We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

1

A Modelling Framework for Energy Harvesting Aware Wireless Sensor Networks

Michael R. Hansen, Mikkel Koefoed Jakobsen and Jan Madsen
Technical University of Denmark, DTU Informatics, Embedded Systems Engineering
Denmark

1. Introduction

A Wireless Sensor Network (WSN) is a distributed network, where a large number of computational components (also referred to as "sensor nodes" or simply "nodes") are deployed in a physical environment. Each component collects information about and offers services to its environment, e.g. environmental monitoring and control, healthcare monitoring and traffic control, to name a few. The collected information is processed either at the component, in the network or at a remote location (e.g. the base station), or in any combination of these. WSNs are typically required to run unattended for very long periods of time, often several years, only powered by standard batteries. This makes energy-awareness a particular important issue when designing WSNs.

In a WSN there are two major sources of energy usage:

- Operation of a node, which includes sampling, storing and possibly processing of sensor data.
- Routing data in the network, which includes sending data sampled by the node or receiving and resending data from other nodes in the network.

Traditionally, WSN nodes have been designed as ultra low-power devices, i.e., low-power design techniques have been applied in order to achieve nodes that use very little power when operated and even less when being inactive or idle. By adjusting the duty-cycle of nodes, it is possible to ensure long periods of idle time, effectively reducing the required energy.

At the network-level nodes are equipped with low-power, low-range radios in order to use little energy, resulting in multi-hop networks in which data has to be carefully routed. A classical technique has been to find the shortest path from any node in the network to the base station and hence, ensuring a minimum amount of energy to route data. The shortest path is illustrated in Fig. 1. Fig. 1(b) shows the circular network layout, where the base station is labelled N_x. Fig. 1(a) is a bar-chart showing the distance (y-axis) from a node to the base station, the x-axis is an unfolding of the circular network, placing the base station, with a distance of zero, at both ends.

The routing pattern of a node in this network is based upon the distance from a node (e.g. N_c) and its neighbours (N_b and N_d) to the base station. The node N_c will route to the neighbour with the shortest distance to the base station (in this case N_b). In practice, nodes close to the base station (e.g. N_a and N_g) will be activated much more frequently than those far away from

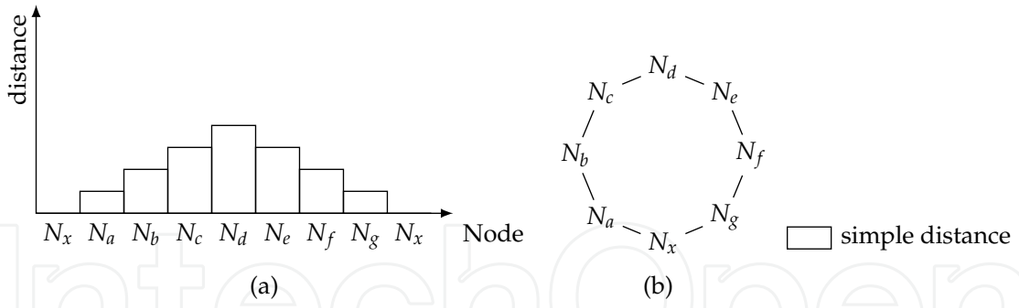


Fig. 1. An example network displaying the shortest distance to the base station. (a) shows each node's distance to the base station while (b) shows the placement of each node.

the base station, resulting in a relative short lifetime of the network. To address this, energy efficient algorithms, such as Bush et al. (2005); Faruque & Helmy (2003); Vergados et al. (2008), have been proposed. The aim of these approaches is to increase the lifetime of the network by distributing the data to *several* neighbours in order to minimize the energy consumption of nodes on the shortest path. However, these approaches do not consider the residual energy in the batteries. The energy-aware algorithms, such as Faruque & Helmy (2003); Hassanein & Luo (2006); Ma & Yang (2006); Mann et al. (2005); S.D. et al. (2005); Shah & Rabaey (2002); Xu et al. (2006); Zhang & Mouftah (2004), are all measuring the residual battery energy and are extending the routing algorithms to take into account the actual available energy, under the assumption that the battery energy is monotonically decreasing.

With the advances in energy harvesting technologies, energy harvesting is an attractive new source of energy to power the individual nodes of a WSN. Not only is it possible to extend the lifetime of the WSN, it may eventually be possible to run them without batteries. However, this will require that the WSN system is carefully designed to effectively use adaptive energy management, and hence, adds to the complexity of the problem. One of the key challenges is that the amount of energy being harvested over a period of time is highly unpredictable. Consider an energy harvester based on solar cells, the amount of energy being harvested, not only depends on the efficiency of the solar cell technology, but also on the time of day, local weather conditions (e.g., clouds), shadows from building, trees, etc.. For these conditions, the energy-aware algorithms presented above, cannot be used as they assume residual battery energy to be monotonically decreasing. A few energy harvesting aware algorithms have been proposed to address these issues, such as Islam et al. (2007); Lattanzi et al. (2007); Lin et al. (2007); Voigt et al. (2004; 2003); Zeng et al. (2006). They do not make the assumption of monotonically decreasing residual battery energy, and hence, can account for both discharging and charging the battery. Furthermore, they may estimate the future harvested energy in order to improve performance. However, these routing algorithms make certain assumptions that are not valid for multi-hop networks.

