On Energy-Aware Forwarding Schemes in Wireless Sensor Networks

Adrian Fr. Kacsó
Computer Science Department
University of Siegen
57068 Siegen, Germany
Email: adrian.kacso@uni-siegen.de

***Abstract*—The paper discusses two forwarding strategies in wireless sensor networks (WSNs) involving the residual energy of nodes and provides the corresponding distributed algorithms; the routing is achieved without additional routing messages. To reduce excessive broadcasts at the setup phase we consider the impact of the delay time. Simulation results are obtained using our sensor network simulator (SNF).**

***Keywords*-WSN, energy-awareness, routing, simulation.**

I. Introduction

A WSN is a self-organizing communication network consisting of a large number of sensor nodes that are randomly (and densely) deployed in a region, to monitor the environment or some physical phenomenon. Therefore, the distributed nodes individually sense the environment and collaboratively preprocess and communicate their captured information to interested clients (sinks). In request-driven WSNs a sink sends a request (interest) in a data-centric manner, where the destination is specified by tuples of attribute-value pairs of the data carried inside the message. Routing protocols determine routes how messages (interest and data) are forwarded between the sink and sources (nodes able to deliver the requested data) using data-centric approaches. Due to energy constraints, source nodes usually cannot sent the data to sink(s) directly. The data is forwarded by intermediate nodes until it reaches the intended sink(s). The limited energy, the restricted communication and computation abilities (capabilities) of battery powered sensor nodes require energy-efficient routing protocols. The data gathered in a sensor network is highly correlated, due to a spatial and temporal correlation between successive measurements. Exploiting the data-centricity and the spatial-temporal correlation characteristics allows to apply effective in-network techniques, which further improve the energy-efficiency of the communication in WSN. Aggregation can eliminate the inherent redundancy of the raw data collected and, additionally, it reduces the traffic in the network, avoiding in this way congestions and induced collisions.

Routing means to find the right route between the many routes from source(s) to sink(s) by defining a path metrics. The strategy to select the next hop employs various metrics which allows to find different paths, e.g., energy-efficient, shorter, rapid, reliable paths according to the application goals. Some metrics combines the power consumption and latency, whereas others focus on prolonging the lifetime of the network by considering the node's residual energy.

In the present paper, we focus on strategies to design energy-aware routing protocols using metrics that aim to prolong the network lifetime and we illustrate the performance of these distributed algorithms using our SNF simulator. We assume that the network is randomly deployed, each node can be a sink, the sources and traffic are not known apriori and the routing is achieved without additional routing messages. Collisions and retransmissions are also taken into account and the energy model computes automatically the energy consumed according to the state of the radio.

The paper is structured as follows. Section II presents the state-of-art and the motivation behind simulating routing protocols for WSNs; Section III describes two energy-aware routing metrics and the corresponding distributed algorithms. Section IV illustrates some simulation results and unexpected behavior. Section V concludes the paper.

II. Related work and objectives

Many energy-aware routing metrics have been proposed, e.g., [1][2][3][4][5][6][7], to minimize the energy consumption and to prolong the lifetime of the network. Several routing algorithms use distance-based forwarding, where the number of hops serves as a distance metrics. Establishing reverse paths is a very used scheme [2][8][9][10]. Shortest-path routing improves the overall energy consumption since the energy needed to transmit a message from source to intended destination is correlated to the path length.

Unfortunately, shortest path (or minimum energy path) will heavily load nodes on the path and these nodes die sooner, thus creating holes or leading to disconnected networks. Various techniques to balance the load among all forwarding nodes are proposed. Some of them consider the node's residual energy to prevent nodes from choosing the same route often [6], other minimize the variance of the remaining energy between different routes [2][4][5] or use traffic spreading and aggregation as GBR [11]. The robustness to different types of failures (unreliable and asymmetric links, node failures) can be improved by multipath routing [2][9][10], where multiple paths to sink are established. In such cases the routing must incorporate packet delivery rate and link quality metrics.

Energy-efficient routing needs joint optimization with the link layer, since the only way to save energy is to *switch off* the transceiver; thus the MAC must use an active-sleep regime with a low duty cycle [12][13]. In order to study and optimize routing protocols, we use our SNF based on a cross-layer design. The simulator and the modular network architecture of a sensor node were described in [14][15].

