, as wireless sensor nodes are limited in energy and processing capacities, designing energy-efficient low-overhead communication protocols to satisfy the performance requirements of different applications is considered as an important research area[1,2]. The design of efficient protocols for Quality of Service (QoS) improvement is further complicated due to the unique features of low-power wireless communication such as link unreliability and high sensitivity to interference[3].

In the effort to support QoS demands, researchers have proposed various algorithms and protocols for different layers of the wireless sensor network protocol stack. Meanwhile, the significance of QoS-aware routing protocols was demonstrated in Refs. [2,4]. Since the traffic generation pattern influences the performance of routing protocols, different applications demand different routing strategies. Since most of the

Received: 2011-06-21; revised: 2011-08-29
** To whom correspondence should be addressed.
E-mail: radi@ieee.org

existing routing protocols are designed based on the single-path routing strategy, they cannot support high data rate event reporting. In this strategy, a single path is established based on the QoS requirements of the underlying application to transmit data packets from the source nodes to the sink node. Therefore, the end-to-end throughput is limited to the capacity of a single path[5]. Additionally, due to the constant flow of data through a specific set of the nodes, these nodes consume their energy quite fast. This issue results in network disjointedness and reduced sensing coverage. In contrast, multipath routing protocols enable the source nodes to discover several paths towards the destination. Since in multipath routing technique data packets are propagated over several paths, it provides higher throughput, more balanced energy consumption and improved latency[6,7].

Since different links of a wired network are independent of each other, the multipath routing approach has been widely utilized to increase network throughput and reliability in this type of networks. However, since a wireless channel is a broadcast medium and wireless links are unreliable, the multipath routing technique in wireless sensor networks seems to be less profitable than those proposed for wired networks. In fact, transmitting data over multiple paths does not necessarily lead to performance improvement without considering the effects of the wireless communications[8,9]. Accordingly, the effects of inter-path and intra-path interference should be considered in the route discovery and load distribution phases of multipath routing protocols.

Regarding to the inherent advantages of multipath routing techniques, we propose a novel multipath routing protocol to improve the QoS demands of event-driven applications (e.g., intrusion detection, disaster management and target tracking). The main contributions of this paper are as follows:

- A Low-overhead Interference-Minimized Multipath Routing Protocol (LIEMRO) is proposed to discover multiple paths from each source node to the sink. The routing protocol includes parameters such as residual energy, link quality, and interference level to establish a set of node-disjoint interference-minimized paths. Since increasing the number of active paths does not necessarily result in higher performance (due to inter-path interference), a newly established path will be used for data transmission if it results in increased end-to-end throughput; otherwise, LIEMRO disables the newly established path and terminates the route discovery and establishment phase.
- We propose a load balancing algorithm in conjunction with the multipath routing protocol to distribute source node traffic over the established paths. The load balancing algorithm estimates the optimal traffic rate of the paths according to their relative quality. Specifically, a lower traffic rate is assigned to paths with higher interference levels. Since finding and establishing interference-mini-mized paths may be time consuming, LIEMRO starts packet transmission immediately after the first path is established. This reduces the time interval between event detection and packet reception at the sink. Afterward, whenever a new path is created, the load balancing algorithm redistributes the source node traffic according to the relative quality of the paths.
- Through comprehensive evaluations, we demonstrate the performance improvements of LIEMRO compared to the single-path routing protocol which uses ETX[10] and the residual battery levels in its route discovery mechanism. The simulation results show the higher performance of LIEMRO compared to the single-path routing approach in terms of data delivery ratio, throughput, latency, and network lifetime.

Partial results of this work were published in Ref. [11]. The new contributions of this paper include presenting implementation details and a comprehensive evaluation of LIEMRO from several new aspects.

The rest of this paper is organized as follows: In Section 1 we provide an overview over the existing multipath routing protocols. Then, we introduce LIEMRO in Section 2. We present the proposed load balancing algorithm in Section 3. Performance evaluation and analysis are performed in Section 4. Section 5 concludes and provides directions for future works.

1 Related Work

Multipath routing or so-called "traffic dispersion" is an active field of research on QoS-aware routing protocols for wired and wireless networks. In recent years, this method has also been used as an effective approach to provide QoS in wireless ad hoc and sensor networks[7,12].

Inspired by directed diffusion[13], Ganesan et al.[14] proposed a multipath routing protocol that uses partially disjoint paths to provide fault tolerance against node or link failures. Since the established paths are not completely disjoint, this approach greatly reduces the cost of path construction and maintenance. In addition, this technique uses only one path for data transmission and switches to an alternative path when the active path fails. Therefore, due to data transmission over one path, this protocol does not benefit from the advantages of load distribution over multiple paths. Through utilizing data redundancy in conjunction with a multipath routing approach, EQSR[15] provides reliable data transmission. Since this protocol uses the XOR-based Forward Error Correction (FEC) technique, it imposes significant overhead on the network to compute error correction codes and retrieve original messages. MCMP[16] is another multipath routing protocol, which provides soft-QoS support in terms of reliability and delay. The end-to-end soft-QoS is formulated as a probabilistic programming problem and is converted to a deterministic linear programming using an approximation technique. In order to provide reliability, during the route discovery process each node chooses one or a few of its one-hop neighboring nodes, which additively provide the desired reliability along the path towards the sink node. Here, the source node and all the intermediate nodes should forward multiple copies of data packets over several paths. Hence, the data transmission redundancy of this protocol is in contrast with the resource limitations of wireless sensor nodes. Moreover, since the constructed paths are usually adjacent, simultaneous utilization of these paths causes high inter-path interference and results in elevated packet loss ratios. Furthermore, as the employed neighbor-selection mechanism does not consider the energy consumption over the links, this protocol cannot construct energy efficient paths. To overcome this problem, ECMP[17] improves the energy efficiency of MCMP by introducing an energy optimization problem, constrained by delay, reliability, and geo-spatial energy consumption. In this protocol, an intermediate node selects a set of its next-hop neighboring nodes which additively satisfies the reliability requirements of target application with minimum energy consumption. Similar to MCMP, this protocol also improves data transmission reliability through introducing some data redundancy into the data transmission process. *N*-to-1 multipath routing[18] considers reliability as the primary demand of its intended application. The route construction phase utilizes spanning tree formation to find the shortest paths to the sink. Consequently, as each node's neighbors may belong to different branches of the tree, nodes are able to find alternative paths towards the sink through some of their neighbors. Still, since this protocol establishes physically proximal paths, it suffers from the negative effects of wireless interference.

Hurni and Braun[19] proposed a multipath routing protocol based on AOMDV[20]. The main objectives of this protocol are to improve energy efficiency and reduce end-to-end latency through load balancing and using cross-layer information. In order to reduce the end-to-end latency of data forwarding, each node utilizes the information provided by the MAC layer to transmit its packets to the neighboring node that wakes up earlier. Since the nodes are aware of their neighbors' schedules, per-hop latency is reduced and the interference problem is addressed at the MAC layer. Nevertheless, the MAC protocol requires the frequent exchange of control packets to update neighbors' schedules. Moreover, this protocol has the main disadvantage of ad hoc-based routing protocols: the whole path information should be propagated throughout the network.