The clustering routing approach used in Islam et al. (2007); Voigt et al. (2004) assumes that all nodes are able to reach the base station directly. A partial energy harvesting ability is used in Voigt et al. (2003), where excess harvested energy can not be stored and the nodes are only battery powered during night. The algorithm in Lattanzi et al. (2007) is an offline algorithm, it assumes that the amount of harvestable energy can be predicted before deployment, which is not aa realistic assumption for most networks. The algorithm in Zeng et al. (2006) requires

that each node have knowledge of its geographic position. Global knowledge is assumed in Lin et al. (2007).

Techniques for managing harvested energy in WSNs have been proposed, such as Corke et al. (2007); Jiang et al. (2005); Kansal et al. (2007; 2004); Moser et al. (2006); Simjee & Chou (2006). These are focussing on local energy management. In Kansal et al. (2007) they also propose a method to synchronise this power management between nodes in the network to reduce latency on routing messages to the base station. They do, however, not consider dynamic routes as such. An interesting energy harvesting aware multi-hop routing algorithm is the REAR algorithm by Hassanein & Luo (2006). It is based on finding two routes from a source to a sink (i.e. the base station), a primary and a backup route. The primary route reserve an amount of energy in each node along the path and the backup route is selected to be as disjunct from the primary route as possible. The backup route does not reserve energy along its path. If the primary route is broken (e.g. due to power loss at some node) the backup route is used until a new primary and backup route has been build from scratch by the algorithm. An attempt to define a mathematical framework for energy aware routing in multi-hop WSNs is proposed by Lin et al. (2007). The framework can handle renewable energy sources of nodes. The advantage of this framework is that WSNs can be analyzed analytically, however the algorithm relies on the ideal, but highly unrealistic assumption, that changes in nodal energy levels are broadcasted instantaneously to all other nodes. The problem with this approach is that it assumes global knowledge of the network.

The aim of this chapter is to propose a modeling framework which can be used to study energy harvesting aware routing in WSNs. The capabilities and efficiency of the modeling framework will be illustrated through the modeling and simulation of a distributed energy harvesting aware routing protocol, Distributed Energy Harvesting Aware Routing (DEHAR) by Jakobsen et al. (2010). In Section 2 a generic modeling framework which can be used to model and analyse a broad range of energy harvesting aware WSNs, is developed. In particular, a conceptual basis as well as an operational basis for such networks are developed. Section 3 shows the adequacy of the modeling framework by giving very natural descriptions and explanations of two energy harvesting based networks: DEHAR Jakobsen et al. (2010) and Directed Diffusion (DD) Intanagonwiwat et al. (2002). The main ideas behind routing in these networks are explained in terms of the simple network in Fig. 1. Properties of energy harvesting aware networks are analysed in Section 4 using simulation results for DEHAR and DD. These results validate that energy harvesting awareness increase the energy level in nodes, and hence, keep nodes (which otherwise would die) alive, in the sense that a complete drain of energy in critical nodes can be prevented, or at least postpone. Finally, Section 5 contains a brief summary and concluding remarks.

2. A generic modelling framework

The purpose of this section is to present a generic modelling framework which can be used to study energy-aware routing in a WSN, where the nodes of the network have an energy harvesting capability. In the next section instantiations of this generic model will be presented and experimental results through simulations are presented in Section 4.

The main idea of establishing a generic framework is to have a conceptual as well as a tool-based fundament for studying a broad range of wireless sensor networks with similar characteristics. In the following we will assume that

- sensor nodes have an energy-harvesting device,
- sensor nodes are using radio-based communication, consisting of a transmitter and a receiver,
- sensor nodes are inexpensive devices with limited computational power, and
- the routing in the network adapts to dynamic changes of the available energy in the individual nodes, i.e. the routing is energy aware.

On the other hand, we will not make any particular assumptions about the kind of sensors which are used to monitor the environment.

These assumptions have consequences concerning the concepts which should be reflected in the modelling framework, in particular, concerning the components of a node. Some consequences are:

- A node may only be able to have a direct communication with a small subset of the other nodes, called its neighbours, due to the range of the radio communication.
- A node needs information about neighbour nodes reflecting their current energy levels in order to support energy-aware routing.
- A node can make immediate changes to its own state; but it can only affect the state of other nodes by use of radio communication.
- The processing in the computational units as well as the sensing, receiving and transmitting of data are energy consuming processes.

These assumptions and consequences fit a broad range of WSNs.

The components of a node

A node consists of five physical components:

- An *energy harvester* which can collect energy from the environment. It could be by the use of a solar panel – but the concrete energy source and harvesting device are not important in the generic setting.
- A *sensor* which is used to monitor the environment. There may be several sensors in a physical node; but we will not be concerned about concrete kinds in the generic setting and will (for simplicity) assume that one generic sensor can capture the main characteristics of a broad range of physical sensors.
- A *receiver* which is used to get messages from the network.
- A *transmitter* which is used to send messages to the network.
- A *computational unit* which is used to treat sensor data, to implement the energy-aware routing algorithm, and to manage the receiving and sending of messages in the network.

The model should capture that use of the sensor, receiver, transmitter and computational unit consume energy and that the only supply of energy comes from the nodes' energy harvesters. It is therefore a delicate matter to design an energy-aware routing algorithm because a risk is that the energy required by executing the algorithm may exceed the gain by using it.

A consequence of this is that exact energy information cannot be maintained between nodes because it requires too much communication in the network as that would imply that too much energy is spent on this administrative issue compared to the harvested energy and the energy used for transmitting sensor-observations from the nodes to the base station.

The identity of a node

We shall assume that each node has a unique identification which is taken from a set Id of identifiers.

