Routing in Intermittently Connected Sensor Networks

Lu Su*, Changlei Liu†, Hui Song‡ and Guohong Cao†

*Dept. of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL, 61801
Email: lusu2@uiuc.edu

‡Dept. of Computer Science
Frostburg State University
Frostburg, MD, 21532
Email: hsong@frostburg.edu

†Dept. of Computer Science & Engineering
The Pennsylvania State University
State College, PA, 16802
Email: {chaliu,gcao}@cse.psu.edu

***Abstract*—To prolong the lifetime of sensor networks, various scheduling schemes have been designed to reduce the number of active sensors. However, some scheduling strategies, such as partial coverage scheduling and target coverage scheduling, may result in disconnected network topologies, due to the low density of the active nodes. In such cases, traditional routing algorithms cannot be applied, and the shortest path discovered by these algorithms may not have the minimum packet delivery latency. In this paper, we address the problem of finding minimum latency routes in intermittently connected sensor networks by proposing an on-demand minimum latency (ODML) routing algorithm. Since on-demand routing algorithm does not work well when the source and destination frequently communicate with each other, we propose two proactive minimum latency routing algorithms: optimal-PML and quick-PML. Theoretical analysis and simulation results show that (1) ODML can effectively identify minimum latency routes which have much smaller latency than the shortest path, and (2) optimal-PML can minimize the routing message overhead and quick-PML can significantly reduce the route acquisition delay.**

I. Introduction

Wireless sensor networks have been envisioned to be useful in many military and civilian applications such as battlefield surveillance, target tracking, habitat monitoring, etc. Since sensor nodes are generally powered by battery, techniques to prolong the network lifetime have become the recent research focus. A variety of energy conservation strategies have been proposed. Among them, a frequently used mechanism is to deploy more sensors than required, and schedule the activity of each sensor node such that sensors perform the given mission in turn and at any time, only a small number of sensors are active to meet the coverage requirement of the mission [1], [2], [3], [4], [5], [6].

With scheduling, the number of active nodes is significantly reduced and thus the network lifetime is prolonged. However, most scheduling schemes are application driven, and their priority is to achieve the desired sensing coverage. As a result, some scheduling strategies such as partial coverage scheduling [3], [4] and target coverage scheduling [5], [6], may result in a sparse distribution of active nodes. Consequently, the network may not be connected at some instant due to the low node density.

This work was supported in part by the National Science Foundation under grant CNS-0519460.

When the network is not fully connected, there will be problems in sending the sensing data to the sink since traditional routing algorithms such as AODV [7] and DSR [8] may fail. Furthermore, in intermittently connected sensor networks, the metric of hop number used in many traditional routing schemes may not work well because the shortest path may not be able to achieve minimum end-to-end packet delivery delay, which is crucial for many applications such as military surveillance and forest fire alarms.

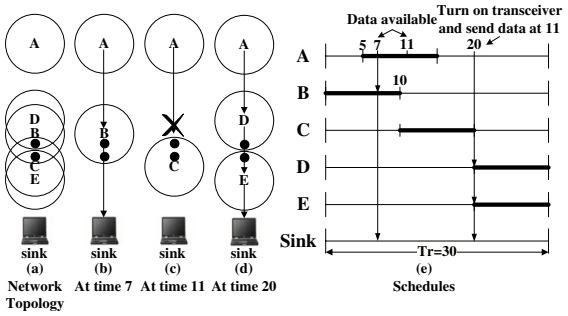


Fig. 1. An example of intermittently connected network, where nodes B,C,D,E monitor the targets in turn, and A wants to send packets to the sink

Example 1.1: Consider a target coverage scheduling example shown in Fig. 1(a). The mission of the sensor network is to continuously monitor (cover) two targets (black points) in the deployed region. Initially, the targets are sensed by multiple nodes B, C, D, E (circles denote the sensing range of the nodes). To save power, the scheduling algorithm proposed in [5] is used to only keep part of the nodes active. As illustrated in Fig. 1(e), node B is active from time 0 to 10, node C is active from time 10 to 20, and both D and E are active from time 20 to 30. Suppose at time 7, node A collects some data and wants to send the data to the sink. If B is within the communication range of A, A can immediately send the data to B which subsequently forwards the data to the sink as shown in Fig. 1(b). However, if A's data is available at time 11, it cannot send the data to B which has been scheduled to sleep. Since only node C is active at this time, and it is outside the communication range of A (Fig. 1(c)), A's data

cannot be forwarded to the sink.

To address this problem, the source node has to buffer the data temporarily and send the data to its neighbor when it wakes up. *Note that the source node can be in sleep and only wake up before sending the data*. As in our example, A buffers the data until D wakes up at time 20. Then, it turns on its transceiver and passes the data to D as shown in Fig. 1(d). The data follows the path $A \rightarrow D \rightarrow E \rightarrow sink$ and arrives at the sink at time 20, thus the *packet delivery latency* is 9 (20-11). On the other hand, if A chooses the shortest path, $A \rightarrow B \rightarrow sink$, it has to wait for B's wakeup in the next round, i.e., time 30, resulting in a much longer delay. Therefore, the end-end packet delivery latency is a better routing metric in intermittently connected sensor networks.

In this paper, we propose an on-demand minimum latency (ODML) routing algorithm to find minimum latency routes in intermittently connected sensor networks. Since on-demand routing algorithm does not work well when the source and destination frequently communicate with each other, we propose two proactive minimum latency routing algorithms: optimal-PML and quick-PML. Theoretical analysis and simulation results show that (1) ODML can effectively identify minimum latency routes which have much smaller latency than the shortest path, and (2) optimal-PML can minimize the routing message overhead and quick-PML can significantly reduce the route acquisition delay.

The rest of the paper is organized as follows. In the next section, we introduce the system model and formulate the problem. Section III and Section IV present our on demand routing scheme and proactive routing schemes respectively. Then, we discuss some related issues in Section V, and evaluate the performance of proposed schemes in Section VI. The related work is discussed in Section VII. Section VIII concludes the paper.

II. System Model and Problem Formulation

A. Network and Scheduling Model

Similar to many application-driven scheduling schemes [1-6], we assume the lifetime of a sensor network is divided into *rounds* with equal duration (T_r). For each node, a round consists of a *working period* (T_w) and a *sleep period*. Each round is divided into time slots, and a packet is only transmitted at the start of a time slot. Note that the schemes in this paper can also work well if T_w varies among the nodes.

To simplify the discussion, in all the examples, the sensor nodes are synchronized. Nevertheless, our algorithms do not pose any restriction on the synchronization between neighboring nodes. Each node only needs to know the relative position of its neighbor's working period (i.e., the offset between the start times of their working period), and this can be easily achieved through the periodic exchange of hello messages.

Under an intermittently connected topology, if a node only delivers packets during the working period, the packets may never be sent out, because its working period may not overlap with that of its neighbors. To address this issue, each node still receives packets within the working period but the sending rule is modified as follows: *if a node has packets to forward outside its working period, it wakes up its transceiver and sends packets at the start of the working period of its next hop neighbor*.

In intermittently connected networks, a new kind of packet delivery latency called *buffer delay* is introduced. Buffer delay is the duration from the moment when a packet is available for sending to the time when the packet is successfully sent out. Since buffer delay (in the order of seconds) is much longer than other kinds of delays (e.g., processing delay, transmission delay and propagation delay, which are in the order of milliseconds), we only consider buffer delay in this paper.