III. Energy-aware routing

We discuss two distributed forwarding algorithms for randomly deployed WSN, involving the residual energy of nodes. Each node can be a sink and the sources and the traffic are not known apriori. The routing is achieved without employing additional routing messages and without having global knowledge about the network's topology. We assume that the sensor nodes and the communication links are reliable enough to relay the data packet along one path from source to the sink. The routing protocol consists of three phases:

P1: *Interest propagation and cost establishment* phase, where the sink broadcasts the interest. This phase includes the maintenance phase, with periodical refreshes.

P2: *Data transmission* phase, where sources send data packets, which are routed in a multihop fashion (along intermediate nodes) to the sink.

P3: *Reconfiguration* in case of transient failures.

P1) During the *interest propagation* phase **some cost** information (to the sink) must be established. Each node keeps in a gradient table one-hop candidate neighbors in the sink direction. Each network packet has a routing header consisting of several fields including the source node, the destination node, a sequence number, hop count, some energy fields and the initiator node (optional). When the request packet leaves the sink the cost field is set to 0, the energy level is set to the node's residual energy, the source and initiator addresses are set to the sink ID, the destination is set to a broadcast address and the sequence number is set to a unique number.

When a node receives the first time a request, it reads the relevant cost information and rebroadcasts a copy of the packet with the updated cost metrics. The node changes the source address and the energy level field to its node ID and its residual energy, respectively. Additionally, the node stores the sender ID and its residual energy in the gradient table (if this does not already exist). Whenever a node receives a copy of the packet leading to a smaller cost metrics, it resets its cost metrics and broadcasts again.

A node recognizes copies of the same packet by using a unique sequence number, which can be a combination of the initiator identifier and the current time (or sequence number). Each interest packet is discarded as soon as it is known (same sequence number) and does not bring any new information.

Finally, each node has determined its minimum cost to the sink and depending on the size of its gradient table, it knows a subset or all of its neighbors and their cost.

P2) The *data transmission* phase starts when a node concludes that it can deliver data matching the interest attributes. This node, referred as source, sends a data reply, which is routed to its best neighbor (the gradient) and so on hop by hop until the data reaches the sink. Each node is able to address the data packet (radio unicast) to next receiver, and only this receiver forwards the data further. The routing algorithm in each node is now able to adaptively select among several possible candidates, the one having the best cost. The routing header of the data does not change and is used in a similar way as for the interest. The initiator field is set to the source ID in order to inform the sink where the data comes from.

To let an intermediate node conclude that the data packet sent has reached the receiver, we use an implicit acknowledgment scheme. It is based on the omnidirectional radio signal property and exploits the fact that, when the intermediate node receives a packet and forwards it, the sender node can overhear the transmission and concludes (by inspecting only the header) that the transmission succeeded. If the sender node didn't hear the forwarded packet, it means the packet is possibly lost. Then the sender node either retransmits the packet up to a maximum number of retries or it sends the packet to another candidate intermediate node. In the latter case the node stores in the gradient entry of the failed intermediate information about the quality of the link, which can be used in the routing decision.

Some remarks considering both phases:

• Update routing information: To spread the routing information the (routing) protocol does not send extra routing messages but uses instead the interest to set-up routing information about the way back (return path) to sink. The routing information must be updated periodically to reflect the network state. Updates are also required to discover topology changes (i.e., new/depleted nodes) and ensure that the interest reaches all the nodes and was not lost. Therefore, the sink periodically broadcasts a copy of the original interest, referred as an interest refresh packet. Such a refresh updates cost information due to propagation failures (collision), node's energy reserve depletion or the addition of a new node. Moreover, the broadcast nature of the transmission medium allows to update some other information (e.g., node's residual energy, quality of the link) also during the data transmission phase (using the piggyback principle). The frequency of such updates depends on the sensor network application, where factors such as the numbers of sinks, the network's data traffic, and the dynamics of topology change play a decisive role.

• Adaptivity and alternative intermediate nodes: The routing protocol stores in its gradient table *several* candidate neighbors. This is advisable since in WSN, (transient) node and link failures are common; when nodes along the default route fail, providing alternative relay nodes (for each intermediate node) saves a lot of overhead, since a new route establishment process is not necessary. This enables each node to self-adapt its routing behavior to current network conditions.