The state of a node

The *state* of a node is partitioned into a *computational state* and a *physical state*. The physical state contains a model of the real energy level in the node as well as a model of the dynamics of energy devices, like, for example, a capacitor. The computational state contains an approximation of the physical energy model, including at least an approximation of the energy level. The computational state also contains routing information and an abstract view of the energy level in neighbour nodes. Furthermore, the computational state could contain information needed in the processing of observations, but we will not go into details about that part of the computational state here, as we will focus on energy harvesting and energy-aware routing.

We shall assume the existence of the following sets (or types):

- PhysicalState – which models the real physical states of the node,
- Energy – which models energy levels,
- ComputationalState – which models the state in the computational unit in a node, including a model of the view of the environment (especially the neighbours) and information about the energy model and the processing of observations, and
- AbstractState – which models the abstract view of a computational state. An abstract state is intended to give a condensed version of a computational state and it can be communicated to neighbour nodes and used for energy-aware routing. It is introduced since it is too energy consuming to communicate complete state information to neighbours when radio communication is used.

The state parts of a node may change during operation. The concrete changes will not be described in the generic framework, where it is just assumed that they can be achieved using the functions specified in Fig. 2. Notice that a node can change its own state only.

Sets: PhysicalState, ComputationalState, AbstractState, and Energy

Operations:

consistent?	: ComputationalState $\rightarrow \{\text{true}, \text{false}\}$
abstractView	: ComputationalState $\rightarrow$ AbstractState
updateEnergyState	: ComputationalState $\times$ Energy $\rightarrow$ ComputationalState
updateNeighbourView	: ComputationalState $\times$ Id $\times$ AbstractState $\rightarrow$ ComputationalState
updateRoutingState	: ComputationalState $\rightarrow$ ComputationalState
transmitChange?	: ComputationalState $\times$ ComputationalState $\rightarrow \{\text{true}, \text{false}\}$
next	: ComputationalState $\rightarrow$ Id

Fig. 2. An signature for operations on the computational state

The intuition behind each function is given below. A concrete definition (or implementation) of the functions must be given in an instantiation of the generic model.

- consistent?(cs) is a predicate which is true if the computational state *cs* is *consistent*. Since neighbour and energy information, which are used to guide the routing, are changing dynamically, a node may end up in a situation where no neighbour seems feasible as the next destination on the route to the based station. Such a situation is called inconsistent, and the predicate consistent?(cs) can test for the occurrences of such situations.
- abstractView(cs) gives the abstract view of the computational state *cs*. This abstract view constitutes the part of the state which is communicated to neighbours.
- updateEnergyState(cs, e) gives the computational state obtained from *cs* by incorporation of the actual energy level *e*. The resulting computational state may be inconsistent.
- updateNeighbourView(cs, id, as) gives the computational state obtained from *cs* by updating the neighbour knowledge so that *as* becomes the abstract state of the neighbour node N_{id}. The resulting computational state may be inconsistent.
- updateRoutingState(cs) gives the computational state obtained from *cs* by updating the routing information on the basis of the energy and neighbour knowledge in *cs* so that the resulting state is consistent.
- transmitChange?(cs, cs') is a predicate which is true if the difference between the two computational states are so significant that the abstract view of the "new state" should be communicated to the neighbours.
- next(cs) gives, on the basis of the computational state *cs*, the identifier of the "best" neighbour to which observations should be transmitted.

The computation costs

Each of the above seven functions in Fig. 2 are executed on the computational unit of a node. Such an execution will consume energy and cause a change of the physical state. For simplicity, we will assume that the cost of executing the predicates consistent? and transmitChange? can be neglected or rather included in other functions, since they always incurs the same energy cost in these functions. These functions are specified in Fig. 3.

costAbstractView	: PhysicalState $\rightarrow$ PhysicalState
costUpdateEnergyState	: PhysicalState $\rightarrow$ PhysicalState
costUpdateNeighbourView	: PhysicalState $\rightarrow$ PhysicalState
costUpdateRoutingState	: PhysicalState $\rightarrow$ PhysicalState
costNext	: PhysicalState $\rightarrow$ PhysicalState

The costs of the predicates consistent? and transmitChange? are assumed negligible.

Fig. 3. An signature for cost operations on the computational state

For simplicity it is assumed that execution of each of the five functions have a constant energy consumption, so that all functions have the type PhysicalState $\rightarrow$ PhysicalState. It is easy to make this model more fine grained. For example, if the cost of executing abstractView depends on the computational state to which it is applied, then the corresponding cost function should have the type: PhysicalState $\times$ ComputationalState $\rightarrow$ PhysicalState. This level of detail is, however, not necessary to demonstrate the main principles of the framework.

Input events of a node

The computational unit in a node can react to *events* originating from the energy observations on the physical state, e.g. due to the harvesting device, the sensor and the receiver. There are two energy related events, where one is concerned with the change of the physical state while the other is concerned with reading the energy level in the node. The rationale for having two events rather than a "combined" one is that the change of the physical state is a cheap operation which does not involve a reading nor any other kind of computation, whereas a reading of the energy level consumes some energy.

A sensor recording results in an observation o belonging to a set Observation of observations. An observation could be temperature measurement, a traffic observation or an observation of a bird – but the concrete kind is of no importance in this generic part of the framework.