In our model, the wireless communication link between any pair of nodes is symmetric, that is, two neighboring nodes can talk to each other if their Euclidean distance is within the communication range. We assume all the sensor nodes are stationary, and each sensor node does not need to know its own location in this paper.

B. Problem Formulation

First, we introduce some notations:

- S_i denotes the start time of node v_i's working period.
- E_i denotes the end time of node v_i's working period, and $E_i \equiv S_i + T_w \pmod{T_r}$.
- $d^{i,j}(t)$: the one hop buffer delay between node v_i and v_j when v_i has a packet to send at time *t*.
- $\mathcal{R}_i^{j,k}$: the i^{th} route between the source node v_j and the destination node v_k. We can also use this notation to depict a subsegment of a route. For instance, $\mathcal{R}_i^{0,n}$ is a n-hop route $v_0, v_1, ..., v_n$, and $\mathcal{R}_i^{j,k}, 0 < j < k < n$ denotes one of its subroutes. For simplicity, we use $\mathcal{R}_i$ to represent $\mathcal{R}_i^{j,k}$ if there is no confusion.
- $\mathcal{L}_i^{j,k}(t)$: the *packet delivery latency* of a given route $\mathcal{R}_i^{j,k}$, in which *t* is the time when the packet is available. For instance, the packet delivery latency of a route $\mathcal{R}_i^{0,n} = \{v_0, v_1, ..., v_n\}$ can be calculated by the following:

$$\mathcal{L}_i^{0,n}(t) = \sum_{k=1}^{n} d^{k-1,k} \tag{1}$$

  For simplicity, we use $\mathcal{L}_i(t)$ to represent $\mathcal{L}_i^{j,k}(t)$ if there is no confusion.

With these notations, we can formally define *minimum latency route*.

Definition 1: Minimum Latency Route. Suppose there are *n* different routes between the source node v_s and the destination node v_d, denoted as $\mathcal{R}_1^{s,d}, ..., \mathcal{R}_n^{s,d}$. If a route $\mathcal{R}_k^{s,d}$ satisfies $\mathcal{L}_k^{s,d}(t) \leq \mathcal{L}_i^{s,d}(t), i \neq k, 1 \leq i \leq n$, we call this route the *minimum latency route* from v_s to v_d when the packet is available at *t*, and use $\mathcal{ML}\mathcal{R}^{s,d}(t)$ and $\mathcal{MDL}^{s,d}(t)$ to denote the minimum latency route and its packet delivery latency respectively.

Similar to traditional routing schemes in mobile ad hoc networks, the minimum latency routing algorithm can be on-demand or proactive. The on-demand approach is used when

the source and destination nodes are not frequently communicate with each other. Formally, the *On Demand Minimum Latency Routing problem* is defined as follows.

Given the source node v_s and the destination node v_d, and suppose the data is available at time t in the source node, our goal is to find the minimum latency route from v_s to v_d on-demand.

In this paper, we present an **O**n **D**emand **M**inimum **L**atency routing algorithm (ODML), which can find a minimum latency route reactively given the time when the source node has a packet to send. However, on-demand routing does not work well when the source and the destination frequently communicate with each other. In intermittently connected network, the network topology varies with time. As a result, the minimum latency route may also change accordingly. Since the route provided by ODML is only optimal at the time when the packet is ready for sending, the source node has to invoke ODML again if it wants to send another packet at different time. Besides the delay of finding the route, it will incur enormous unnecessary communication overhead. For instance, consider the example shown in Fig. 1, if there is a packet available at each time point within the working period of node A, A has to run ODML 10 times to find minimum latency route for each packet. Apparently, this is not necessary because the minimum latency route does not change within some consecutive time slots, e.g., from time 5 to 10. In this scenario, proactive routing should be used. Instead of finding a minimum latency route at each time slot, we aim to find the time points where the minimum latency route changes. We define these time points as below.

Definition 2: Route Transition Point (RTP) denotes the time point when the minimum latency route is about to change. Formally, suppose at time *t*, $\mathcal{R}_1^{s,d}$ is the minimum latency route from v_s to v_d. If at time *t+1*, the minimum latency route changes to a different route, say $\mathcal{R}_2^{s,d}$, t is referred to as the *route transition point*.

As shown in Fig. 1, within the working period of A, there is only one route transition point located at time 10. Thus, ODML only needs to be called twice (at time 5 and 11 respectively), which implies a significant decrease in message overhead. We formulate the *Proactive Minimum Latency Routing problem* as follows.

Given the source node v_s, and the destination node v_d, our goal is to find all the route transition points within the working period of the source node, and find the route during each time period defined by the route transition points.

In this paper, we propose two proactive schemes: the **Optimal P**roactive **M**inimum **L**atency Routing algorithm (optimal-PML), and the **Quick P**roactive **M**inimum **L**atency Routing algorithm (quick-PML). Among them, optimal-PML can minimize the communication overhead, whereas quick-PML can reduce the route acquisition delay, which is the delay to find the minimum latency route.

III. On-Demand Minimum Latency Routing

In this section, we present our on-demand minimum latency routing algorithm (ODML). In ODML, to find a route, the source sends a route request (RREQ) to its neighbors. When an intermediate node receives a RREQ, it records the latency of RREQ, and updates the latency field of RREQ by adding the buffer delay of the next link to the original value. Then, it unicasts the request to other neighbors except the one from which it receives RREQ. Clearly, the first RREQ arriving at the destination went through the minimum latency route. The destination node then unicasts a route reply (RREP) back to the source along the minimum latency route. As the RREP travels back, each node along the path sets up a forward pointer to the node from which the RREP came. In addition, the arrival time of RREQ and the corresponding latency to the destination are recorded in the routing table. As can be seen, ODML is based on AODV but with some fundamental difference.

First, instead of broadcasting the RREQ as in AODV, each node in ODML unicasts RREQ to its neighbor when this neighbor's working period starts because the working periods of neighbors are usually different. In the header of RREQ, a latency field (instead of hops) is used to represent the latency that the packet has accumulated so far.

Second, two new fields are added in the routing table. The first field is used to record the packet arrival time, and the second field is used to record the packet delivery latency to the destination. After the route is constructed, upon receiving a packet, the node forwards the packet to the next hop only when it can find an entry matching both the destination and the arrival time of that packet.

Third, during the route discovery process of AODV, the *expanding ring search* technique is used to reduce the message overhead. In this scheme, a TTL field is placed in the RREQ header, denoting the maximum number of hops the RREQ can travel. TTL is initially set to 1 and increased by 1 after each search. This continues until a route to the destination is found. In this way, the first route found is the shortest route in terms of hop number, but may not be the minimum latency route.

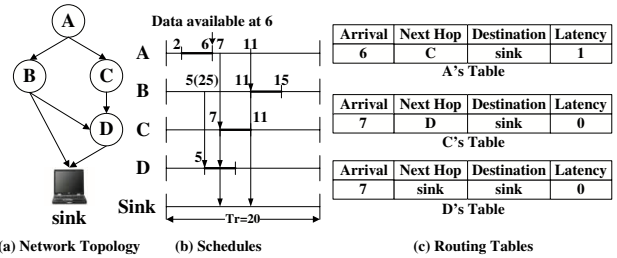
To address this problem, in ODML, TTL is set as *the product of the maximum hop number and the expected per-hop buffer delay (E(d))*. *E(d)* can be estimated using the following formula:

$$E(d) = 0 \times \frac{T_w}{T_r} + \frac{T_r - T_w}{2} \times \frac{T_r - T_w}{T_r} = \frac{(T_r - T_w)^2}{2T_r} \qquad (2)$$

The maximum hop number is initiated to be 1. During the route discovery process, before an intermediate node forwards the RREQ to its subsequent node, it subtracts the buffer delay of this link from the current TTL value. When TTL becomes negative, the packet is dropped. When time out, the source node re-sends the RREQ after increasing the maximum hop number by 1.