Since sensor nodes have limited energy capacity, the quality of some applications is influenced by the network lifetime and the energy consumption. The multipath routing protocol introduced in Ref. [21] utilizes a multipath routing approach to provide energy-efficient communications through balancing of the network traffic over multiple paths. To this aim, the residual battery life in the nodes is the most important metric considered in the route discovery phase. Nevertheless, as this protocol neglects the effects of wireless interference and assumes error-free links, it cannot achieve significant performance improvement in throughput and data delivery ratio. A non-interfering multipath routing protocol (I2MR) to support high rate streaming in wireless sensor networks was presented in Ref. [22]. This protocol constructs zone-disjoint paths by assuming a specific network model and localization support. It is assumed that there are several gateway nodes (as the final packet destinations) connected directly to the command center using non-interfering and high capacity links. The source node constructs three zone-disjoint paths towards the three distinct gateway nodes. After that, the source node utilizes the primary and secondary paths for data transmission and preserves the third path for prompt packet recovery from path failures. Although I2MR shows higher performance compared to the standard AODV, the network model assumptions and the overhead caused by the localization algorithm offset the potential benefits of this protocol. Moreover, to reduce the negative effects of intra-path interference, I2MR constructs shortest paths towards the gateway nodes. Since the longest hops should be used to create the shortest paths, the time-varying properties of wireless links highly affect the performance achieved by this protocol[10,23].

There exist few approaches to reduce the effects of interference through techniques such as directed antenna[24], multi-channel transmission[25], geographic routing[26], and network conflict graph[27,28]. These techniques cannot be easily utilized in resource-constrained wireless sensor networks.

In contrast to the methods that use data redundancy (either through packet replication or through coding), the protocol proposed in this paper uses the estimated link quality metric to reduce data transmission overhead and improve end-to-end reliability. Furthermore, the existing approaches rely on specific hardware or a network conflict graph to cope with the inter-path interference problem while the proposed approach uses the broadcast nature of the wireless medium to estimate inter-path interference and establish interference-minimized paths in a localized manner. Finally, since the existing protocols mainly utilize the node residual battery life to determine the optimal traffic rate of the paths, they do not account for the effects of wireless interference on the capacity of individual paths. Through including the experienced interference level in the load balancing algorithm, we consider the effect of interference on the tolerable traffic rate of the paths.

2 Routing Protocol

The proposed multipath routing protocol is composed of three phases: (1) initialization phase, (2) route discovery and establishment phase, and (3) route maintenance phase. In the initialization phase, each node acquires its neighborhood information. This information will be used in the route discovery and establishment phase to find the best next-hop node towards the sink. The route discovery and establishment phase is triggered whenever an event is detected. The outcome of this phase is multiple interference-minimized paths between the source and sink. Finally, the route maintenance phase handles path failures during data transmission. The rest of this section provides the detailed operations of each phase.

2.1 Initialization phase

In the initialization phase, each node obtains its neighborhood information, which also includes the ETX cost of its neighbors towards the sink. The ETX value of a link indicates the required number of transmissions for successful packet reception at the receiver. The ETX value of a link is defined as follows:

$$\mathrm{ETX} = \frac{1}{p \times q} \tag{1}$$

where p and q are the probabilities of forward and backward packet reception over that link, respectively.

Thus, ETX is affected by the link loss ratio, the difference between forward and backward reception rates, and the interference level of successive links (i.e., intra-path interference)[10].

At the beginning of the initialization phase, each node broadcasts a fixed number of control packets and records the number of successfully received packets from its neighbors. Accordingly, each node fills the first two columns of its neighborhood table, i.e., p and q, as demonstrated in Fig. 1. In the second step of this phase, the sink node sets its cost to zero and broadcasts this cost to its neighbors. When a node receives a cost-included packet, it records the retrieved cost as the accumulated ETX cost (i.e., accETX) of the respective neighboring node. For example, when node i receives a broadcast packet from node j, it saves the cost included in this packet as the accumulated ETX cost of node j to the sink. Then, node i calculates its own accumulated ETX value to the sink node as follows:

$$\mathrm{accETX}_i = \mathrm{accETX}_j + \frac{1}{p_{ij} q_{ij}} \tag{2}$$

Neighbor's Information / Neighbors	p	q	accETX	resBatt	interferenceLevel(Σq_i)	Hop Counts
1st Neighbor						
2nd Neighbor						
*n*th Neighbor						

Fig. 1 Neighborhood table

Node i should broadcast the newly calculated accumulated ETX cost if it is lower than the current cost of node i towards the sink. In fact, whenever a node receives a broadcast packet from one of its neighbors, it should calculate its accumulated ETX cost through that node and broadcast that value if it is lower than its current ETX cost towards the sink.

In addition to the initialization phase, the cost update process should also be performed during normal network operation whenever a node finds a new transmission cost to the sink.

2.2 Route discovery and establishment phase

Whenever an event is detected, the route discovery and establishment phase is triggered to establish multiple paths from the event detection area towards the sink. The proposed multipath routing protocol uses several variables, which will be preserved in the neighborhood table and may be updated during the route discovery and data transmission.

The source node starts the route discovery by transmitting a route request packet (*Route_request*) towards the sink node. Whenever a node receives a *Route_request* packet, it computes the transmission cost for the neighboring nodes which are not included in any path from the current source to the sink. This limitation avoids nodes being shared on different paths for the same source node. Then, it forwards the *Route_request* packet to the neighboring node with minimum cost. Node i computes the transmission cost through node j as follows:

$$\text{Cost}_{i,j} = \left(\text{accETX}_j + \frac{1}{p_{i,j}q_{i,j}}\right) \bullet \left(\frac{1}{\text{resBatt}_j}\right) \bullet (1 + \text{interferenceLevel}_j) \quad (3)$$

In Eq. (3), accETX_j is the ETX cost of node j to the sink, which is contained in the neighborhood table of node i. $p_{i,j}$ and $q_{i,j}$ are the forward and backward packet reception rates between node i and node j, respectively. resBatt_j is the remaining battery level of node j expressed in percentage. $\text{interferenceLevel}_j$ is the maximum interference level that node j has experienced.

Whenever a node forwards the *Route_request* packet, it records the node ID of the next-hop node in its routing table. Furthermore, in order to form the reverse path, each node records the ID of the node from which the *Route_request* packet has been received. Since a node cannot be shared among different paths of the same source node (i.e., same event ID), the nodes participating in the route discovery and establishment phase should record a path membership variable for each source node that tries to establish a path through that node. Accordingly, whenever a node receives a *Route_request* packet, it defines a *pathMembership* variable for the event ID included in the *Route_request* packet and sets that variable to "1". This is to indicate that the node is a member of a path belonging to a specific event. Whenever a node forwards the *Route_request* packet, its neighbors overhear this packet and become aware of the current path membership status of this node. However, due to link unreliability, some neighbors may not overhear the *Route_request* packet. Therefore, it is possible that a node receives a *Route_request* packet from a source node for which this node is already a path member. When a node receives such a duplicate *Route_request* packet, it just informs the sender node about its current path membership status. Afterwards, the sender node tries to forward the *Route_request* packet to another qualified node (according to the cost function (3)); otherwise, it notifies its predecessor about the failure in delivering the *Route_request* packet. This backpressure mechanism continues until a node finds another qualified neighbor towards the sink or the *Route_request* packet reaches the source node. Receiving a *Route_request* packet at the source node indicates that the algorithm cannot establish another node-disjoint path. Notice that to forward the *Route_request* packet at each hop, the automatic repeat request (ARQ) mechanism is used to ensure packet delivery.