The events are described as follows:

- readEnergyEvent(e, ps), where $e \in$ Energy and $ps \in$ PhysicalState, which is an event signalling a reading e of the energy level in the node and a resulting physical state ps, which incorporates that the reading actually consumes some energy.
- physicalStateEvent(ps), where $ps \in$ PhysicalState is a new physical state. This event occurs when a change in the physical state is recorded. This change may, for example, be due to energy harvesting, due to a drop in energy level, or due to some other change which could be the elapse of time.
- observationEvent(o, ps), where $o \in$ Observation is a recorded sensor observation and $ps \in$ PhysicalState is a physical state which incorporates the energy consumption due to the activation of the sensor.
- receiveEvent(m, ps), where $ps \in$ PhysicalState and $m \in$ Message, which could be an observation to be transmitted to the base station or a message describing the state of a neighbour node. Further details are given below. The receiver maintains a *queue of messages*. When it records a new message, that message is put into the queue. The event receiveEvent(m, ps) is offered when m is the front element in the queue. Reacting to this event will remove m from the queue and a new receive event will be offered as long as there are messages in the queue. It is unspecified in the generic setting whether there is a bound on the size of the queue.

Input messages

A node has a queue of messages received from the network. There are two kinds of messages:

- *Observation Messages* of the form obsMsg(dst, o), where dst is the identity of the next destination of the observation $o \in$ Observation on the route to the base station.
- *Neighbour Messages* of the form neighbourMsg(src, as), where src is the identity of the source, i.e. the node which have sent this message, and $as \in$ AbstractState is the contents of the message in the form of an abstract state.

Let Message denote the set of all messages, i.e. observation and neighbour messages.

Output messages and communication

A node N_{id} can use the transmitter to broadcast a message $m \in$ Message to the network using the command $\texttt{send}_{id}(m)$. Intuitively, nodes which are within the range of the transmitter will receive this message and this may depend on the strength of the signal, it may depend on geographical positions, or on a variety of other parameters.

A model for sending and receiving messages could include a *global trace* of the messages send by nodes, a *local trace* of messages received by the individual nodes, and a description of a *medium*, that determines which nodes can receive messages sent by a node N_{id} on the basis of the current state of the network and on the basis of the various parameters, for example, concerning geographical positions of the nodes. In instances of the generic model, such a medium must be described. In this chapter we will not be formal about network communication. A formal model of communication along the lines sketched above can be found in Mørk et al. (1996); Pilegaard et al. (2003).

The cost of sending messages

Sending a message consumes energy which is reflected in a change of the physical state of a node. To capture this a function

$$\text{costSend} : \text{PhysicalState} \times \text{Message} \rightarrow \text{PhysicalState}$$

can compute a new physical state on the basis of the current one and a broadcasted message.

An operational model of a node

During its lifetime, a node can change between two main *phases*: *idle* and *treat message*.

- The node is basically inactive in the idle phase waiting for some event to happen. It processes an incoming event and makes a phase transition.
- The node treats a single message in the treat message phase and after that it makes a transition to the idle phase.

Each phase is parameterrised by the computational state *cs* and the physical state *ps*. The state changes and phase transitions for the idle phase are given in Fig. 4. The node stays inactive in the idle phase until a event occurs.

- A physical-state event leads to a change of physical state while staying in the idle phase.
- A read-energy event leads to an update of the energy and routing parts of the computational state, and the physical state is updated by incorporation of the corresponding costs. If the changes of the computational state are insignificant then these changes are ignored (so that the nodes have a consistent knowledge of each other) and just the physical state is changed. Otherwise, the abstract view of the new computational state is computed and send to the neighbours, and both the computational and the physical states are changed.
- An observation event leads to a computation of the next node (destination) to which the observation should be transmitted on the route to the base station, and a corresponding observation message is sent. The physical state is changed with the cost of computing the destinations and the cost of sending a message while staying in the idle phase.

$$
\begin{array}{l}
\mathrm{Idle}_{id}(cs, ps) = \\
\quad \texttt{wait} \\
\qquad \mathrm{physicalStateEvent}(ps') \ \rightarrow \ \mathrm{Idle}_{id}(cs, ps') \\
\qquad \mathrm{readEnergyEvent}(e, ps') \ \rightarrow \\
\qquad\quad \texttt{let}\ cs' = \mathrm{updateRoutingState}(\mathrm{updateEnergyState}(cs, e)) \\
\qquad\quad \texttt{let}\ ps'' = \mathrm{costUpdateEnergyState}(\mathrm{costUpdateRoutingState}(ps')) \\
\qquad\quad \texttt{if}\ \mathrm{transmitChange?}(cs, cs') \\
\qquad\quad \texttt{then let}\ m = \mathrm{neighbourMsg}(id, \mathrm{abstractView}(cs')) \\
\qquad\qquad\quad \texttt{send}_{id}(m); \mathrm{Idle}_{id}(cs', \mathrm{costSend}(\mathrm{costAbstractView}(ps''), m)) \\
\qquad\quad \texttt{else}\ \mathrm{Idle}_{id}(cs, ps'') \\
\qquad \mathrm{observationEvent}(o, ps') \ \rightarrow \\
\qquad\quad \texttt{let}\ dst = \mathrm{next}(cs) \\
\qquad\quad \texttt{let}\ m = \mathrm{obsMsg}(dst, o) \\
\qquad\quad \texttt{send}_{id}(m); \mathrm{Idle}_{id}(cs, \mathrm{costSend}(\mathrm{costNext}(ps'), m)) \\
\qquad \mathrm{receiveEvent}(m, ps') \ \rightarrow \ \mathrm{TreatMsg}_{id}(m, cs, ps')
\end{array}
$$

Fig. 4. The Idle Phase

- A receive event indicates a pending message in the queue. That message is treated by a transition to the treat message phase.