Example 3.1: We use an example to illustrate how ODML works. As shown in Fig. 2, A wants to find a minimum latency route to the sink after the data is available at time 6. Knowing

its neighbors' schedule, A sends a RREQ to C at time 7 (the start time of C's working period) and to B at time 11, with the latency field (of RREQ) set to be 1 (7-6) and 5 (11-6), respectively.



Arrival	Next Hop	Destination	Latency
6	C	sink	1

A's Table

Arrival	Next Hop	Destination	Latency
7	D	sink	0

C's Table

Arrival	Next Hop	Destination	Latency
7	sink	sink	0

D's Table

(c) Routing Tables

Fig. 2. Illustration of ODML

After B receives the RREQ from A, it records the field of packet arrival time as 11, updates the latency field of RREQ to be 19 (5+14) before forwarding it to D at time 25, as it has to wait until the start of D's next working period. Similarly, C records the field of packet arrival time, which is 7, and sends it to D immediately. When D receives the route requests from B and C, it only forwards the one from C (the one first arrives) to the destination. After the sink receives the request from D, which has the minimum latency, it sends a RREP back to the source following the same route that the request has traveled along, with the routing table of each intermediate node updated (shown in Fig. 2(c)). Later, the RREQ sent by B will be dropped by the sink.

IV. Proactive Minimum Latency Routing

If the source and the destination node frequently communicate with each other, the on-demand approach may not work well, and the proactive approach should be used. The objective of proactive minimum latency routing is to provide a minimum latency route whenever a packet is ready to be sent. To achieve this goal, ODML should be proactively invoked to locate all the route transition points (RTP) and find the corresponding minimum latency routes.

In this section, we introduce two efficient proactive schemes: the optimal proactive minimum latency routing algorithm (optimal-PML) which can minimize the *call number of ODML* (the number of times ODML is called) and the quick proactive minimum latency routing algorithm (quick-PML) which can significantly reduce the route acquisition delay.

A. Optimal-PML

The goal of optimal-PML is to find the route transition points. If we can identify which points are likely to become the route transition points and which points are not, the search space can be significantly reduced. To achieve this, let's observe the latencies of routes in Fig. 1. Fig. 3 compares the latencies of route $A \rightarrow B \rightarrow sink$ (B for short) and $A \rightarrow D \rightarrow E \rightarrow sink$ (DE for short), and displays the minimum between them at each point within A's working period. As can be seen, route B's optimality ends at time 10, where its latency increases abruptly at the next point. By observation, we find that time point 10 has following characteristics: (1) if a packet is available at this point, it will arrive at B exactly at the end point of B's working period. (2) if a packet is available to be sent at the next point, i.e., time 11 (suppose this packet still follows route B), it will arrive at B at the start point (time 30) of B's working period in the next round (since B has entered sleep period of current round). Motivated by this observation, we make a hypothesis that the minimum latency route only changes at the points with the above characteristics. Now we formally define this kind of points and prove our hypothesis.

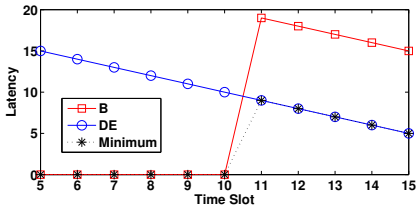


Fig. 3. Latencies of routes in Fig. 1

Definition 3: Sleep Transition Point (STP). As illustrated in Fig. 4, suppose $\mathcal{R}_k$ is a route from v_0 to v_n, and at time *t* and *t+1*, two packets (P_t and P_{t+1}) are available to be sent at v_0. If the time when P_t arrives at an intermediate node v_i is exactly the end time of v_i's working period, i.e., $t + \mathcal{L}_k^{0,i}(t) = E_i$, and the time when P_{t+1} arrives at v_i is the start time of v_i's working period in the next round, i.e., $t + 1 + \mathcal{L}_k^{0,i}(t+1) = S_i + T_r$, time t is referred to as a *sleep transition point* of $\mathcal{R}_k$, denoted as $STP_k^{0,n}$. v_i is referred to as the *sleep transition node (STN)* of $STP_k^{0,n}$.

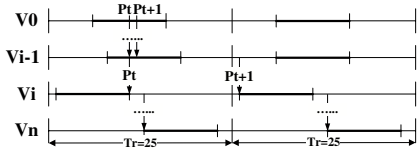


Fig. 4. Sleep transition point

The sleep transition point denotes the time when a node in a route is about to enter sleep, which usually incurs an abrupt increase in the latency of this route. As shown in Fig. 4, the delivery latency of packet P_{t+1} is much larger than that of P_t. Actually, *the sleep transition point is exactly the end time of the corresponding sleep transition node*. Below is the formal proof.

Lemma 1: Suppose $\mathcal{R}_k$ is a route from v_0 to v_n, t_1 and t_2 are two time points within the working period of v_0, and furthermore, there is no STP_k from t_1 to t_2-1. If $\mathcal{L}_k(t_1) = 0$, then $\mathcal{L}_k(t_2) = 0$.

Proof: As $\mathcal{L}_k(t_1) = 0$, $\mathcal{R}_k$ is connected at time t_1. Since there is no STP_k from t_1 to $t_2 - 1$, $\mathcal{R}_k$ is always connected from t_1 to t_2. Therefore, the latency of $\mathcal{R}_k$ is always zero from t_1 to t_2. ■

Lemma 2: Suppose $\mathcal{R}_k$ is a route from v_0 to v_n, t_1 and t_2 are two time points within the working period of v_0, and furthermore, there is no STP_k from t_1 to $t_2 - 1$. If $\mathcal{L}_k(t_1) \geq t_2 - t_1$, then $\mathcal{L}_k(t_2) = \mathcal{L}_k(t_1) - (t_2 - t_1)$.

Proof: Suppose $t_1 + j, 0 \leq j < t_2 - t_1$ is any time point between t_1 and t_2. Since $\mathcal{L}_k(t_1) \geq t_2 - t_1$, it can be inferred