When the sink receives a *Route_request* packet, it replies by transmitting a *Route_reply* packet along the reverse path (as demonstrated in Algorithm 3, the sink node may first check the data reception rate from the active paths before replying to a *Route_request* packet). Whenever a node receives a *Route_reply* packet, it sets the corresponding *pathMembership* variable to "2" to indicate an active path passing through this node. In addition, when this node forwards the *Route_reply* packet, its neighbors overhear this packet and become aware of the current path membership status of this node.

When the *Route_reply* packet moves from the sink to the source, some nodes overhear this packet. For example, in Fig. 2a, node *F* overhears this packet from node *E* and node *J*. Whenever a node overhears a *Route_reply* packet, it should update its interferenceLevel variable. To this aim, it adds the reverse packet reception rate (i.e., q) of the node from which the *Route_reply* packet has been overheard to its interferenceLevel variable. In Fig. 2a, node *F* retrieves the values of $q_{F,E}$ and $q_{F,J}$ from its neighborhood table and adds them to the current value of its interferenceLevel variable. In addition, the new interference level should be broadcasted to notify neighboring nodes. Since we have used S-MAC[29] as the underlying MAC layer protocol to evaluate our proposed multipath routing protocol, we used the SYNC part of the frame (a frame includes a full cycle of listen and sleep durations) to broadcast the updated interference values. Forwarding of *Route_reply* packets continues until this packet reaches the source node. Reception of a *Route_reply* packet by the source node confirms a new path establishment.

In order to reduce the end-to-end latency, LIEMRO starts packet transmission immediately after the first path is established. Concurrent with data transmission over the first path, the source node sends a new *Route_request* packet to initiate another route discovery and establishment phase to establish the second path. Whenever the sink receives the *Route_request* packet corresponding to the second path, it immediately responds by sending a *Route_reply* packet along the reverse path. Figure 2b shows how the *Route_reply* packet moves from the sink to the source while data packets are being transmitted over the first path. This figure also demonstrates how the nodes update their interference level as they overhear the *Route_reply* packet. When the second path is established, the source node distributes its data packets over the first and second paths using the load balancing algorithm. In addition, the source node sends a *Route_request* packet to start the route discovery and establishment phase for the third path. However, upon receiving this packet at the sink node, it does not send a *Route_reply* packet immediately. Instead, it first investigates if receiving data packets through two paths has resulted in higher performance. The metric used at the sink node to measure the performance improvement is the Received Packets Throughput (RPT). This metric is defined as the packet reception rate at the sink node. Notice that RPT should be computed for the data packets of each source node (i.e., event ID) separately. If receiving data packets through two paths results in higher RPT compared to receiving data packets through one path, the sink node sends a positive feedback message over the second path and sends a *Route_reply* packet over the third path. If receiving data packets through two paths

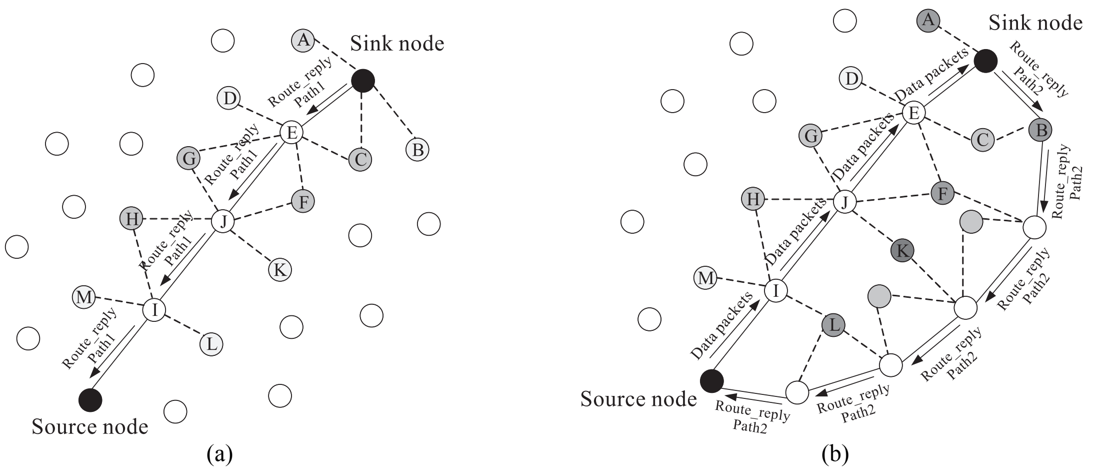


Fig. 2 (a) Updating interference level during the construction of the first path. The various gray levels indicate the amount of experienced interference. (b) Updating interference level during the construction of the second path.

results in lower RPT, the sink node merely sends a negative feedback message over the second path. When the source node receives the *Route_reply* packet over the third path, it redistributes its traffic over the three established paths. In addition, the source node starts the route discovery and establishment process for the fourth path. In contrast, if the source node receives a negative feedback message, it disables the second path and sends its traffic through the first path.

The route discovery and establishment phase can be summarized as follows: The *Route_request* packets corresponding to the first, second, and third paths are transmitted without the need to receive any feedback message from the sink node. Before transmitting the fourth and subsequent *Route_request* packets, the source node should first receive a positive feedback message from the sink. Suppose the n-th path ($n \geqslant 2$) is established and data packets are being transmitted through n paths. Whenever the sink node receives the *Route_ request* packet for the (n+1)-th path, it compares the RPT of n paths with the RPT of $n-1$ paths to decide if transmitting data packets through n paths results in higher performance compared to transmitting data packets through $n-1$ paths. If the performance is improved, the sink node sends a positive feedback on the n-th path and sends a *Route_reply* packet on the (n+1)-th path. Otherwise, if the performance is reduced, the sink node sends a negative feedback message through the n-th path. If the source node receives a positive feedback plus the *Route_reply* packet, it redistributes traffic over the $n+1$ established paths. Additionally, the source node initiates the route discovery and establishment phase for (n+2)-th path. Otherwise, if the source node receives a negative feedback, then it disables the n-th path and redistributes traffic over the $n-1$ remaining paths.

No specific packet type is defined to transmit the feedback message from the sink to the source; instead, an additional field is added to the ACK packets. Therefore, since ARQ is assumed as the underlying error recovery protocol, the source node receives the feedback message after multi-hop propagation of ACK packets from the sink to the source.

Whenever it is not possible to establish a new node-disjoint path (say the n-th path), *Route_request* packet will be backpressured towards the source node. In this case, the source node sends a *Feedback_request* packet to the sink through the last established path, i.e., the ($n-1$)-th path. Upon receiving this packet at the sink, it compares the RPT of $n-1$ paths with the RPT of $n-2$ paths and sends a feedback message accordingly.