Notice that all phase transitions from the idle phase preserve the consistency of the computational state. The only non-trivial transition to check is that from $\mathrm{Idle}_{id}(cs, ps)$ to $\mathrm{Idle}_{id}(cs', \mathrm{costSend}(\mathrm{costAbstractView}(ps''), m)$. The consistency of cs' follows since $cs' = \mathrm{updateRoutingState}(\mathrm{updateEnergyState}(cs, e))$ and updateRoutingState is expected to return a consistent computational state, at least under the assumption that cs is consistent.

The state changes and phase transitions for the treat message phase are given in Fig. 5. In this phase the node treats a single message. After the message is treated a transition to the idle phase is performed, where it can react to further events including the receiving of another message. A message is treated as follows:

- An observation message is treated by first checking whether this node is the destination for the message. If this is not the case, a direct transition to the idle phase is performed. Otherwise, the next destination is computed, the observation is forwarded to that destination and the physical state is updated taking the computation costs into account. The energy consumed by the test whether to discard or process a message is included in the energy consumption for receiving a message.
- A neighbour message must cause an update of the neighbour view part of the computational state giving a new state cs'. A new routing state cs'' must be computed. If the changes to the computational state is insignificant (in the sense $\mathrm{transmitChange?}(cs, cs'')$ is false and cs' is consistent), then a transition to the idle phase is performed with a computational state that is just updated with the new neighbour knowledge, and the physical which is updated by the computation cost. Otherwise, an abstract view of the computational state must be communicated to the neighbours, and the computational and the physical states are updated accordingly.

```
TreatMsg_id(m, cs, ps) =
    case m of
        obsMsg(dst, o) →
            if id = dst
            then let dst' = next(cs)
                     let m' = obsMsg(dst', o)
                     send_id(m'); Idle_id(cs, costSend(costNext(ps), m))
            else Idle_id(cs, ps))

        neighbourMsg(src, as) →
            let cs' = updateNeighbourView(cs, src, as)
            let cs'' = updateRoutingState(cs')
            let ps' = costUpdateNeighbourView(costUpdateRoutingState(ps))
            if transmitChange?(cs, cs'') ∨ ¬consistent?(cs')
            then let as' = abstractView(cs')
                     let m = neighbourMsg(id, as')
                     send_id(m); Idle_id(cs'', costSend(costAbstractView(ps'), m))
            else Idle_id(cs', ps')
```

Fig. 5. The Treat-Message Phase

Notice that all phase transitions from the treat-message phase preserve the consistency of the computational state. The consistency preservation due to observation messages is trivial. The transition from $\text{TreatMsg}_{id}(m, cs, ps)$ to $\text{Idle}_{id}(cs'', \text{costSend}(\text{costAbstractView}(ps'), m))$ preserves consistency since cs'' is constructed by application of updateRoutingState, and this function is expected to return a consistent computational state. The transition from $\text{TreatMsg}_{id}(m, cs, ps)$ to $\text{Idle}_{id}(cs', ps')$ also preserves consistency since that transition can only occur when the `if`-condition $\text{transmitChange?}(cs, cs'') \vee \neg\text{consistent?}(cs')$ is false.

Some of the main features of the operational descriptions in Fig. 4 and Fig. 5 are:

- A broad variety of instances of the operational descriptions can be achieved by providing different models for the sets and operations in Fig. 2 and Fig. 3. This emphasizes the generic nature of the model.
- The energy and neighbour parts of the model appear explicitly through the occurrence of the associated operations. Hence it is clear that the model reflects energy-aware routing using neighbour knowledge, and it is postponed to instantiations of the model to describe how it works.
- The energy cost model appears explicit in the form of the cost functions including the cost of events.
- A node will send a local view of its state to the neighbours only in the case when a significant change of the computational state has happened, which is determined by the transmitChange? predicate. The adequate definition of this predicate is a prerequisite for achieving a proper routing, as it is not difficult to imagine how it could load the network and drain the energy resources, if minimal changes to the states uncritically are broadcasted.
- The model is not biased towards a particular energy harvester and it is not biased towards and particular kind of sensor observation.

The generic model is based on the existence of a description of the medium through which the nodes communicates. This medium should at least determine which nodes can receive a message send by a given node in a given state. It may depend on the available energy, the geographical position, the distance from the sender, and a variety of other parameters. Furthermore, the medium may be unreliable so that messages may be lost.

The model describes the operational behavior (including the dynamics of the energy levels in the nodes) for the normal operation of a network. It would be natural to extend the model with an initialization phase where a node through repeated communications with the neighbours are building up the knowledge of the environment needed to start normal operations, i.e. making observations and routing them to the base station. We leave out this initialization part in order to focus on energy harvesting and energy-aware routing.

3. Instantiating the modelling framework

In this section it will be demonstrated that the energy-aware routing protocol DEHAR Jakobsen et al. (2010) can be considered as an instance of the generic modelling framework presented in the previous section. In order to do so, meaning must be given to the sets and operations collected in Fig. 2 and Fig. 3. This will provide a succinct presentation of the main ideas behind DEHAR. Furthermore, we will show that the DD protocol Intanagonwiwat et al. (2002) can be considered a special case of DEHAR. Concrete experiments, based on a simulation framework, depends on descriptions of the medium. This will be considered in Section 4.

3.1 A definition of the states

The abstract state comprises:

- A *simple distance* $d \in \mathcal{R}_{\geq 0}$ to the base station. This is described by a non-negative real number, where larger number means longer distance.
- An *energy-aware adjustment* $a \in \mathcal{R}_{\geq 0}$ of the distance for the route to the base station, where a larger distance means less energy is available.