that $\mathcal{L}_k(t_1+j) \geq \mathcal{L}_k(t_1)-j > 0$ because the packet available at t_1 cannot arrive at the destination earlier than the packet available at t_1+j. Therefore, the one-hop buffer delays of some links along $\mathcal{R}_k$ are nonzero at time t_1+j. Suppose link v_iv_{i+1} is the first link with nonzero buffer delay, in other words, $\mathcal{L}_k^{0,i}(t_1+j)=0$ and $d^{i,i+1}(t_1+j)>0$. Since t_1+j is not a STP_k, by *lemma 1*, $\mathcal{L}_k^{0,i}(t_1+j+1)=0$. Therefore, the packets available at t_1+j and t_1+j+1 will reach node v_i at t_1+j and t_1+j+1 respectively. Since $d^{i,i+1}(t_1+j)>0$, it can be inferred that v_i has to buffer the data available at t_1+j for at least one slot until v_{i+1} wakes up, i.e., $t_1+j \leq S_{i+1}-1$, from which we can derive $d^{i,i+1}(t_1+j)=S_{i+1}-(t_1+j)>0$ (i). On the other hand, since $t_1+j \leq S_{i+1}-1$, so $t_1+j+1 \leq S_{i+1}$, then we can get $d^{i,i+1}(t_1+j+1)=S_{i+1}-(t_1+j+1)\geq 0$ (ii). Therefore, by Eqn.(i) and Eqn.(ii), $d^{i,i+1}(t_1+j+1)=d^{i,i+1}(t_1+j)-1$. Consequently, these two packets arrive at node v_{i+1} at the same time, i.e., S_{i+1}. As a result, from node v_{i+1} to the destination node, the packet delivery latency of these two packets will be the same. In summary, the delivery latency of the packet available at time t_1+j+1 is equal to $\mathcal{L}_k^{0,n}(t_1+j)-1$. Therefore, the difference between the delivery latencies of $\mathcal{R}_k$ at t_1 and t_2 is: $\mathcal{L}_k^{0,n}(t_1)-\mathcal{L}_k^{0,n}(t_2)=\sum_{i=0}^{t_2-t_1-1}(\mathcal{L}_k^{0,n}(t_1+i)-\mathcal{L}_k^{0,n}(t_1+i+1))=t_2-t_1$ ■

Theorem 1: Suppose $\mathcal{R}_k$ is a route from v_0 to v_n. Furthermore, t is a sleep transition point of $\mathcal{R}_k$, and v_i is the sleep transition node of t, then $t=E_i$, i.e., $\mathcal{L}_k^{0,i}(t)=0$.

Proof: By contradiction, suppose $E_i-t=d>0$. First of all, we suppose there is only one sleep transition node along $\mathcal{R}_k$, therefore, t is not a sleep transition point of the subroute $\mathcal{R}_k^{0,i}$. Since $\mathcal{L}_k^{0,i}(t)=d>0$, by Lemma 2, $\mathcal{L}_k^{0,i}(t+1)=\mathcal{L}_k^{0,i}(t)-1$. Therefore, the time when the packet available at *t* arrives at v_i is the same as the time when the packet available at *t+1* arrives at v_i; however, this conflicts with the definition of sleep transition point. Similarly, it can be proved that the conclusion still holds if there are multiple sleep transition nodes of t with the same end times of working periods. ■

Now, we prove that *only the sleep transition point has the potential to become a route transition point*.

Theorem 2: Suppose there are *m* routes from v_s to v_d, denoted as $\mathcal{R}_1$, $\mathcal{R}_2$,..., $\mathcal{R}_m$, t_1 and t_2 are two time points within the working period of v_s. In addition, at time t_1, the minimum latency route is $\mathcal{R}_1$. If there is no STP_1 from t_1 to t_2, then the minimum latency route does not change, i.e., at any time point t_1+j, $0 \leq j \leq t_2-t_1$, $\mathcal{MLR}^{s,d}(t_1+j)=\mathcal{R}_1$.

Proof: By assumption, $\mathcal{R}_1$ is the route with minimum delivery latency at time t_1. There are three possible cases regarding $\mathcal{L}_1(t_1)$.

Case 1: $\mathcal{L}_1(t_1)=0$.
Since there is no STP_1 from t_1 to t_2, by *lemma 1*, $\mathcal{L}_1(t_1+j)=0, 0\leq j\leq t_2-t_1$. Obviously, $\mathcal{L}_1(t_1+j)\leq \mathcal{L}_i(t_1+j), 2\leq i\leq m$, thereby $\mathcal{MLR}^{s,d}(t_1+j)=\mathcal{R}_1$.

Case 2: $\mathcal{L}_1(t_1)\geq t_2-t_1$.
By *lemma 2*, $\mathcal{L}_1(t_1+j)=\mathcal{L}_1(t_1)-j, 1\leq j\leq t_2-t_1$. On the other hand, $\mathcal{L}_i(t_1+j)\geq \mathcal{L}_i(t_1)-j$ as proved in lemma 2. Therefore, given $\mathcal{L}_1(t_1)\leq \mathcal{L}_i(t_1)$, $\mathcal{L}_1(t_1+j)\leq \mathcal{L}_i(t_1+j)$ can be derived.

Case 3: $\mathcal{L}_1(t_1)<t_2-t_1$.
Let $t_3=t_1+\mathcal{L}_1(t_1)$, by *lemma 2*, $\mathcal{L}_1(t_1)-\mathcal{L}_1(t_3)=t_3-t_1=\mathcal{L}_1(t_1)$, thus $\mathcal{L}_1(t_3)=0$. Similarly, it can also be proved that $\mathcal{L}_1(t_3-1)=1$. Therefore, during the period from t_1 to t_3-1, we can apply the proof of *Case 2*, and during the period from t_3 to t_2, the proof of *Case 1* is applicable. In summary, $\mathcal{R}_1$ is always the minimum latency route from t_1 to t_2. ■

Based on *Theorem 2*, if *t* is not a sleep transition point, it can not be a route transition point. In other words, a route transition must happen at a sleep transition point. Therefore, to find the time point where the current minimum latency route loses its optimality, only its sleep transition points need to be checked.

optimal-PML: *Theorem 1* tells us that only the end time of the working period of a node could be a sleep transition point. Therefore, the basic idea of optimal-PML is to sequentially check all the end times of the nodes in the current minimum latency route, until a new minimum latency route is found. This process repeats and eventually all the minimum latency routes will be identified. Note that, if the current minimum latency route is found at time t, only those end times that are larger than t need to be checked.

The pseudo code of the algorithm is shown in Algorithm 1. Variables stp and rtp are used to store the latest sleep transition point and route transition point. $time\_list$ records the start point (1 slot after the route transition point) of each minimum latency route, which is stored in $route\_list$.

Algorithm 1 Optimal Proactive Minimum Latency Routing

Procedure: Optimal_Proactive_Routing;
Input: $(G, s, start, end)$;
Output: $(time\_list, route\_list)$;
1: $stp \leftarrow start-1$
2: $rtp \leftarrow start-1$
3: $start\_route \leftarrow$**ODML**$(G, s, start)$
4: $time\_list \leftarrow start$
5: $route\_list \leftarrow start\_route$
6: $current\_route \leftarrow start\_route$
7: **repeat**
8: *Find node v_i with earliest end time larger than stp in $current\_route$*
9: $stp \leftarrow E_i$
10: $stp\_route \leftarrow$**ODML**$(G, s, stp+1)$
11: **if** $stp\_route \neq current\_route$ **then**
12: $rtp \leftarrow stp$
13: $current\_route \leftarrow stp\_route$
14: $time\_list \leftarrow rtp+1$
15: $route\_list \leftarrow current\_route$
16: **until** $stp \geq end$

Initially, stp and rtp are set as 1 slot ahead of the start time of the source's working period (line 1-2), and the minimum latency route at the start time is used as the current minimum latency route (line 3-6). In each iteration (line 7-16), the node with the earliest end time larger than stp (last sleep transition point checked by optimal-PML) along the current minimum latency route is found (line 8) and its end time is used to update stp (line 9). Then, the algorithm checks whether the minimum latency route at $stp+1$ is the same as the current route (line 11). If the result is true, a new minimum latency

route is found, and the corresponding variants are updated (line 12-15). Otherwise, a new iteration is started.

To implement optimal-PML, we modify ODML as follows.