Moreover, if the application requires that some critical data must reach the sink under any circumstances one can use redundancy by sending the critical data packet on more than one path (tradeoff between energy efficiency and reliability).

• Cost establishment during interest propagation: The interest and its periodical refreshes are used to spread and update the routing information into the network. Each intermediate node rebroadcasts these packets many times. If a node receives several copies of the interest (or refresh) consecutively, each of them leading to a smaller cost metrics, the node rebroadcasts the interest several times. Moreover, each interest copy at a node induces further updates and copies for next relay nodes. Thus, each relay node consumes more energy and excessive retransmissions can lead to congestion in the network.

The reason that a node broadcasts more than once is that it rebroadcasts immediately after receiving a lower cost, without knowing if this cost is minimal, so it may rebroadcast too early. A useful approach to this problem is to postpone the broadcast of each node by a delay time T_w in order to gain time for further cost messages. This delay time can be set to a constant value or chosen directly proportional to the cost at the node; in the latter case, a node with high cost will wait longer in order to be able to receive better costs.

• In-network processing: Aggregation is very useful to reduce the energy consumption of the nodes on the path and, additionally, it can reduce the traffic in the network avoiding in this way congestions and induced collisions.

P3) Recovery mechanisms: Since the interest is refreshed periodically (by broadcast) each node must receive a message in a given interval from its neighbors, since each neighbor rebroadcasts it at least once. But due to unreliable transmissions and collisions not all the messages reach their destination. Therefore, the interval until a message from a neighbor is expected is set larger. In our simulations, this interval was configured to three refresh rounds. Each time a message from a given neighbor arrives the timestamp of the entry is updated. If within the predefined interval the node does not receive any message (incl. SYNC frames at MAC) from a neighbor, it removes the neighbor from the cache tables.

Similarly, when a node is trying to transmit a message to a neighbor that does not react to it (acknowledgment or overhear its retransmission), the node increments a retries counter. If after a predefined number of tries the neighbor does not responds, the node concludes that it is damaged or depleted and removes it from cache. In such a case, the routing algorithm selects from its gradient table a next candidate node where to forward the data message.
New nodes are automatically inserted in the tables as soon as they rebroadcast an interest refresh.

Special mechanisms are necessary when some source becomes disconnected from the sink. In such situations the lack of the refresh, for a predefined number of rounds, stops or decreases the data generation rate at a source. Only when a new refresh is received the data generation is restarted. A similar approach can be used at the sink.
We discuss next two energy-aware strategies that we proposed in [15] and provide the corresponding distributed algorithms.

A. The hcE routing strategy

To get path information we consider a combination of the hop count and the residual energy of each node on the entire path. For that each node j computes the metrics $M(j) = \min\{M(i) + 1/E(j) | i \in Neighbor(j)\}$, where $E(j)$ is the residual energy of node j and $M(i)$ is the summation of the costs on the path from the sink up to and including node i.

Listing 1. Distributed cost establishment algorithm using the hcE metrics. Note: one can use the hop count also directly in the routing decision.

```
 1 Timer timer = Timer( "Tw", BCAST_DELAY_TIMER);
 2 ...
 3 if ( amISink() ) { hc=0;  M=0; // metrics
 4   broadcastPkt( pkt= (nodeId, hc, M) );
 5 } else { hc=INFINITY; M = INFINITY; }
 6 ...
 7 void handleEvent( Network *pkt)
 8 {
 9  case RECEPTION_EVENT: // received packet from i
10   receivePkt( i, hc(i), M(i), E);
11   // Store neighbor information
12   storeUpdateNeighborInCacheTables( i, M(i), hc(i));
13   if ( hc > hc(i) + 1 ) hc = hc(i) + 1;
14   if ( M > (M(i) + cost) ) {  // Compute better M
15    M = M(i) + 1/E; nextHop = i;  // record i as relay node
16    // create new packet and schedule its transmission
17    npkt = createPkt( nodeId, hc, M, E, ...);
18    if ( timer->isScheduled() )  cancelTimer( timer);
19    scheduleTimerAt( crtTime() + TWAIT * 1/E, timer); // Tw
20   }
21  case BCAST_DELAY_TIMER:  broadcastPkt( npkt ); ...
22 }
```

This additive metrics represents a quantitative characterization for the goodness of the entire route and balances the energy consumption of the network by redistributing the traffic load more uniformly on the nodes. The distributed cost field establishment algorithm is given in Listing 1.