Hence an abstract state is a pair $(d, a) \in \text{AbstractState}$, where

$$\text{AbstractState} = \mathcal{R}_{\geq 0} \times \mathcal{R}_{\geq 0}$$

For an abstract state (d, a), we call $\text{dist}(d, a) = d + a$ the *energy-adjusted distance*.

The computational state comprises:

- A *simple distance* $d \in \mathcal{R}_{\geq 0}$ to the base station, like the simple distance of an abstract state.
- An *energy level* $e \in \text{Energy}$.
- An *energy-faithful adjustment* $f \in \mathcal{R}_{\geq 0}$ capturing energy deficiencies along the route to the base station.
- A table *nt* containing entries for the *abstract state of neighbours*. This is modelled by the type: $\text{Id} \to \text{AbstractState}$.

Hence a computational state is a 4-tuple $(d, e, f, nt) \in \text{ComputationalState}$, where

$$\text{ComputationalState} = \mathcal{R}_{\geq 0} \times \text{Energy} \times \mathcal{R}_{\geq 0} \times (\text{Id} \rightarrow \text{AbstractState})$$

We shall assume that there is a function energyToDist : Energy $\rightarrow \mathcal{R}_{\geq 0}$ that converts energy to a distance so that less energy means longer distance.

The value energyToDist(e) provides a local adjustment of the distance to the base station by just taking the energy level in the node into account. The intension with the energy-faithful adjustment is that the energy deficiencies along the route to the base station is taken into account, and the energy-faithful part is maintained by the use of the neighbour messages.

The *energy adjustment of a computational state* is the sum of the converted energy and the energy-faithful adjustment:

$$\text{adjust}(d, e, f, nt) = \text{energyToDist}(e) + f$$

and the energy-adjusted distance of a computational state is:

$$\text{dist}(d, e, f, nt) = d + \text{adjust}(d, e, f, nt) = d + \text{energyToDist}(e) + f$$

where we overload the dist function to be applied to both abstract and computational states. Furthermore, dist(id), $id \in \text{Id}$, is the distance of the abstract state of the neighbour node N_{id}.

The function next : ComputationalState $\rightarrow$ Id should give the neighbour with the shortest energy-adjusted distance to the base station, i.e. the "best" neighbour to forward an observation. Hence, next(d, e, f, nt) is the identity id of the entry $(id, as) \in nt$ with the smallest energy-adjusted distance to the base station, i.e. the smallest dist(as). If several neighbours have the smallest distance an arbitrary one is chosen.

A computational state cs is consistent if next(cs) has a smaller energy-adjusted distance than cs, i.e. dist$(cs) >$ dist(next(cs)), hence

$$\text{consistent?}(cs) = \text{dist}(cs) > \text{dist}(\text{next}(cs))$$

A node with a consistent computational state has a neighbour to which it can forward an observation. But if the state is inconsistent, then all neighbours have longer energy-adjusted distances to the base station and it does not make sense to forward an observation to any of these neighbours.

We illustrate the intuition behind the adjusted distance using the example network example from Fig. 1. If the energy level in node N_e of this network is decreased, then the distance of N_e to the base station is increased accordingly (by the amount energyToDist(e)) as shown in Fig. 6. All nodes are still consistent; but in contrast to the situation in Fig. 1, the node N_d (in Fig. 6) has just one neighbour (N_c) with a shorter energy-adjusted distance to the base station.

Consider now the situation shown in Fig. 7 with energy adjustments for the nodes N_f and N_g. These adjustments make the node N_e inconsistent, since its neighbours N_d and N_f both have energy-adjusted distances which are longer than that of N_e. In the shown situation it would make no sense for N_e to forward observations to its "best" neighbour, which is N_f, since N_f would immediately return that observation to N_e since N_e is the "best" neighbour of N_f.

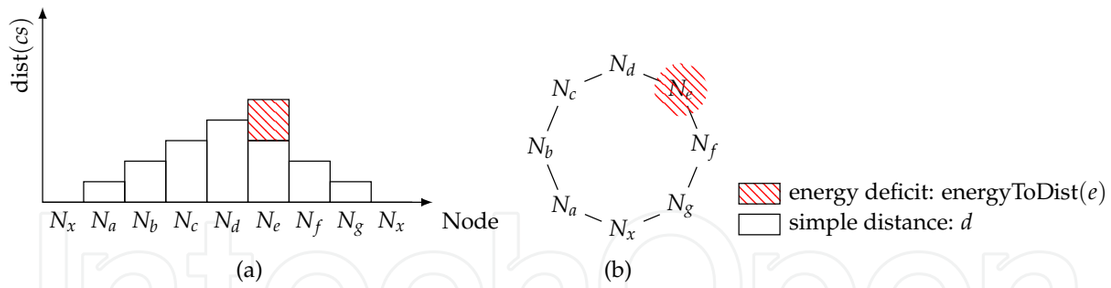


Fig. 6. The example from Fig. 1 with an energy adjustment for N_e due to shaded region shown to the right.

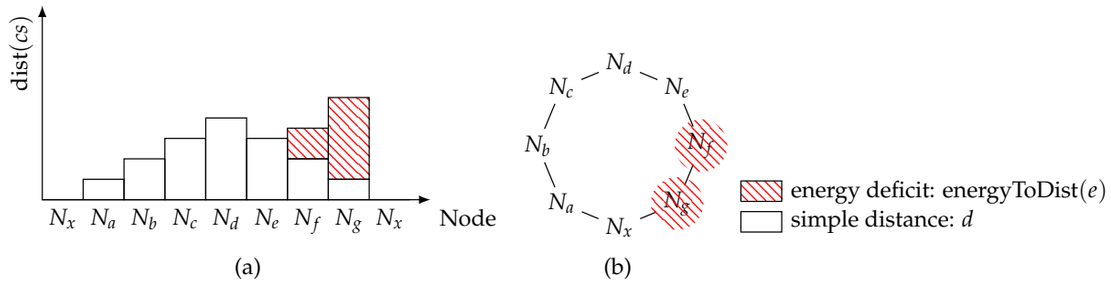


Fig. 7. Revised example with an inconsistent node: N_e.