- To get the next STP (end time) to be checked, we add two new fields to the header of RREP. *last_stp* is used to record the last sleep transition point checked by optimal-PML, and *earliest_stp* is used to record the earliest STP larger than last STP. When RREP travels back to the source, each intermediate node compares its own end time with the values in these two fields. If its end time is earlier than the value in *earliest_stp*, and larger than the value in *last_stp*, it updates the *earliest_stp* field with its end time.
- In order to compare the routes at different time points, whenever ODML is called, the information of the whole route must be returned to the source node. However, it is difficult to keep the IDs of all nodes along the route in RREP since the packet size is very small (e.g., in TinyOS, the default packet size is 36 bytes, out of which 29 bytes are used for the actual payload).

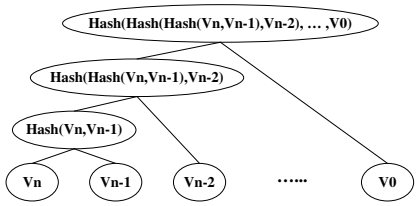
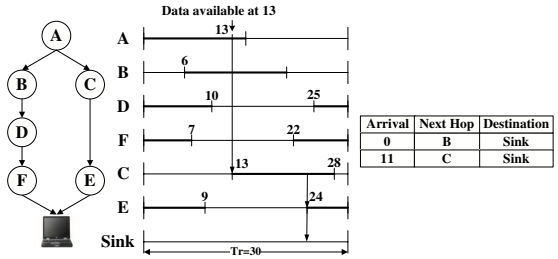


Fig. 5. A Merkle-hash tree constructed from the ids of the nodes traveled by a RREP

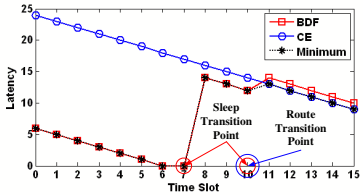
To address this problem, we apply Merkle hash tree [9], and add a new field (*route_hash*) to the header of RREP. As illustrated in Fig. 5, when RREP travels back to the source, in each intermediate node, *route_hash* is updated by hashing the concatenation of the old value and the ID of this node. Therefore, the length of *route_hash* is independent of the length of the route. To compare the routes, *route_hash* stored in RREP can be used instead of the ID sequence along the route.

Example 4.1: We use a simple example to illustrate how optimal-PML works. As shown in Fig. 6[1], node A has two possible routes to the sink: $A \rightarrow B \rightarrow D \rightarrow F \rightarrow sink$ (BDF for short) and $A \rightarrow C \rightarrow E \rightarrow sink$ (CE).

Initially, through ODML, at the start time (time 0) of A's working period, route BDF is the minimum latency route. In the first iteration of optimal-PML, since time 7 (the end time of F's working period) is route BDF's earliest sleep transition point larger than 0, ODML is invoked for a second time at time 8(7+1). However, as shown in Fig. 6(d), the latency of route BDF is still the minimum at this point. This shows that *a sleep transition point is not necessarily a route transition point*. In the second iteration, ODML is invoked at the point 1 slot after the earliest transition point larger than 8, which is point 11 (1 slot after the end time of D's working



(a) Network Topology (b) Schedules (c) A's Routing Table



(d) Packet Delivery Latency

Fig. 6. Illustration of optimal-PML

period), resulting in a new minimum latency route, namely CE. Optimal-PML terminates at this time because route CE's earliest sleep transition point larger than 11 (the end time of node C, which is 28) is located outside the working period of A. Thus, A has two minimum latency routes: BDF from time 0 to 10, and CE from time 11 to 15.

In optimal-PML, when a node has a packet to send, it may find multiple entries having the same destination in its routing table. These entries have different packet arrival times, with corresponding next hops. At this time, the node forwards the packet according to the entry which has the largest packet arrival time earlier than the arrival time of the packet to be sent. For example, as shown in Fig. 6(c), node A has two entries destined to the sink. Suppose a packet is available at A at time 13. The second entry has the largest packet arrival time earlier than 13, and thus the packet should be sent to C.

B. quick-PML

Although optimal-PML can minimize the call number of ODML, it has long route acquisition delay because it can find only one minimum latency route in each *route discovery round*, which is defined as the period from the moment when a RREQ is ready to be sent to the time when the corresponding RREP returns[2]. To address this problem, we propose a quick proactive minimum latency routing algorithm (quick-PML). Different from optimal-PML, which sends RREQs serially, quick-PML applies multiple route discoveries simultaneously.

quick-PML is inspired by *binary search algorithm*, which borrows the idea of divide and conquer. In each route discovery round, there are two time points: the start point and the end point, whose corresponding minimum latency routes (referred to as the start route and end route) are different. quick-PML finds the middle point between the start and the end points, and compares the middle route with the start and end routes. If the middle route is different from the start (end) route, the middle

[1]Note that bold lines denote the working periods, and the end time of a node may be earlier than its start time, such as node D.

[2]Note that route discovery round is conceptually different from the round defined in Section II, which is the unit of sensor's working cycle.

point becomes a new end (start) point. Recursively, quick-PML narrows the search space and approaches the route transition points. In the end, all the route transition points within the working period of the source node will be identified.

quick-PML is based on the following assumption: If the minimum latency routes at two time points are identical, there is no route transition point between these two points. Actually, this is not always true. However, if one of the following two conditions is satisfied, the minimum latency route between two points, t_1 and t_2, will not change.

1) $\mathcal{MDL}(t_2) > 0$ and $\mathcal{MDL}(t_1) - \mathcal{MDL}(t_2) = (t_2 - t_1)$.
2) $\mathcal{MDL}(t_2) = 0$ and $t_2 - t_1 < T_r - T_w$

Condition 1 can be guaranteed by *theorem 3*.

Theorem 3: Suppose t_1 and t_2 are two time points within the working period of v_s. In addition, at time t_1 and t_2, the minimum latency routes are both $\mathcal{R}_1$. If $\mathcal{L}_1(t_1) - \mathcal{L}_1(t_2) = (t_2 - t_1)$, then the minimum latency route does not change from t_1 to t_2.

Proof: Since $\mathcal{L}_1(t_1) - \mathcal{L}_1(t_2) = (t_2 - t_1)$, the packet available at t_1 reaches the destination at the same time as the packet available at t_2. Thus, it can also be inferred that the packets available between t_1 and t_2 reach v_d simultaneously, i.e., $\mathcal{L}_1(t_1 + j) = \mathcal{L}_1(t_1) - j, 0 \leq j < (t_2 - t_1)$. On the other hand, for any other route $\mathcal{R}_i$, $\mathcal{L}_i(t_1 + j) \geq \mathcal{L}_i(t_1) - j$ as proved in lemma 2. Therefore, given $\mathcal{L}_1(t_1) \leq \mathcal{L}_i(t_1)$, $\mathcal{L}_1(t_1 + j) \leq \mathcal{L}_i(t_1 + j)$ can be derived, the minimum latency route keeps fixed. ■

Condition 2 can be guaranteed by *theorem 4*. Below is the formal description of *theorem 4*, preceded by *lemma 3* which is used to prove *theorem 4*.

Lemma 3: Suppose $\mathcal{R}_k$ is a route from v_0 to v_n, t is a time point within the *working period* of v_0. If t is a STP_k, then $\mathcal{L}_k(t + 1) \geq T_r - T_w - 1$.