To alleviate the problem of excessive broadcasts during flooding, caused by the fact that a node broadcasts instantly after receiving a lower cost without knowing whether this cost is minimal we introduce (in Listing 1, line 19) the waiting time T_w proportional to the cost between the receiver and sender (along the path the waiting time of each intermediate node will sum up, so at a node j T_w is proportional to $M(j)$).

The metrics used by hcE captures path information, but due to the summation it can still lead to special cases when a chosen path with small cost goes through a node with very low residual energy.

B. The hccE routing strategy

In order to avoid a situation as above we considered the hccE strategy, which uses a combined metrics involving both the hop count and the critical energy on the entire path. The intuition behind this strategy is that the bootleneck node's energy is propagated along, to be able to skip it if there are better paths (even longer ones).

Each node computes and forwards the pair: [distance to sink (in hops); critical energy on path], as $[hc(j); cE(j)] = [hc(i); cE(i)] \oplus [1; E(j)]$, where $[hc(i); cE(i)]$ is the hop count and critical energy pair corresponding to node $v = arg \min\{hc(i)/cE(i) | i \in Neighbor(j)\}$ and the operator $\oplus$ is defined for each term as $hc(j) = hc(i) + 1$ and $cE(j) = \min\{cE(i), E(j)\}$. The cost establishment algorithm is given in Listing 2.

Listing 2. Distributed cost establishment algorithm using the hccE metrics.

```
pair M = [INFINITY; E];     // E the node's initial energy
if ( amISink() ) {  M = [0; E];
 broadcastPkt( pkt= (nodeId, M, E));}
...
void handleEvent( Network *pkt ) {
 case RECEPTION_EVENT:  // received packet from node i
  receivePkt( i, M(i)=[hc(i);cE(i)], E(i) );
  storeUpdateNeighborInCacheTables( i, M(i), E(i) );
  Mcrt = add( M(i), [ 1; E] );

  if ( isNewCostBetter( M, Mcrt ) {
   M = Mcrt; nextHop = i;  // record i as relay node
   npkt = createPkt( nodeId, M, E, ...);
   if ( timer->isScheduled() )  cancelTimer( timer);
   scheduleTimerAt( crtTime()+ TWAIT*M.hc/M.cE, timer);//Tw
  }
 case BCAST_DELAY_TIMER:  broadcastPkt( npkt ); ...
}
pair add( pair M1, pair M2) {  // Addition of two costs
 pair Res; Res.hc=M1.hc + M2.hc; Res.cE= min(M1.cE, M2.cE);
 return Res;
}
bool isCostBetter( M1, M2) { // Comparison of costs
 return (M1.hc/M1.cE > M2.hc/M2.cE ); }
```

As can be seen, at each reception of a message from a node i carrying the pair $[hc(i); cE(i)]$ the receiver node records the pair in its cache, computes the new cost using the `add` method and compares it with its previous cost using the `isCostBetter` method. Hereby, we compare the ratios hc/cE for the receiver and sender node, since it is naturally to promote shorter paths (having the hop count in numerator) and paths with higher critical energy (having cE in denominator). After the comparison the (receiver) node propagates the better cost as a pair and the node that leads to the better cost is the preferred next hop. The algorithm is illustrated in Figure 1.

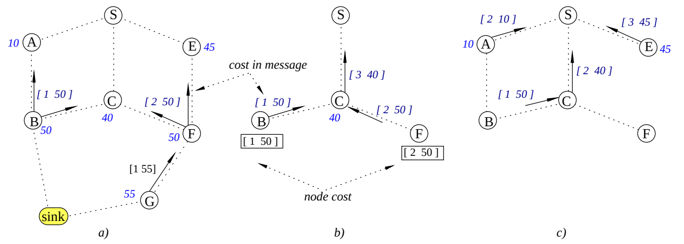

Fig. 1. Cost establishment process: a) Nodes G, B, and F send their cost; b) C receives first the packet from F and it broadcasts the cost $[3,\ 40]$; c) C receives a smaller cost from B and broadcasts again the cost $[2,\ 40]$. Nodes A and E send their cost. The source S knows three paths (along nodes A, C, E) and selects the path with the smaller cost (along C).