Energy-faithful adjustments can be used to cope with inconsistent nodes. By adding such adjustments to the "problematic nodes" inconsistencies may be avoided. This is shown in Fig. 8, where energy-faithful adjustments (f) have been added to N_e and N_f. Every node is consistent, and there is a natural route from every node to the base station. From N_f there are actually two possible routes.

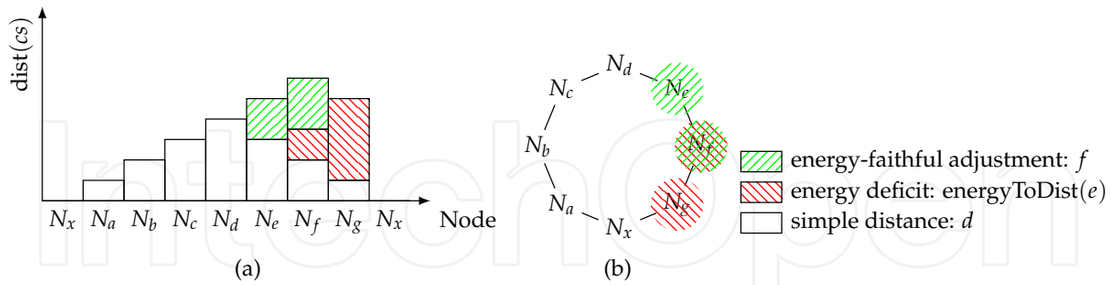


Fig. 8. A with consistent nodes using energy-faithful adjustments

The physical state comprises:

- The stored energy $e \in$ Energy.
- A model of the energy harvester. In the DEHAR case it is a solar panel, which is modelled by a function $P(t)$ describing the effect of the solar insolation at time t.
- A model of the energy store. In the DEHAR case it is an *ideal capacitor* with a given capacity. It is ideal in the sense that it does not loose energy.

- A model of the computational unit. This model must define the costs of the computational operations by providing definitions for the cost functions in Fig. 3. A simple way of doing this is to count the instructions needed for executing the individual functions, and multiply it with the energy needed per instruction. The model can be more fine grained by taking different modes of the processing unit into account.
- A model of the transmitter. This model must give a definition of the cost function: costSend : PhysicalState $\times$ Message $\rightarrow$ PhysicalState. In the DEHAR case the cost of sending is a simple linear function in the size of the message.
- A model of the receiver. This model must explain the cost of a receive event receiveEvent(m, ps). This involves the cost of receiving the message m and it must also take the intervals into account when the receiver is *idle listening*, i.e. it actively listens for incoming messages. Thus ps should reflect the full energy consumption of the receiver since the last receive event.
- A model of the sensor. This model must explain the cost of an observation event observationEvent(o, ps). This involves the cost of sensing o and ps should reflect this energy consumption.

The model should also describe two transitions of the physical state which relate to the two events physicalStateEvent(ps) and readEnergyEvent(e, ps).

The transition related to a physicalStateEvent must take into account at least the dynamics of the energy harvester, the dynamics of the energy store, the time the computational unit spent in the idle phase, and the time elapsed since the last physical state event. For example the new stored energy e' in the physical state at time t' is given by:

$$e' = e + \int_t^{t'} P(t)dt$$

where t is the time where the old energy e was stored.

The transition related to a readEnergyEvent(e, ps) must take into account at least the cost of reading the energy.

3.2 Definition of operations

The function for extracting the abstract view is defined by:

$$\text{abstractView}(d, e, f, nt) = (d, \text{adjust}(d, e, f, nt))$$

Notice that the distance to the base station is preserved by the conversion from a computational state to an abstract one:

$$\text{dist}(d, e, f, nt) = \text{dist}(\text{abstractView}(d, e, f, nt))$$

The definitions of the functions for updating the energy state and the neighbour view are simple:

$$\begin{aligned} \text{updateEnergyState}((d, e, f, nt), e') &= (d, e', f, nt) \\ \text{updateNeighbourView}((d, e, f, nt), id, as) &= (d, e, f, \text{update}(nt, id, as)) \end{aligned}$$

where update(nt, id, as) gives the neighbour table obtained from nt by mapping id to the abstract state as. These two operations may transform a consistent state into an inconsistent one.

The function updateRoutingState(d, e, f, nt) must update the energy adjustment of a computational state in order to arrive at a consistent one. If the state is consistent even when $f = 0$ then no adjustment is necessary. Otherwise, an adjustment is made so that the distance of the computational state becomes K larger than the distance of its "best" neighbour (given by the next function):

$$\begin{aligned}\text{updateRoutingState}(d, e, f, nt) = \\ &\texttt{if } \text{consistent?}(d, e, 0, nt) \\ &\texttt{then } (d, e, 0, nt) \\ &\texttt{else let } \text{distNext} = \text{dist}(\text{next}(d, e, f, nt)) \\ &\qquad (d, e, K + \text{distNext} - (d + \text{energyToDist}(e)), nt)\end{aligned}$$

where $K > 0$ is a constant used to enforce a consistent computational state.