2.3 Route maintenance phase

Since in event-driven applications data packets should be transmitted after event detection, the period of data transmission and, therefore the chance of node failure is lower than that of monitoring applications in which there is frequent data transmission from all the nodes to the sink. However, due to the dynamics of wireless networks, the potential effects of link failure should be considered. According to the packet transmission mechanism at the data link layer, if a node on an active path does not receive the ACK packet from the next-hop node after k efforts, it notifies the network layer about the link failure. Assuming at least two efforts for a perfect link (i.e., $p_{i,j}$=1), k can be calculated using the geometric distribution:

$$k = \frac{1}{p_{i,j}} + 1 \tag{4}$$

After link failure detection, an error message will be transmitted to the source node through the reverse path. Upon reception of this message at the source node, it disables the path from which this message has been received and redistributes traffic over the remaining paths. Furthermore, to prevent performance degradation, the source node initiates a new route discovery and establishment process.

The implementation details are given in the algorithms for the source node, intermediate nodes, and sink node in Algorithms 1, 2, and 3.

Algorithm 1 Source Node's Algorithm

```
1. If (no Route_request packet is sent before)
2. {
3.    For (all of the source node's neighbors)
4.       Calculate $cost_{i,j}$ for neighbor $j$;
5.    Add the event ID to the Route_request packet;
6.    // event ID is the ID of the current source node
7.    Send the Route_request packet to the node with
      minimum $cost_{i,j}$;
8. }
9. If (a Route_reply packet is received from the first or
    second path)
10. {
11.   Transmit data packets over the established path(s)
      using the load balancing algorithm;
12.   ID = event ID;
13.   For (all of the source node's neighbors)
14.     If (node j is not a member of the path with the
        identification number equal to ID)
15.       Calculate cost_{i,j} for neighbor j;
16.   If (a good neighbor is found)
17.     {
18.     Add the event ID to the Route_request packet;
19.     Send the Route_request packet to the node with
        minimum cost_{i,j};
20.     }
21.   Else
22.     If (The Route_reply packet is received from the
        second path)
23.       Send a Feedback_request packet to the sink
          through the second path;
24. }

25. If ((a Route_reply packet is received from the n-th
    path)&&($n \geqslant 3$))
26.   Transmit data packets over the n established
      paths using the load balancing algorithm;

27. If ((a positive feedback is received from the n-th
    established path)&&($n \geqslant 2$))
28. {
29.   ID = event ID;
30.   For (all of the source node's neighbors)
31.     If (node j is not a member of the path with the
        identification number equal to ID)
32.       Calculate cost_{i,j} for neighbor j;
33.   If (a good neighbor is found)
34.     {
35.     Add the event ID to the Route_request packet;
36.     Send the Route_request packet to the node
        with minimum cost_{i,j};
37.     }
38.   Else
39.     Send a Feedback_request packet to the sink
        through the last established path;
40.     // the last constructed path is the (n+1)-th path
41. }

42. If (a negetive feedback is received from the n-th
    path)
43. {
44.   Disable the n-th path;
45.   Transmit data packets over the n−1 remaining
      paths;
46. }

47. If (a Route_request packet is received from the
    n-th path)
48. // due to the backpressure mechanism
49.   Send a Feedback_request packet to the sink
      through the (n−1)-th path;

50. If (an error message is received) // when a path
    failure occurs
51. {
52.   Disable the path from which this message has
      been received;
53.   Redistribute data packets over the remaining
      paths;
54.   Initiate a new route discovery and establishment
      phase;
55. }
```

Algorithm 2 Intermediate Nodes' Algorithm

```
1.  If (a Route_request packet is received)
2.  {
3.    ID = event ID included in the Route_request
      packet;
4.    If (this node is a member of the path with the iden-
      tification number equal to ID)
5.      Backpressure the received Route_request packet;
6.    Else
7.      {
8.      For (all of this node's neighbors)
9.        If (node j is not a member of the path with the
          identification number equal to ID)
10.         Calculate cost_{i,j} for neighbor j;
11.     If (a good neighbor is found)
12.       {
13.       pathMembership variable for the current event
```

```
        ID = 1;
14.     Send the received Route_request packet to the
        node with minimum $cost_{i,j}$;
15.   }
16.   Else
17.     Backpressure the received Route_request
        packet;
18.   }
19. }

20. If (a Route_reply packet is received)
21. {
22.   Send the received Route_reply packet in the re-
      verse path towards the source node;
23.   pathMembership variable for the current event
      ID = 2;
24.   Broadcast the newly updated path membership
      tuple;
25. }

26. If (a Route_reply packet is overheard from node j)
27. {
28.   Refer to the neighborhood table and extract the
      backward packet reception rate to node j;
29.   Add the extracted value to the interferenceLevel
      variable;
30. }

31. If (a link failure occurred during the data transmis-
    sion)
32. // when a path failure occurs
33.   Send an error massage in the reverse path to-
      wards the source node;

34. If (a backpressure Route_request packet is re-
    ceived)
35. {
36.   ID = event ID included in the Route_request
      packet;
37.   For (all of this node's neighbors except the node
      from which the Route_request packet has been
      received)
38.     If (node j is not a member of the path, with the
        identification number equal to ID)
39.       Calculate $cost_{i,j}$ for neighbor j;
40.   If (a good neighbor is found)
41.     Send the received Route_request packet to the
        node with minimum $cost_{i,j}$;
42.   Else
43.   {
44.     Backpressure the received Route_request
        packet;
45.     pathMembership variable for the current event
        ID = 0;
46.   }
47. }
```

Algorithm 3 Sink Node's Algorithm

```
1. If (a Route_request packet is received from the first
   or second path)
2. {
3.   Send a Route_reply packet over the first or sec-
     ond path towards the source node;
4. }

5. If ((a Route_request packet is received from the n-
   th path)&&( $n \geqslant 3$ ))
6. {
7.   If ((RPT of $n$–2 paths)<=(RPT of $n$–1 paths))
8.   {
9.     Send a positive feedback over the ($n$−1)-th
       path towards the source node;
10.    Send a Route_reply packet over the $n$-th path
       towards the source node;
11.  }
12.  Else
13.    Send a negative feedback over the ($n$−1)-th
       path towards the source node;
14. }

15. If (a Feedback_request packet is received from
    the $n$-th path)
16. {
17.   If ((RPT of $n$–1 paths)<=(RPT of $n$ paths))
18.     Send a positive feedback over the $n$-th path
        towards the source node;
19.   Else
20.     Send a negative feedback over the $n$-th path
        towards the source node;
21. }
```

3 Load Balancing Algorithm

Since different paths exhibit dissimilarities in their link qualities, number of hops, residual energy, and interference level, they have different transmission capacities. Hence, it is important to calculate the optimal traffic rate for each path according to their relative quality.

3.1 Prerequisites of load balancing

During forwarding of the *Route_reply* packet from the sink to the source, some information regarding the traversed path can be included in this packet. This embedded information includes: 1) the experienced interference level of the path, 2) the accumulated residual battery level of the nodes, and 3) the path quality based on the ETX metric. A higher interference level results in more energy consumption and longer latency along the path mainly due to issues such as severe contention for medium access and higher packet corruption rates. In addition, the residual battery level of the nodes is used to provide more balanced energy consumption in the network. The importance of ETX is that: the path with the lower ETX incurs lower packet corruption rates and lower energy consumption. The information brought to the source node by the *Route_reply* packets can be used to estimate the relative quality of each path. However, since this information may not be accurate, there should be another update mechanism. For example, consider the situation in which the source node has just received a *Route_reply* packet indicating the establishment of the third path. Therefore, the source node distributes its traffic over the three established paths. However, the load balancing algorithm may be using inaccurate values of each path's interference level. In fact, when the third path's *Route_reply* packet moves from the sink to the source, intermediate nodes along the first and second path may update their experienced interference level. In order to enable the source node to acquire the updated path information, ACK packets are used to propagate this information towards the source. Therefore, after a short period of data transmission, the load balancing algorithm stabilizes. Updating of the path quality information occurs whenever a node sees a more than 10% change in the interference level, battery level, or ETX, compared to the last propagated values.