A node will wait for a time T_w, which is chosen directly proportional with the ratio hc/cE (computed from the pair $[hc; cE]$ of the sender an its local energy). During this period, the node computes from all received packets the minimal cost field and, if this is better than its own, it updates its local cost and resets the timer. When the timer expires, the node broadcasts the packet with its local cost. Finally some remarks:

- The strategies `hcE` and `hccE` capture path information inside their metrics. The `hcE` strategy uses an additive path cost function. Having a minimal path, it can be split (divided) in intermediate paths and all are minimal. The second metrics is not additive, but is a strictly monotonically increasing function.
- Setting the waiting time T_w depends on the underlying MAC protocol, especially if it has an adaptive duty-cycle such as the T-MAC. For the computation of T_w see §IV-B.

IV. SIMULATIONS RESULTS

We analyse the following performance parameters: the energy consumption and throughput.

We use the following general setting for the simulations. As MAC protocol we use our variant of T-MAC [16] with a listen time of $30ms$ and a frame time of $600ms$ and overhearing avoidance flag enabled (nodes in the NAV-state turn off their radio to save energy). The interest is refreshed each 5s and the simulation time is 180s. To configure the radio we used the CC2420 transmitter [17] with the following parameters:

current [mA]			**power** [mW]			
SL(sleep)	RX(receive)	TX(transmit)	SL	RX	TX	Switch
0.02	24	14	0.04	48	28	30

switching time [μs]					
$SL \rightarrow RX$	$SL \rightarrow TX$	$RX \rightarrow SL$	$TX \rightarrow SL$	$RX \rightarrow TX$	$TX \rightarrow RX$
580	580	10	10	580	580

The energy consumed according to the node's different states was multiplied with a factor of 10 to make the results sooner visible and to reduce the simulation running time.

We start with a network scenario where three sinks send a request for different network's zones and wait for data to be reported by one or several sources.

A. Impact of the routing strategy on the depletion time

We illustrate comparatively the impact of strategies on the time when the first three nodes run out of energy.
Settings: The three sinks are node 21, 24 and 41 (see Fig.2). The first two of them are placed in the bottom right corner and gather data from two zones placed on the left side, the upper one (the red rectangle) and the bottom one (the green rectangle) with three sources, respectively. The third sink, node 41, is placed on the buttom left side and gathers data from the opposite upper-right corner (the purple rectangle). Node 21 and 24 request data at each 400ms and node 41 at each 800ms (aggregation disabled) respectively. The interest is refereshed at 1s for the first sink and at 4s for the left two.

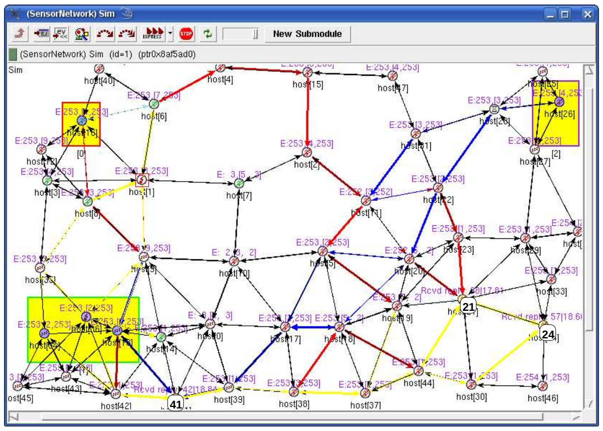

Fig. 2. Snapshot for the `hccE` strategy.

Since the routes from sources to sinks cross themselves, we set a very low initial energy for several nodes in the middle of the network: 700mJ (3eU) for node 10, and 1000mJ (5eU) for nodes 0 and 7. The energy of a node is converted in a scale between 0-255, which are called *energy units* (eU).

For space constraints we illustrate comparatively only the impact of the `hc` and `hccE` strategies on the nodes' depletion time. The `hc` strategy uses the shortest path between source and sink, meaning that the routes are along the low energy nodes (0,7 and 10). As explained in section §III-A, the `hccE` strategy avoids the low energy zone of the 3 nodes; the routes are either upper or lower as illustrated in Figure 2. To illustrate the routes that each data packet follows, we animated with a different color the links on which data packets are forwarded. In the figure the data packets for sink 21 travel on red paths, for sink 24 on yellow paths and for sink 41 on blue paths, respectively. Even though the three colors overwrite themselves when two data packets follow the same line (sometimes in opposite direction, e.g., node 1, 9, etc.) it is easy to identify during simulation the route that a packet follows to reach the corresponding sink.