Proof: Suppose v_i is t's first sleep transition node along $\mathcal{R}_k$. Since v_{i-1} has to buffer the packet available at $t + 1$ until the start time of v_i's working period in the next round, $d^{i-1,i}(t+1) = T_r - T_w - 1$ (the packet available at $t+1$ gets to v_{i-1} at $E_i + 1$, gets to v_i at $S_i + T_r$). Furthermore, since $\mathcal{L}_k^{0,i-1}(t+1) = 0$ and $\mathcal{L}_k^{i,n}(S_i) \geq 0$, therefore, $\mathcal{L}_k^{0,n}(t+1) = \mathcal{L}_k^{0,i-1}(t+1) + d^{i-1,i}(t+1) + \mathcal{L}_k^{i,n}(S_i) \geq T_r - T_w - 1$. ■

Theorem 4: Suppose $\mathcal{R}_k$ is a route from v_0 to v_n, t_1 and t_2 are two time points within the working period of v_0, and there is at least one sleep transition point from t_1 to $t_2 - 1$. It's always true that $t_2 - t_1 \geq T_r - T_w - \mathcal{L}_k(t_2)$.

Proof: Suppose there is only one sleep transition point from t_1 to $t_2 - 1$, denoted as t. According to *lemma 3*, $\mathcal{L}_k(t + 1) \geq T_r - T_w - 1$, further by *lemma 2*, $t_2 - (t + 1) = \mathcal{L}_k(t + 1) - \mathcal{L}_k(t_2) \geq T_r - T_w - 1 - \mathcal{L}_k(t_2)$, so we get $t_2 - t \geq T_r - T_w - \mathcal{L}_k(t_2)$ (i). Furthermore, $t - t_1 = \mathcal{L}_k(t_1) - \mathcal{L}_k(t) \geq 0$ (ii) by *lemma 2*. Therefore, by Eqn.(i) and Eqn.(ii), it can be deduced that $t_2 - t_1 \geq T_r - T_w + (\mathcal{L}_k(t_1) - \mathcal{L}_k(t)) - \mathcal{L}_k(t_2) \geq T_r - T_w - \mathcal{L}_k(t_2)$.

Obviously, the difference between t_1 and t_2 will be even larger if there are multiple sleep transition points from t_1 to $t_2 - 1$. Therefore, $T_r - T_w - \mathcal{L}_k(t_2)$ is the lower bound of $t_2 - t_1$. ■

Based on *theorem 4*, when $\mathcal{MDL}(t_2) = 0$, if $t_2 - t_1 < T_r - T_w$, there is no sleep transition point along the minimum latency route between t_1 and t_2. Further by *theorem 2*, the minimum latency route does not change.

Algorithm 2 Quick Proactive Minimum Latency Routing

Procedure: Quick_Proactive_Routing;
Input: $(G, s, start, end)$;
Output: $(time\_list, route\_list)$;
1: $start\_route \leftarrow$ **ODML**$(G, s, start)$
2: $end\_route \leftarrow$ **ODML**(G, s, end)
3: $time\_list \leftarrow start$
4: $route\_list \leftarrow start\_route$
5: **Binary_Routing**(*G,s,start,end,start_route,end_route*)

Procedure: Binary_Routing;
1: **if** $start = end - 1$ **then**
2: $time\_list \leftarrow end$
3: $route\_list \leftarrow end\_route$
4: **else**
5: $mid \leftarrow \lfloor (low + up)/2 \rfloor$
6: $mid\_route \leftarrow$**ODML**(*G,s,mid*)
7: **if** $start\_route \neq mid\_route$ **then**
8: **Binary_Routing**(*G,s,start,mid,start_route,mid_route*)
9: **else if** $mid\_route.latency > 0$ **and** $start\_route.latency - mid\_route.latency < mid - start$ **then**
10: **Binary_Routing**(*G,s,start,mid,start_route,mid_route*)
11: **else if** $mid\_route.latency = 0$ **and** $mid - start \geq T_r - T_w$ **then**
12: **Binary_Routing**(*G,s,start,mid,start_route,mid_route*)
13: **if** $mid\_route \neq end\_route$ **then**
14: **Binary_Routing**(*G,s,mid,end,mid_route,end_route*)
15: **else if** $end\_route.latency > 0$ **and** $mid\_route.latency - end\_route.latency < end - mid$ **then**
16: **Binary_Routing**(*G,s,mid,end,mid_route,end_route*)
17: **else if** $end\_route.latency = 0$ **and** $end - mid \geq T_r - T_w$ **then**
18: **Binary_Routing**(*G,s,mid,end,mid_route,end_route*)

The pseudo code of quick-PML is depicted in Algorithm 2. Initially, the minimum latency routes at start and end points are obtained by calling ODML (line 1-2), with the start time and the corresponding route recorded in $time\_list$ and $route\_list$ respectively (line 3-4). Then, the *binary_routing* (BR for short) procedure is called as a subroutine (line 5). The *binary_routing* procedure is self-recursive. It finds the minimum latency route at the middle point (BR:line 6) and compares it with the routes at start and end points (BR:line 7-18). If the middle route (route at the middle point) is different from the start route or end route, the *binary_routing* procedure will be recursively called to approach the route transition points (BR:line 8 and 14). Otherwise, quick-PML will check the latencies of the middle route and the start (end) route to determine whether the conditions discussed earlier are satisfied. If none of the conditions are satisfied, quick-PML will also invoke the *binary_routing* procedure (BR:line 10,12,16,18). The recursion ends when the start point is exactly one slot before the end point (BR:line 1-3).

Example 4.2: Let's still use the example shown in Fig. 6. Figure 7 shows the search tree of quick-PML. In the search tree, each node denotes a time point within the working period of A, and the nodes on the same level of the tree are checked in parallel in the same route discovery round. The black node (time 10) denotes the route transition point, and the shaded nodes denote the time points checked by quick-PML. As can

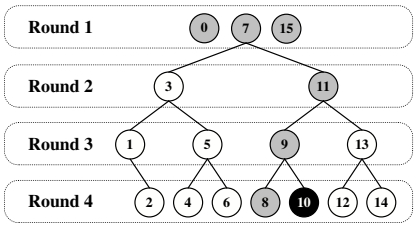


Fig. 7. Search Tree of quick-PML

be seen, in the the first route discovery round, the minimum latency routes at the start point (0), end point (15), and the middle point (7), are found. Since the route at time 0 is the same as that at time 7, and condition 2 is satisfied, the time points between 0 and 7 will not be checked. On the other hand, since the route at 15 is different from that at time 7, a new middle route (at point 11) is checked in the second round. This route is equal to the route at 15, so there's no need to check the routes from time 11 to 15 (condition 1 is satisfied). Since the route at 11 is different from that at time 7, the route at time 9 is checked in the third round. Although the route at time 9 is the same as that at time 7, as neither of the conditions is satisfied, ODML has to be called at time 8. Eventually, the route transition point is found in the forth round.

Suppose there are n minimum latency routes within the working period of the source node. Since quick-PML applies multiple route discoveries simultaneously, it's clear that when $n > \log_2(T_w - 2)$, quick-PML takes smaller number of route discovery rounds than optimal-PML. As a result, quick-PML tends to identify all the route transition points in a shorter time. This claim is justified by the simulation results in Section VI.