3.2 Load balancing mechanism

When the source node receives the path quality information regarding k-th path from node j, it computes the load cost of the k-th path as:

$$p_k = \left(\mathrm{ETX}_j + \frac{1}{p_{\mathrm{src},j} q_{\mathrm{src},j}} \right) \cdot (1 + \mathrm{accIntfr}_k) \cdot \left(\frac{1}{\mathrm{accResBatt}_k} \right) \tag{5}$$

ETX_j is the ETX cost of node j to the sink, $p_{\mathrm{src},j}$ and $q_{\mathrm{src},j}$ are forward and backward packet reception rates between the source node and node j. $\mathrm{accIntfr}_k$ and $\mathrm{accResBatt}_k$ are the accumulated interference level and accumulated residual battery level from the sink to node j along the k-th path. If there exits n paths between the source and sink, the optimal traffic ratio r_k for the k-th path can be obtained as follows:

$$\min(r_1 p_1 = r_2 p_2 = r_3 p_3 = \cdots = r_n p_n)$$

Subject to

$$\sum_{k=1}^{n} r_k = 1 \tag{6}$$

Therefore,

$$r_k = 1 \bigg/ p_k \sum_{f=1}^{n} \frac{1}{p_f} \tag{7}$$

4 Performance Evaluation

In this section, we present a comprehensive evaluation of the proposed multipath routing protocol. First, we introduce the performance evaluation metrics. Then, we describe the applied simulation framework and its parameters. Finally, we analyze and discuss the simulation results.

4.1 Performance parameters

We have designed and evaluated LIEMRO in terms of the following performance metrics:

• **Data Delivery Ratio**: the ratio of successfully received data packets at the sink to the total number of data packets transmitted by the source. Accordingly, this metric reveals the data transmission reliability.

• **Network Lifetime**: the period of time from network initialization until the first node's death. From the network layer perspective, the control packets exchanged for route discovery, establishment, and route maintenance reflect the routing overhead and directly affect the lifetime. Therefore, we used this metric to represent the energy efficiency and overhand of the proposed multipath approach.

• **End-to-End Latency**: measured as the average time between packet transmission from the source and its reception at the sink.

• **End-to-End Throughput**: the total number of data bits received at the sink divided by the transmission duration.

• **Average Queue Length**: the average queue length of the nodes belonging to the active paths. This metric reveals the congestion avoidance capability of the protocol.

4.2 Simulation model

We used the OMNeT++ framework to develop our simulation program. We implemented the link layer model of USC[30] in our simulation framework. Then, we implemented LIEMRO and the medium access control protocol (S-MAC) as two separate modules.

Table 1 demonstrates the default simulation parameters. These parameters are chosen according to the characteristics of the MICA2 mote and data link model of USC[30]. The energy consumptions for different radio operations and sensor sampling are provided in Ref. [31]. In all the simulation scenarios 200 static nodes were placed randomly in a 100 m×100 m area. In order to maximize the distance between source and sink, we placed the sink node at the top right corner and the source node at the down left corner of the area.

Table 1 Simulation settings

	Parameter	Value
Radio	Modulation	FSK
	Output power	0 dBm
	Encoding	NRZ
	Frame	45 bytes
Transmission medium	Path loss exponent	4
	Noise floor	−105 dBm
	PLD_0	55 dB
	D_0	1 m
MAC protocol (S-MAC)	Listening duration	115 ms
	Contention window for SYNC	15 slots
	Sleep duration	500 ms
	Contention window for data	31 slots
Other parameters	Number of nodes	200
	Battery capacity	2400 mAh
	Payload size	29 bytes
	Physical and MAC layer headers	16 bytes
	Area size	100 m×100 m
	Simulation duration	600 s
	Control packet size	18 bytes
	ACK packet size	23 bytes

Existing multipath routing protocols use IEEE 802.11 as their MAC layer protocol. However, since IEEE 802.11 has considerable energy waste in the idle listening mode, it does not meet the demands of wireless sensor networks for energy efficient communications. On the other hand, the periodic sleep/listen schemes used in the MAC protocols for wireless sensor networks significantly affect the network performance[29]. For example, the periodic sleep/listen scheduling introduced by S-MAC[29] divides the time into two parts: listen and sleep. In the listen period, nodes use the four-way handshake mechanism (i.e., RTS/CTS/DATA/ACK) to access the medium and transmit their data packets. The nodes participating in the communication (i.e., sender and receiver node), should stay awake during the sleep period to complete their data transmission. When two nodes communicate, their neighboring nodes should sleep until the start of the next listening period. In high traffic loads, this mechanism worsens the congestion and increases packet loss due to buffer overflow. Implementing S-MAC as the underlying MAC protocol helps to investigate the effects of periodic sleep/listen schemes on the proposed multipath routing protocol.

As mentioned earlier, LIEMRO establishes node-disjoint paths for each detected event. However, when more than one event exists in the environment, some nodes may be shared between the paths of different events, specifically near the sink node. Since this situation results in higher contention for medium access around the shared nodes and affects network performance, we also evaluated LIEMRO under a multiple-event scenario. In this scenario, multiple paths are established for two source nodes (i.e., two events). The additional source node is placed at the center of the area to maximize the effects of inter-path interference.

In the rest of this section, we evaluate LIEMRO with various packet generation intervals and transmission power levels. In each figure, the transmission power is abbreviated as TP. The performance of LIEMRO is compared against a single-path routing protocol which uses ETX and the residual battery level in its path cost function.

4.3 Simulation results

In the following subsections, we provide the simulation results in terms of the metrics described in Section 4.1.

4.3.1 Data delivery ratio

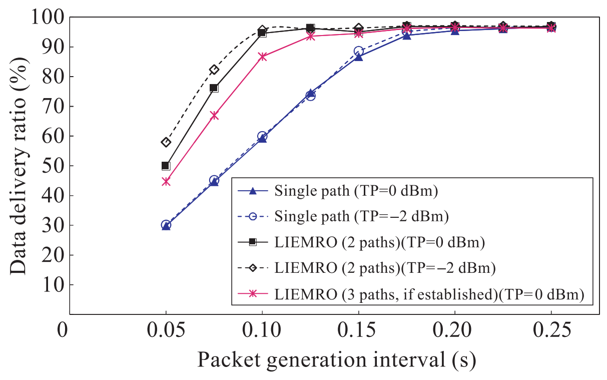
The unreliability of wireless links and the limited buffer capacity of each node highly affects the transmission reliability in wireless sensor networks. Figure 3a shows that at the high traffic condition LIEMRO improves the data reception rate up to 85% compared to single-path routing.