Depletion time [s]	Nodes		
Strategy	Node 10	Node 0	Node 7
`hc`	42.62	53.56	73.84
`hccE`	54.64	63.60	85.20

Table I. Impact of high traffic and strategy on depletion time.

The depletion time result are given in Table I. The low energy nodes are sooner completely discharged in the case of `hc` since they are participating in forwarding the data. In the case of `hccE` the low energy reserve is consumed by the active phase of active-sleep regime of T-MAC. The time procentage gain is between 15% (node 7) and 28% (node 10).

We observed during simulation (with different seeds) that the `hccE` can occasionally route for short time a packet along a "bottleneck" node even though there are other paths. More about the causes of such a behavior in §IV-C.

B. The backoff waiting time

The metric establishment process takes place in the first phase and periodically at the rate of refreshes sent by the sink. A large number of rebroadcasts (especially when the refresh rate is high) impacts on the active time of a sensor node and the network traffic leading to a higher energy consumption. In order to overcome this problem we analyse the impact of various waiting times by using different strategies in a particular network topology (see Figure 3) with sensor nodes using an active-sleep regime.

Settings: The sink is node 21 and the rectangle zone contains one source, node 16, which generates data at each $300ms$. We set a very low initial energy for several nodes: 0.7mJ (3eU) for node 19 and 1J (5eU) for node 17. We have in this scenario paths of different length and due to low energy nodes we have various metrics depending on the chosen strategy. The simulation time is 180s and the interest refresh rate is 5s. That means that each node should broadcast at least 36 times, i.e., for 30 nodes this gives a total of 1080 times. Since the nodes 17 and 19 have very few energy, the total number of rebroadcasts is reduced to 1050 broadcasts (optimum), as these two nodes are broadcasting together no more than 30 times.

Rebroadcasts	Broadcast delay (T_w) [ms]			
Strategy	0	40	40*M	600*M
`hc`	1043 (5)	1044 (2)	1037 (2)	1036 (1)
`hcE`	1044 (4)	1043 (4)	1033 (1)	1023 (0)
`hccE`	1277 (3)	1252 (3)	1091 (4)	1050 (0)

Table II. Impact of T_w on the number of rebroadcasts.

The total number of rebroadcasts is given in Table II, including the number of missed refreshes (in parenthesis). To avoid side effects the source does not generate any data packets. We consider two fixed values for 0 and 40ms and two variable ones (chosen in accordance with the 30 ms listen time of T-MAC). The variable delay is achieved by multiplying the fixed delay with the metric computed by the current node as explained in section §III-A.

Note that for the `hc` and `hcE` strategies a fixed T_w has a low influence on the number of broadcasts. In the case of `hccE` strategy the number of broadcasts for fixed T_w is about 20% higher than for `hc`, as nodes rebroadcast often since here the metrics changes faster. Therefore, a fixed T_w is not recommended here and we use a variable one. When T_w is chosen proportional to the metrics (40*M, 600*M in Table II) the number of rebroadcasts improves considerable for the `hccE` strategy and reaches its optimum.

The same measurements with data traffic are given in Table III. Due to collisions not all rebroadcasts are received by all nodes, but the routing is not affected since all data packets reach the sink.

Rebroadcasts	Broadcast delay (T_w) [ms]			
Strategy	0	40	40*M	600*M
`hc`	1013 (4)	1020 (1)	1010 (4)	1010 (3)
`hcE`	1014 (3)	1011 (0)	1010 (2)	1011 (4)
`hccE`	1179 (7)	1111 (9)	1055 (7)	1043 (6)

Table III. Number of rebroadcasts with data traffic.

Thus, for a fixed broadcast delay the number of rebroadcasts is high and therefore by using `hccE` strategy a variable T_w larger than 40*M is recommended.

We illustrate in Table IV the impact of the broadcast delay (T_w) on the total energy consumption.

Energy [J]	without data			with data	
Strategy	0	40*M	600*M	40*M	600*M
`hc`	42.7	42.0	47.7	42.0	43.7
`hc/E`	43.8	42.7	44.2	42.9	43.4
`hccE`	45.1	43.5	45.3	43.8	44.5

Table IV. Energy consumption with data traffic.