V. Discussions

A. Call Number of ODML

For optimal-PML, its call number of ODML (number of times ODML is called) is equal to the number of sleep transition points checked by it. Simulation result shows that this number is slightly larger than the number (denoted as n) of route transition points. On the other hand, for quick-PML, it's evident that its call number of ODML can be bounded by n (lower bound) and the product of n and the height of the search tree (upper bound). Furthermore, the expected call number of ODML of quick-PML, denoted as $E(N_c)$, can be estimated by the following formula:

$$E(N_c) = \sum_{i=2}^{\log_2(T_w-2)} 2^{i-1}(1-(1-\frac{n}{T_w})^{(2^{\log_2(T_w-2)-i+1}-1)}) + 3 \quad (3)$$

B. Energy Consumption and Balancing

Energy Consumption: Since the schemes of this paper are dedicated to achieving minimum packet delivery latency between communicating nodes, the minimum latency route found by our schemes may not have as small number of hops as the shortest path. Therefore, if all the nodes in the network transmit packets using the same power, the energy consumption of the minimum latency route may be larger than that of the shortest path. However, if each sensor node can adjust its transmitting power based on the distance to the receiving node and the background noise, the energy consumption gap between the minimum latency route and the shortest path can be significantly reduced.

Nowadays, the capability of power control is already available for the state-of-the-art sensor nodes. For example, the RF power of a MICA2 node can be adjusted ranging from -20 dBm to 5 dBm [10]. In the most common power-attenuation model [11], the signal power falls as $1/d^{\alpha}$ where d is the Euclidean distance between two transmitting nodes and α is the pass loss exponent of the wireless environment, a real constant typically between 2 and 5. In this paper, we assume that all sensors have the same power threshold for signal detection, which is typically normalized to 1. Furthermore, we normalize the time to transmit a packet to 1. Therefore, the energy consumption required for a node v_a to transmit a packet to a neighbor node v_b is $d_{a,b}^{\alpha}$. Using this model, the minimum latency route can achieve a smaller per hop energy consumption than the shortest path due to the smaller per hop Euclidean distance.

Energy Balancing: Although the shortest path can achieve the minimum number of transmissions, it may not be optimal from the perspective of network lifetime and long-term connectivity. Since the shortest path keeps unchanged all the time, in each transmission, only the energy of the nodes along the shortest path is consumed, resulting in a wide disparity in energy level, and eventually a disconnected network. In contrast, in our proactive minimum latency routing schemes, the routes traveled by the packets between a source-destination pair vary over time. As a result, nodes burn energy in a more equitable way, and thus the lifetime of the network is extended. This could be regarded as another merit of our schemes.

C. Imperfect Link

In our discussions and examples, we assume perfect link quality. If the link quality is not perfect, retransmissions may occur. A simple solution is to increase the length of time slots to accommodate retransmissions. As aforementioned, we focus on the buffer delay, which can be in the order of seconds, and thus the delay caused by these retransmissions (in the order of milliseconds) can be ignored.

VI. Performance Evaluations

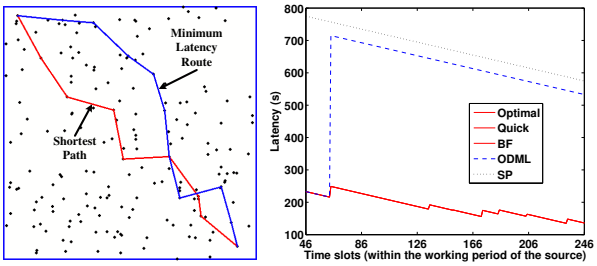
In this section, we evaluate the performance of the schemes proposed in this paper. The evaluation consists of three parts: 1) we use a scenario to illustrate the effectiveness of our schemes; 2) we compare the performance of the minimum latency route and the shortest path, and 3) we compare the performance of the two proactive minimum latency routing algorithms.

A. A Scenario

Consider a scenario where 200 sensor nodes are deployed in a $500 \times 500m^2$ area. The communication range of a sensor node is $100m$, $T_r = 500ms$, and $T_w = 200ms$. Initially each node randomly selects its start time of the working period. We evaluate the routing schemes when the node on the upper left

corner sends packets to the node on the lower right corner. Fig. 8(a) shows the shortest path and the minimum latency route found by ODML at time 46, which is the start time of upper left node's working period.

Figure 8(b) shows the packet delivery latencies of the routes provided by 5 different schemes at each time point within the working period of the node on the upper left corner. Apparently, the shortest path (SP) is the worst one because its latency is the highest all the time. Although the route discovered by ODML has the minimum latency at the start time, it is not optimal after a STP. In contrast, the two proactive minimum latency routing algorithms can guarantee that whenever a packet is available within the working period of the source node, it follows the minimum latency route. For comparison purpose, we also give a brute force scheme (BF), which runs ODML at each time point within the working period of the source node. As shown in the figure, BF, optimal-PML, quick-PML have the same delay.



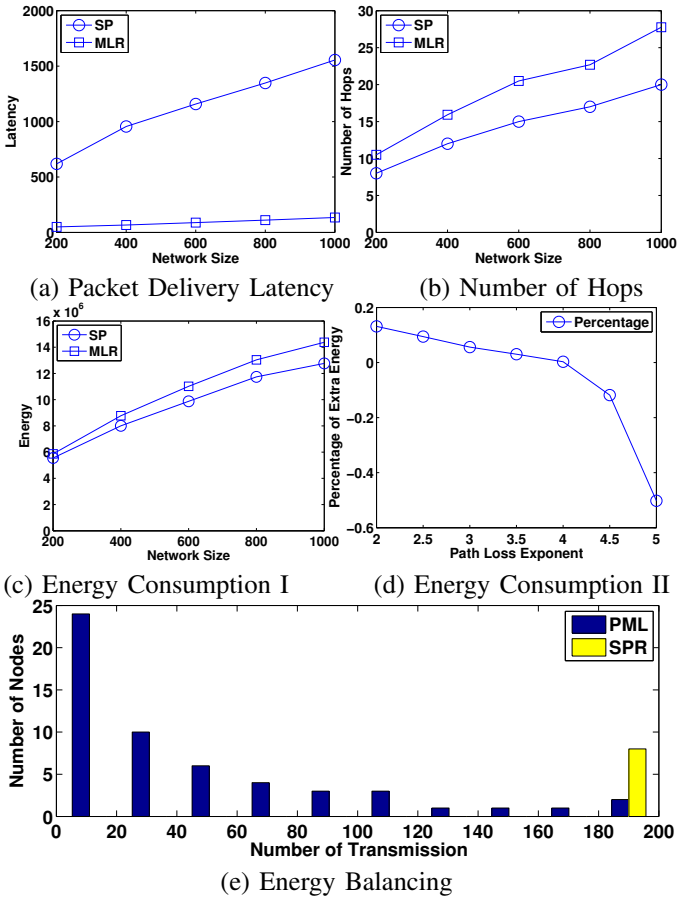
(a) Network Topology (b) Packet Delivery Latency

Fig. 8. A scenario where the node on the upper left corner wants to send packets to the node on the lower right corner

B. Minimum Latency Route VS Shortest Path

In this subsection, we compare the performance of the minimum latency route (MLR) and the shortest path (SP). We still use the aforementioned simulation setup. To test the scalability, the number of sensor nodes increases from 200 to 1000, with fixed node density, i.e., the area of the deployed region increases proportionally to the network size. For each network size, we generate 100 different network topologies, and record the average results shown in Fig. 9.