It can be observed from Fig. 3a that extra paths do not necessarily result in higher data reception rates. At high traffic loads, the data reception rate with three paths is about 13% lower than that with two paths. When we forced the algorithm to establish the third path, more inter-path interference was experienced specifically near the source and sink. This observation justifies why it is important to build an efficient number of paths in multipath routing protocols. Consequently, as described in Section 2.2, LIEMRO uses the data reception rate at the sink node to decide if the newly established path should be disabled.

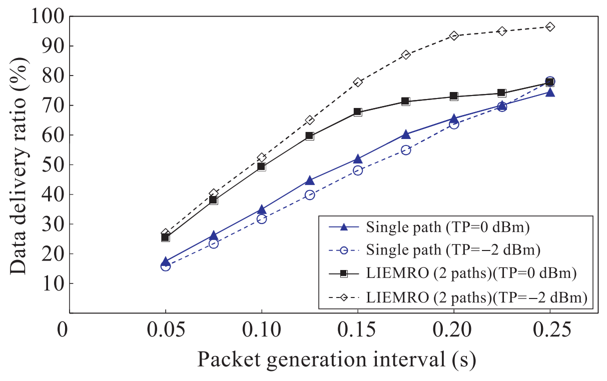
Figure 3b demonstrates that when multiple events coexist in the network, LIEMRO still provides up to nearly 70% improvement relative to the single-path routing protocol under intensive traffic loads. Since the two source nodes are positioned in the way that their established paths share some nodes, a high data rate brings about severe contention for medium access around the shared nodes. This results in lower performance improvement of LIEMRO over the single-path routing protocol. In addition, in both Fig. 3a and Fig. 3b, using LIEMRO with −2 dBm demonstrates higher performance improvement compared to LIEMRO with 0 dBm. This is the result of less interference with this power level.

4.3.2 Lifetime

According to Fig. 4a, transmitting data packets over two paths improves the network lifetime about 25% compared to the single-path approach. At high traffic loads, LIEMRO and the single-path routing protocol both provide a very low lifetime. In fact, as the network traffic increases, contention for medium access intensifies and the packet corruption rate rises. Consequently, in order to provide medium access and recover corrupted packets, the radio should spend a longer time at the active state (i.e., send or receive). This condition

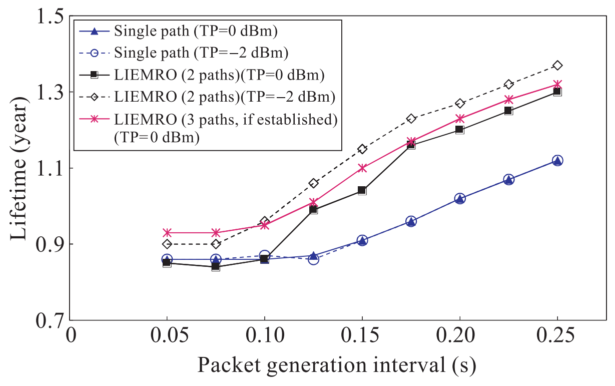


(a) One source nodes

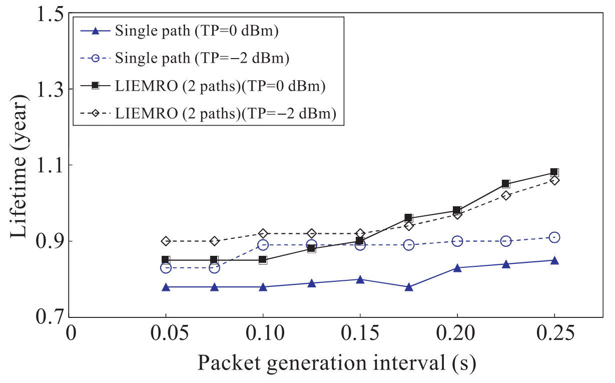


(b) Two source nodes

Fig. 3 Data delivery ratio comparison for the single-path routing and LIEMRO versus different traffic loads



(a) One source nodes



(b) Two source nodes

Fig. 4 Lifetime comparison for the single-path routing and LIEMRO versus different traffic loads

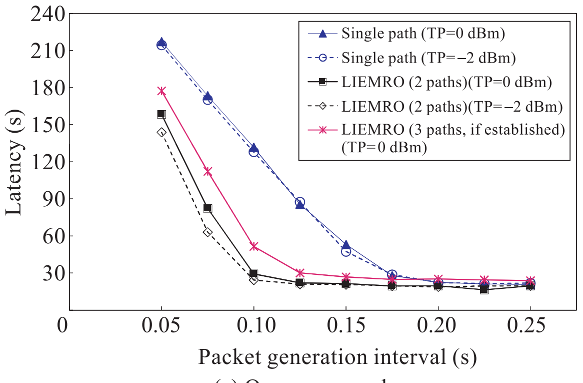
results in a higher duty cycle with a shorter network lifetime. From the network layer standpoint, at the heavy traffic conditions each path created by LIEMRO is used at its saturated level. Therefore, the demonstrated network lifetime defined by the first node's death is almost the same for the single-path and multipath routing approaches. Another observation that can be drawn from Fig. 4a is that three paths result in higher network lifetime compared to two paths. This can be explained as the effect of more balanced energy consumption. However, as explained in Section 4.3.1, when the third path is established, the data reception rate at the sink node degrades and LIEMRO disables this path. As a result, although network lifetime may be increased through establishing additional paths, it may also result in reduced throughput due to increased inter-path interference. Using the data delivery ratio to decide about the efficiency of a newly established path not only improves end-to-end throughput from the source to the sink, but also improves the network lifetime compared to the single-path approach.

Figure 4b demonstrates the behavior of LIEMRO in the multiple-event scenario. As can be perceived, LIEMRO still surpasses the lifetime of the single-path approach. However, due to higher contention for medium access and increases in the number of packet retransmissions, the performance improvement is not as good as that of the single-event case.

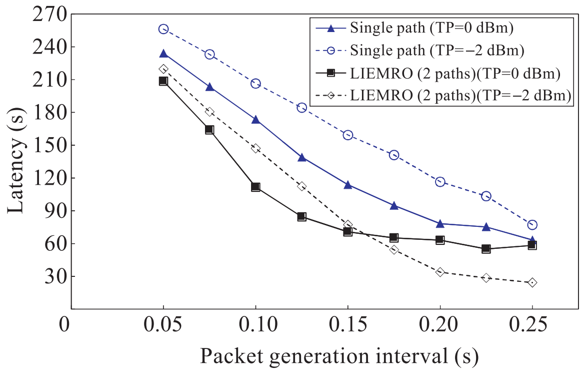
As stated in Section 4.1, network lifetime reflects the energy efficiency and overhead of the route construction and data transmission phases. Based on the results presented in Fig. 4, although multipath routing requires more control packets to discover and construct multiple paths, efficient distribution of network traffic over several paths offsets the overhead imposed by the control packets.

4.3.3 End-to-end latency

The capability of LIEMRO to meet the delay requirements of event-driven applications is shown in Fig. 5a and Fig. 5b. As the offered load grows, the average queue length at each node increases and data packets suffer longer queuing delays. From the MAC layer point of view, when a node overhears an RTS/CTS packet it should sleep until the next frame. Therefore, the sleep delay encountered at each hop increments as the number of interfering nodes for that hop scales up. Additionally, the required number of retransmissions



(a) One source nodes



(b) Two source nodes

Fig. 5 Average end-to-end latency comparison for the single-path routing and LIEMRO versus different traffic loads

increases as the inter-path interference intensifies. According to Fig. 5a, LIEMRO reduces the end-to-end latency of the single-path routing protocol more than 60% for intensive traffic loads. The most significant benefit is about 80%, which can be observed at the packet generation interval of around 0.1 second. At low traffic loads, the queuing delay is almost negligible and the observed delay is mainly due to the inherent delays of the MAC protocol such as the sleep delay, transmission delay, and carrier sense delay. Therefore, the latencies of LIEMRO and the single-path routing protocol are almost the same for the light traffic load conditions.