As expected, for $T_w = 0$ the energy consumption for the `hc` strategy is a bit lower than for the `hcE` and `hccE` strategies (2.5% and 5.6%, respectively). This situation can change when using a $T_w > 0$. Although introducing an adjustable waiting time reduces the number of broadcasts, its impact in the energy consumption is not necessarily as expected in theory. This is due to the fact that nodes are spending more time in idle state, which leads to higher energy consumption.

For the `hccE` strategy the energy consumption decreases with a variable delay. For `hc` with a smaller adjustable waiting time the energy consumption decreases, but it increases for larger delay due to the T-MAC's aggressive time-out policy. Since T-MAC extends its listen period at each send/receive event, the total time the node is in idle state is longer for a 600ms delay than for a 40ms delay. This can be seen by means of our SNF when examining the transceiver states of the involved nodes (for place reason we omit the graphs here).

C. Behavior anomalies

We discuss next an example of unexpected behavior. We consider the special network scenario given in Figure 3, with the settings given in §IV-B. The depletion time of of nodes 17 and 19 are 78.04s and 70.83s for hc and 84.07s and 62s for hccE. It is expected that in the case of hccE strategy, paths along low energy node are avoided. The results show a gain of about 8% at node 17, but a lose of more than 4% at node 19. To explain this behavior we take a closer look to the simulation.

In the case of the hc the source 16 selects the minimal hop count route, thus a four hops paths, with next hop either 17 or 23 or 25. All routes are along low energy nodes (either 17 or 19). After a while both nodes are depleted, but due to different causes. Node 17 is depleted at 78.04s being a relay node in forwarding all data packets. Node 19 loses all its energy at 70.83 being passive, by following its active-sleep schedule imposed by T-MAC. After both nodes are down, the source selects the next shortest path namely 23-24-18-20-21.

In case of hccE strategy, by deferring the broadcast with a time proportional to the received metrics the optimal path 23-24-18-20-21 is not chosen. A random contention time in T-MAC (maximal 9ms) may cause that a broadcast propagates faster on a longer path than on a shorter one. Moreover, if a node goes to sleep, the broadcast is additionally delayed with the time of the sleep period. Different velocity of propagation of the broadcast, collisions, short term disconnection and transient failures are common in WSNs. The cumulative impact of all these factors is very difficult to be predicted, but using our simulation framework we identified three cases.

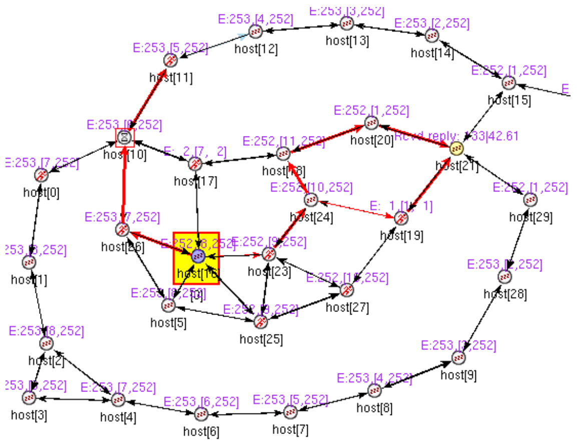

Fig. 3. Data forwarding on the path 26-10-11-12-...(lightblue arrow).

In the ideal case the minimal cost broadcast arrives in the required time or later. In the latter case the node should rebroadcast again, but the forwarding path is the optimal one.

In the suboptimal case the broadcast does not reach some relevant nodes on the optimal path between sink and source, e.g., 24 failed to receive from 18. A low energy node (19) acts as a relay node to forward the data; thus its energy decreases sooner. The source may receive a better routing cost (broadcast in the same round) from a node further from sink than the optimal one, that improves the routing information. In our scenario this happens when the broadcast on the upper path, along nodes 15-14-...-10-26, arrives and the source infers that the cost [7;252] through 26 is better than [3;2] along 17. In this way, the hccE strategy selects a suboptimal path 26-10-11-12-13-14-15-21 to forwards the data and avoids the nodes 17 and 19. For longer simulation time the lower route is selected, as soon as the metric on the upper path deteriorates.