Figure 9(a) illustrates the comparison of packet delivery latency between the minimum latency route and the shortest path. Obviously, the minimum latency route can achieve much smaller latency than the shortest path. Fig. 9(b) compares the number of hops between two routes. As can be seen, the average length of the minimum latency route is around 30 percent larger than that of the shortest path, which implies 30 percent extra energy consumption (suppose the energy consumed by MLR is denoted as E_{MLR}, and the energy consumed by the SP is denoted as E_{SP},we define *the percentage of extra energy consumption of MLR* as $\frac{E_{MLR}-E_{SP}}{E_{SP}}$) if all the sensor nodes transmit in same power level. However, if power control is adopted, the extra energy consumed by the minimum latency route would be radically reduced. Fig. 9(c) shows the comparison result of the two routes' energy consumption when the path loss exponent α is 3. As we can see, the minimum



(a) Packet Delivery Latency (b) Number of Hops

(c) Energy Consumption I (d) Energy Consumption II

(e) Energy Balancing

Fig. 9. Comparisons between shortest path and minimum latency route

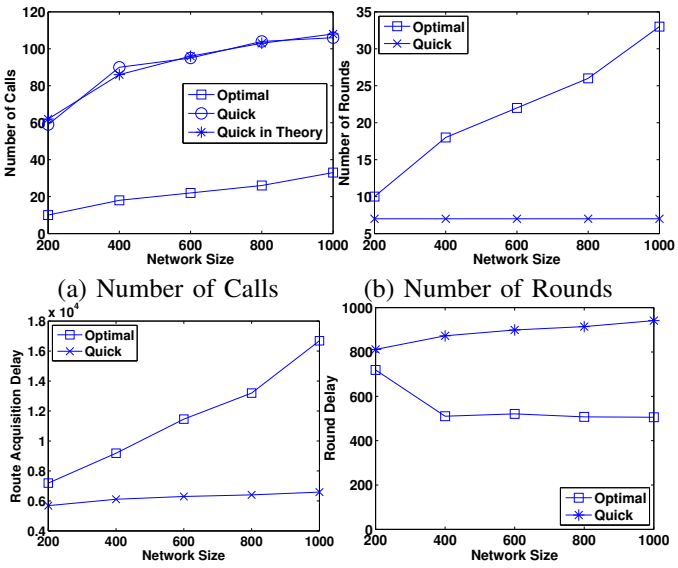
latency route only consumes less than 10 percent extra energy. Moreover, with the increase of the path loss exponent, the extra energy consumption could be further reduced. Fig. 9(d) further illustrates the effect of varying path loss exponent on the percentage of the extra energy consumption of MLR. As can be seen, when the path loss exponent is larger than 4, the percentage becomes negative, which implies that the energy consumption of MLR becomes smaller than that of SP.

To compare the energy balance, we consider a network of 200 nodes. Suppose the working period of each node includes 200 time slots, and the source node sends one packet at each time slot within a working period. Then, we observe how many nodes are involved in the dissemination processes of these 200 packets, and how many transmissions are conducted by each of them. The results of comparison are plotted in Fig. 9(e). In this figure, each bar denotes the number of nodes whose number of transmissions fall into the corresponding region. For example, the leftmost bar implies that there are 24 nodes whose number of transmissions is within the range of [0, 20]. As we can see, when shortest path routing protocols (SPR) are used, since the shortest path does not change throughout the working period of the source, to deliver 200 packets to the destination, each of the nodes along the path needs to transmit 200 times. Therefore, the number of nodes involved in the packet dissemination is fixed to be the length of the shortest path, which is 8. That's the reason why SPR only has one

bar displayed in the graph. On the contrary, when proactive minimum latency routing protocols (PML) are employed, the variation of the minimum latency routes result in more nodes being involved in the packet disseminations. Altogether, there are 54 nodes, each of which works as a relay for at least one time in different minimum latency routes. As observed from the figure, most of the nodes transmit less than 40 times, which implies a more equitable energy consumption from the perspective of the whole network.

C. optimal-PML VS quick-PML

In this subsection, we use the same experimental setting as in the last subsection, and compare the performance of the two proactive routing schemes: optimal-PML and quick-PML.



(a) Number of Calls (b) Number of Rounds

(c) Route Acquisition Delay (d) Round Delay

Fig. 10. Comparisons between optimal-PML and quick-PML

Figure 10(a) compares the message overhead (in terms of call number of ODML) of two schemes. As we can see, optimal-PML has a much better performance than quick-PML in message overhead (its call number of ODML is actually optimal according to our proof in section IV). Figure 10(a) also shows quick-PML's theoretical call numbers of ODML (calculated by Eqn. 3),which are very close to the experimental values. On the contrary, optimal-PML suffers from a large route acquisition delay. As illustrated in Fig. 10(b)(c), both the number of route discovery rounds and the real route acquisition delay of optimal-PML are much larger than those of quick-PML. By contrast, quick-PML can achieve a bounded route acquisition delay, however, at the cost of message overhead. Finally, in Fig. 10(d), optimal-PML is found to have a smaller per round delay (the ratio of the route acquisition delay over the number of route discovery rounds) than quick-PML.

VII. Related Work

The presented work has some similarity to Delay Tolerant Network (DTN), although with much difference. The work in [12] demonstrates that optimal delivery paths in a DTN can be discovered by constructing a directed graph of nodes if the whole network topology and the schedules of all nodes are known. However, this is impossible for a large scale wireless sensor network with limited resources. Furthermore, different from DTN, where the network is partitioned due to the mobility of the mobile nodes, in the sensor networks scheduled by application-driven schemes, although the communication link is intermittently connected, the physical topology of the network is always connected. Therefore, in our scheme, a sensor node can still forward the packet anywhere outside of its working period.

In [13], a dynamic switch-based forwarding (DSF) algorithm is designed to address routing problems in extremely low duty-cycle sensor networks. Here extremely low duty-cycle means that each sensor works for a very short T_w (e.g., a single time slot) within a long T_r (e.g., 1000 slots). Under this assumption, there is probably no route transition. In contrast, the proposed schemes in this paper can be applied for any percentage of the duty-cycle (In this sense, the scenario described in [13] can be considered as a special case of our problem). Furthermore, our schemes can identify route transitions when the working periods of sensor nodes are relatively long, and this is supposed to be the largest contribution of this paper.

VIII. Conclusions

In this paper, we first proposed an on-demand minimum latency (ODML) routing algorithm to find minimum latency routes in intermittently connected sensor networks. Since on-demand routing algorithm does not work well when the source and destination frequently communicate with each other, we then proposed two proactive minimum latency routing algorithms: optimal-PML and quick-PML. In summary, the schemes proposed in this paper can provide generic routing functionalities for most of the existing scheduling schemes.

References

[1] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *SenSys*, 2003.

[2] T. Yan, T. He, and J. A. Stankovic, "Differentiated surveillance for sensor networks," in *SenSys*, 2003.

[3] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *MobiCom*, 2004.

[4] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection," in *IPSN*, 2005.

[5] M. Cardei, M. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *INFOCOM*, 2005.

[6] H. Liu, P. Wan, C.-W. Yi, X. Jia, S. Makki, and P. Niki, "Maximal lifetime scheduling in sensor surveillance networks," in *INFOCOM*, 2005.

[7] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, 1999.

[8] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996, vol. 353.

[9] R. C. Merkle, "A certified digital signature," in *CRYPTO*, 1989.

[10] http://www.xbow.com/.

[11] T.S.Rappaport, *Wireless Communications: Principles and Practices*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1996.

[12] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *SIGCOMM*, 2004.

[13] Y. Gu and T. He, "Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links," in *SenSys*, 2007.