Figure 5a confirms that establishing extra paths does not necessarily result in latency improvement. When three paths are in use, the end-to-end latency increases about 36% compared to the case where two paths exist. Nevertheless, since LIEMRO disables the third path, it improves the end-to-end latency.

Figure 5b represents the latency of LIEMRO for the multiple-event scenario. LIEMRO provides about 40% lower latency than the single-path approach.

4.3.4 End-to-end throughput

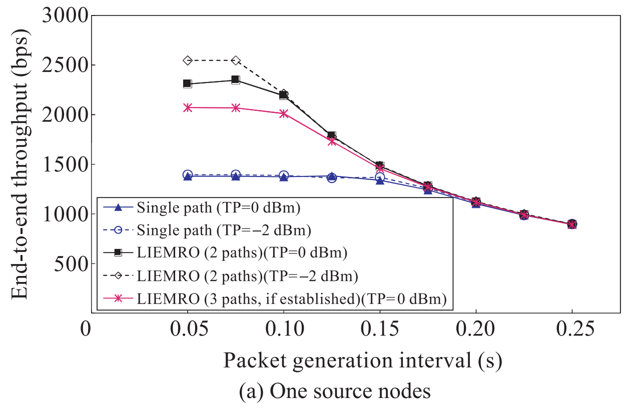
As can be seen from Fig. 6a, in the intensive traffic loads LIEMRO provides about 80% improvement in the end-to-end throughout relative to the single-path routing protocol. Because LIEMRO establishes multiple interference-minimized paths, it increases per-path capacity. In addition, the load balancing algorithm determines the best traffic rate of each path based on its relative quality. With respect to Fig. 6a, the end-to-end throughput increases as the traffic rate intensifies until the packet generation interval reaches around 0.075 s. In fact, from this point onwards, further increases in the packet generation rate cannot result in throughput improvement because the active paths are used at their saturated rate.

Figure 6b confirms that in the multiple-event scenario LIEMRO provides up to 40% improvement relative to the single-path routing protocol. Nevertheless, the multiple-event scenario shows a lower slope versus packet generation interval. This is due to the packet generation at two source nodes, which keeps the network in a high traffic condition for a wider range of packet generation intervals.

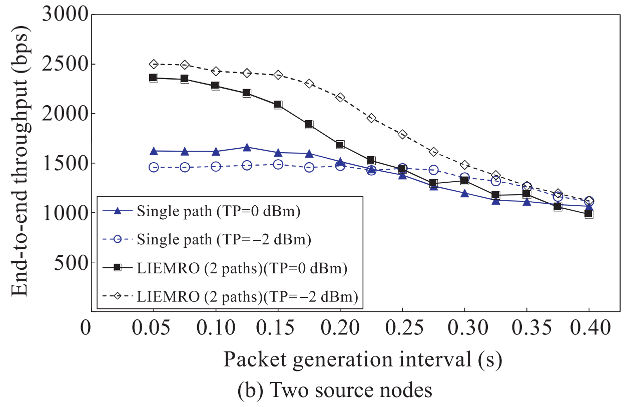
4.3.5 Average queue length

Since sensor nodes have limited memory capacity, buffer overflow results in significant packet loss in congested areas. Figure 7a compares the average queue length for LIEMRO and the single-path approach. As the figure shows, at high traffic loads LIEMRO reduces the average queue length per node by roughly 64% compared to the single-path routing protocol. The benefit is less obvious as the generated traffic increases and multiple paths are used at their saturated levels. In addition, Fig. 7a shows that using three paths instead of two paths results in longer queue lengths, which is mainly due to two reasons. First, intensified inter-path interference causes increased packet loss ratio and severe contention for medium access, which in turn results in higher queuing delay. Second, as LIEMRO tries to discover interference-minimized paths, the third path includes more hops compared to the first and second paths. This issue results in more packets accumulated along the path.

According to Fig. 7b, the average queue length per

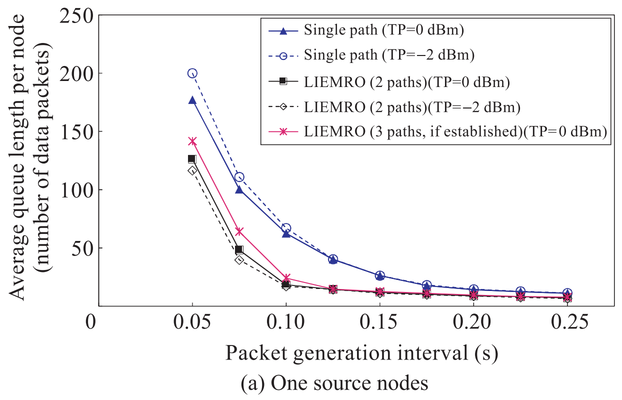


(a) One source nodes

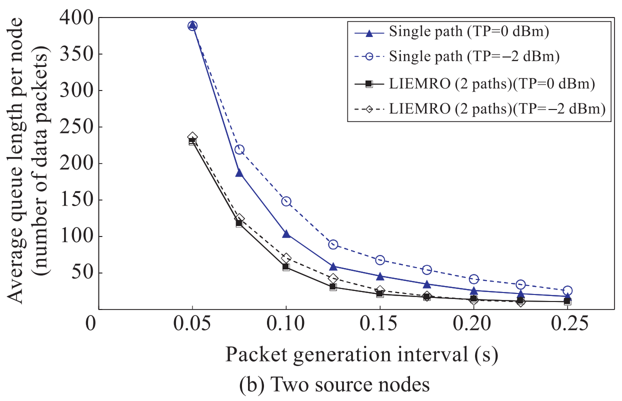


(b) Two source nodes

Fig. 6 End-to-end throughput comparison for the single-path routing and LIEMRO versus different traffic loads



(a) One source nodes



(b) Two source nodes

Fig. 7 Average queue length comparison for the single-path routing and LIEMRO versus different traffic loads

node in the multiple-event scenario is about 50% higher than the single-event scenario for both protocols. The multiple-event scenario demonstrates higher packet congestion due to two issues. First, some nodes are shared between multiple paths belonging to different source nodes. Second, there are two source nodes both generating data packets. Still, LIEMRO reduces the average queue length per node more than 50% compared to the single-path routing protocol.

5 Conclusions

In this paper, we introduce a multipath routing protocol (LIEMRO) to improve QoS demands of event-driven applications in wireless sensor networks. Through three main mechanisms, LIEMRO benefits from data transmission over multiple paths: First, motivated by the destructive effects of inter-path interference on the performance of multipath routing, LIEMRO tries to establish multiple node-disjoint interference-minimized paths from each source node to the sink. Additionally, it reduces inter-path interference between the paths originating from different source nodes. Second, LIEMRO adjusts the number of established paths according to the packet delivery ratio perceived at the sink node. Third, through an efficient load balancing algorithm, the source node distributes its traffic over multiple paths based on each path's relative quality.