The worst case happens when a suboptimal broadcast (of the same round) is unable to correct an already suboptimal routing information, e.g., paths along 26 are invisible since 26 failed to receive the broadcast from 10.

Even though such cases are rare, the hccE strategy cannot always avoid them and the simulation framework gives insights to find the causes for an initially unexplained behavior.

V. Conclusion

In this paper we supplemented our research [15] on energy-aware strategies randomly deployed WSNs, where multiple sinks are allowed, each node can be a sink and the sources and the traffic are not known apriori. We provided distributed routing algorithms that compute the next hop without employing additional routing messages. We further investigated the effects of introducing a broadcast delay on the number of messages and on the energy consumption. We presented simulation results that also give insights for unexpected behavior.

References

[1] W. Liang and Y. Liu, "On-line disjoint path routing for network capacity maximization in energy-constrained ad hoc networks," *Ad Hoc Networks Journal*, vol. 5, no. 2, pp. 272–285, 2007.

[2] M. Busse, T. Hänselmann, and W. Effelsberg, "Energy-efficient forwarding schemes for wireless sensor networks," in *Proc. Int. Symp. on WoWMoM*. New York, USA, June 2006, pp. 125–133.

[3] H. Hassanein and J. Luo, "Reliable energy aware routing in wsns," in *Proc. 2nd IEEE Workshop on DSSNS*, 2006, pp. 54–64.

[4] H. Nurul, M. Hossain, S. Yamada, E. Kamioka, and O.-S. Chae, "Cost-effective lifetime prediction based routing protocol for manet," in *Proc. of ICOIN*. Springer-Verlag Berlin, 2005, pp. 170–177.

[5] M. Maleki, K. Dantu, and M. Pedram, "Lifetime prediction routing in mobile ad-hoc networks," in *Proc. IEEE WCNC*. New Orleans, LA, USA, March 2003, pp. 1185–1190 (vol.2).

[6] J. Aslam, Q. Li, and D. Rus, "Three power-aware routing algorithms for sensor networks," *Wireless Comm. and Mob. Computing*, vol. 3, no. 2, pp. 187–208, 2003.

[7] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. 33rd Annual HICSS*. Washington, USA, 2000, pp. 3005–3014.

[8] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion a scalable and robust communication paradigm for sensor networks," in *Proc. ACM MobiCom*. Boston, MA, USA,, 2000, pp. 56–67.

[9] F. Ye, G. Zhong, S. Lu, and L. Zhang, "GRAdient Broadcast: A robust data delivery protocol for large scale sensor networks," *Wireless Networks, Springer, The Netherlands*, vol. 11, no. 2, pp. 285–298, 2005.

[10] F. Ye, A. Chen, S. Lu, and L. Zhang, "A scalable solution to minimum cost forwarding in large sensor networks," in *Proc. 10th Int. Conf. on Comp. Comm. and Networks*. Scottsdale, USA, 2001, pp. 304–309.

[11] C. Schurgers and M. Srivastava, "Energy efficient routing in wireless sensor networks," in *Proc. MILCOM on Comm. for Network-Centric Operations: Creating the Inform. Force*. Virginia, 2001, pp. 357–361.

[12] G. Halkes, T. Dam, and K. Langendoen, "Comparing energy-saving mac protocols for wireless sensor networks," *Mob. Netw. Appl.*, vol. 10, no. 5, pp. 783–791, 2005.

[13] T. Dam and K.Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proc. 1st Int. Conf. on Embedded Networked SenSys*. LA, California, USA, 2003, pp. 171–180.

[14] A. Kacsó and R. Wismüller, "A simulation framework for energy-aware wireless sensor network protocols," in *Proc. 18th Int. Conf. on Comp. Comm. and Networks*. San Francisco,CA,USA, August 2009, pp. 1–7.

[15] A. Kacsó, "Simulation of multihop energy-aware routing protocols in wireless sensor networks," *Int. Journal On Advances in Internet Technology*, vol. 3, no. 1&2, pp. 88–103, 2010.

[16] A. Kacsó and U. Schipper, "Receiver-based routing service for T-Mac protocol," in *Proc. 4th Int. Conf. on Sensor Technologies and Appl. (SensorComm)*. Venice, Italy, July 2010, pp. 489–494.

[17] *CC2420.pdf*, www-inst.eecs.berkeley.edu/∼cs150/Documents, 7.1.2011.