We evaluated LIEMRO and compared its performance with the single-path routing protocol that uses ETX and the residual battery life in its cost function. In order to achieve more accurate results, we implemented S-MAC as the underlying MAC protocol. Simulation results demonstrate significant performance improvement over the single-path approach in terms of data delivery ratio, latency, end-to-end throughput, and lifetime, which are the essential QoS parameters of event-driven applications.

According to the simulation results, when multiple source nodes coexist in the sensor field and try to establish several paths, some nodes may be shared by the paths belonging to different source nodes. In this situation, the efficacy of the multipath routing protocol highly degrades greatly due to the intense contention for medium access, buffer overflow, and high loss ratio. Accordingly, multiple-event scenarios demand for an interference-minimized m-to-n multipath routing protocol that establishes several paths from m source nodes to n sink nodes. A joint MAC and power control mechanism can be applied to reduce inter-path and intra-path interference.

References

[1] Yick J, Mukherjee B, Ghosal D. Wireless sensor network survey. *Computer Networks*, 2008, **52**(12): 2292-2330.

[2] Li Yanjun, Chen Chung-Shue, Song Ye-Qiong, et al. Real-time QoS support in wireless sensor networks: A survey. In: Proceedings of the 7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems (FET '07). Toulouse, France, 2007: 373-380.

[3] Son D, Krishnamachari B, Heidemann J. Experimental study of concurrent transmission in wireless sensor networks. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06). Boulder, Colorado, USA, 2006: 237-250.

[4] Akkaya K, Younis M. A survey on routing protocols for wireless sensor networks. *Ad Hoc Network Journal*, 2005, **3**(3): 325-349.

[5] Li Jinyang, Blake C, Couto D S J D, et al. Capacity of ad hoc wireless networks. In: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom '01). Rome, Italy, 2001: 61-69.

[6] Tsai J, Moors T. A review of multipath routing protocols: From wireless ad hoc to mesh networks. In: Proceedings of the First ACoRN Early Career Researcher Workshop on Wireless Multihop Networking. Sydney, Australia, 2006: 17-18.

[7] Lou Wenjing, Liu Wei, Zhang Yanchao. Performance optimization using multipath routing in mobile ad hoc and wireless sensor networks. *Combinatorial Optimization in Communication Networks*, 2006, **18**: 117-146.

[8] Wang Ying-Hong, Lin Hung-Zu, Chang Shu-Min. Interference on multipath QoS routing for ad hoc wireless network. In: Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW '04). Hachioji, Tokyo, Japan, 2004:104-109.

[9] Jones E, Karsten M, Ward P. Multipath load balancing in multi-hop wireless networks. In: Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob '05). Montreal, Canada, 2005: 158-166.

[10] Couto D S J D, Aguayo D, Bicket J, et al. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 2005, **11**(4): 419-434.

[11] Radi M, Dezfouli B, Razak S A, et al. LIEMRO: A low-

interference energy-efficient multipath routing protocol for improving QoS in event-based wireless sensor networks. In: Proceedings of the 4th International Conference on Sensor Technologies and Applications (SENSORCOMM ’10). Venice, Italy, 2010: 551-557.

[12] Tarique M, Tepe K E, Adibi S, et al. Survey of multipath routing protocols for mobile ad hoc networks. *Journal of Network and Computer Applications*, 2009, **32**(6): 1125-1143.

[13] Intanagonwiwat C, Govindan R, Estrin D. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom’00). Boston, Massachusetts, USA, 2000: 56-67.

[14] Ganesan D, Govindan R, Shenker S, et al. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communication Review*, 2001, **5**(4): 11-25.

[15] Ben-Othman J, Yahya B. Energy efficient and QoS based routing protocol for wireless sensor networks. *Journal of Parallel and Distributed Computing*, 2010, **70**(8): 849-857.

[16] Huang Xiaoxia, Fang Yuguang. Multiconstrained QoS multipath routing in wireless sensor networks. *Journal of Wireless Networks*, 2007, **14**(4): 465-478.

[17] Bagula A, Mazandu K. Energy constrained multipath routing in wireless sensor networks. In: Proceeding of the 5th International Conference on Ubiquitous Intelligence and Computing. Oslo, Norway, 2008: 453-467.

[18] Lou Wenjing. An efficient N-to-1 multipath routing protocol in wireless sensor networks. In: Proceedings of the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor System (MASS ’05). Washington D.C., USA, 2005: 672-680.

[19] Hurni P, Braun T. Energy-efficient multi-path routing in wireless sensor eetworks. In: Proceedings of the 7th International Conference on Ad-hoc, Mobile and Wireless Networks (ADHOC-NOW’08). Sophia Antipolis, France, 2008: 72-85.

[20] Marina M K, Das S R. On-demand multipath distance vector routing in ad hoc networks. In: Proceedings of the 9th International Conference on Network Protocols. Riverside, CA, USA, 2001: 14-23.

[21] Lu Yeming, Wong V W S. An energy-efficient multipath routing protocol for wireless sensor networks. *International Journal of Communication Systems*, 2007, **20**(7): 747-766.

[22] Teo J Y, Ha Y, Tham C K. Interference-minimized multipath routing with congestion control in wireless sensor network for high-rate streaming. *IEEE Transactions on Mobile Computing Mobile Computing*, 2008, **7**(9): 1124-1137.

[23] Woo A, Tong T, Culler D. Taming the underlying challenges of reliable multihop routing in sensor networks. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SENSYS’03). Los Angeles, CA, USA, 2003: 14-27.

[24] Roy S, Bandyopadhyay S, Ueda T, et al. Multipath routing in ad hoc wireless networks with Omni directional and directional antenna: A comparative study. In: Proceedings of the 4th International Workshop on Distributed Computing, Mobile and Wireless Computing (IWDC '02). Calcutta, India, 2002: 184-191.

[25] Tarn W H, Tseng Y C. Joint multi-channel link layer and multi-path routing design for wireless mesh networks. In: Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM ’07). Anchorage, AK, USA, 2007: 2081-2089.

[26] Wang Zijian, Bulut E, Szymanski B K. Energy efficient collision aware multipath routing for wireless sensor networks. In: Proceedings of the 2009 IEEE International Conference on Communications (ICC’09). Dresden, Germany, 2009: 91-95.

[27] Wei Wei, Zakhor A. Interference aware multipath selection for video streaming in wireless ad hoc networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 2009, **19**(2): 165-178.

[28] Jain K, Padhye J, Padmanabhan V N, et al. Impact of interference on multi-hop wireless network performance. *Wireless Networks*, 2005, **11**(4): 471-487.

[29] Ye Wei, Heidemann J, Estrin D. An energy-efficient MAC protocol for wireless sensor networks. In: Proceedings of the 21th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM’02). New York, NY, USA, 2002: 1567-1576.

[30] Zamalloa M Z, Krishnamachari B. An analysis of unreliability and asymmetry in low-power wireless links. *ACM Transactions on Sensor Networks*, 2007, **3**(2): 7.

[31] Polastre J, Hill J, Culler D. Versatile low power media access for wireless sensor networks. In: Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys ’04). Baltimore, MD, USA, 2004: 95-107.