The energy adjustment in the `else`-branch of this function has the effect that the node becomes less attractive to forward messages to in the case of an energy drop in the node or in the best neighbour.

The function transmitChange?(cs, cs') is a predicate which is true when a change of the computational state from cs to cs' is significant enough to be communicated to the neighbours. This is the case if the change reflects a significant change in distance to base station, where significant in this case means larger than some constant $K_{\text{change}} \in \mathcal{R}_{\geq 0}$.

Hence, the function can be defined as follows:

$$\text{transmitChange?}(cs, cs') = |\text{dist}(cs) - \text{dist}(cs')| > K_{\text{change}}$$

A simple check of the operational descriptions in Fig. 4 and Fig. 5 shows that the new computational state used as argument to transmitChange? (cs' in Fig. 4 and cs'' in Fig. 5) must be consistent as it is created using updateRoutingState. Hence it is just necessary to define transmitChange? for consistent computational states.

Directed Diffusion – another instantiation of the generic framework

It should be noticed that the routing algorithm DD Intanagonwiwat et al. (2002) is a simple instance of the generic framework, which can be achieved by simplifying the DEHAR instance so that

- the simple distance is the number of hops to the based station (as for DEHAR) and
- the energy is assumed perfect and hence the adjustments have no effect (are 0).

Hence DD do not support any kind of energy-aware routing.

Actually, it is the algorithm behind DD which is used to initialize the simple distances of nodes in the DEHAR algorithm.

The DD algorithm provides a good model of reference for comparison with energy harvesting aware routing algorithms like DEHAR, since DD incorporates nodes with an energy

harvesting capability, but the routing is static in the sense that an observation is always transmitted along the path with the smallest number of hops to the base station. Energy harvesting aware routing algorithms will not necessarily choose this shortest path, since problematic low-energy nodes should be avoided in order to keep all nodes "alive" as long as possible. Therefore, the total energy consumption in a DD based network should be smaller than the total energy consumption of any energy harvesting aware network (due to longer pathes in the latter). On the other hand, energy harvesting awareness can spare low-energy nodes, and there are two important consequences of this:

- A drain of low-energy nodes can be avoided or at least postponed. With regard to this aspect DD should perform worse since these nodes are not spared at all in the routing.
- The total energy stored in a network should exceed that of a corresponding DD based network, since messages are transmitted through nodes with good energy harvesting capability. The reason for this is that low-energy nodes get a chance to recover and that transmissions through high-energy nodes, with a full energy storage, are close to be "free of charge" since there would be almost no storage available for harvested energy in these high-energy nodes.

4. Results from simulation of the model

In this section we will study the properties of the energy harvesting aware routing algorithm DEHAR by analyzing results Jakobsen et al. (2010) of a simulator implementing the DEHAR and DD algorithms. The simulator is a custom-made simulator Jakobsen (2008) implemented in the language Java. It can be configured through a comprehensive xml configuration file which includes the network layout, environmental properties (insolation, shadows, etc.) and properties of nodes (such as processor states, radio model, and frequency of observations). The simulator features a classic event driven engine. The simulator produces a trace of observations of the nodes, including energy levels, activity of devices, and environmental properties

The considered network is given in Fig. 9. The network has one very problematic node, due to a strong shadow, at coordinate $(1,3)$, and five nodes with potential problems due to light shadows. We will analyse the ability of the routing algorithms to cope with these problematic nodes using simulations.

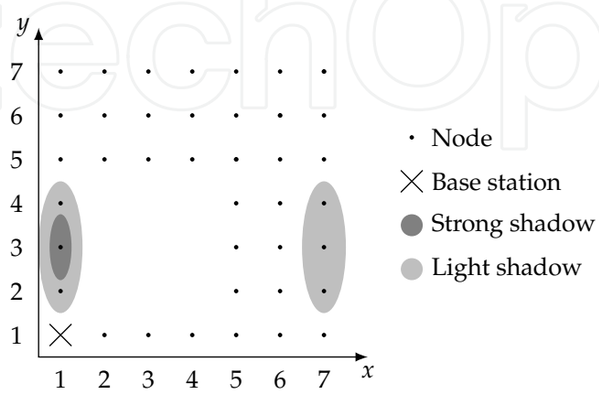


Fig. 9. A network structure with illustrating problematic nodes

The medium and the physical setting must be defined for the experiments. It is assumed that a node can communicate with its immediate horizontal and vertical neighbour, i.e. the radio range is 1. Two experiments $S1$ and $S2$ are conducted, one with a low and another with a high rate of conducted observations. Table 1 shows the parameters that are used in the presented simulations. Only the observation rate is changed between the two simulations.

			S_1	S_2	unit
Radio	Range		1	1	
	Transmit power		50	50	mW
	Idle listening power		5.5	5.5	mW
	Bandwidth		45	45	kb/s
Processor	Sleep	Power	1	1	μW
	Active	Frequency	1	1	MHz
		Power	10	10	μW
Battery	Capacity		4	4	kJ
Solar panel	Efficiency		6.25	6.25	%
	Area		12.5	12.5	cm^2
Application parameters	Observation rate		$\frac{1}{900}$	$\frac{1}{60}$	sec^{-1}
Routing parameters	Sense rate		$\frac{1}{1800}$	$\frac{1}{1800}$	sec^{-1}

Table 1. Parameters used in simulations.

The energy model is based on real insolation data for a two-weeks period. The data is repeated in simulations over longer periods. To emphasize the effect of the DEHAR algorithm, the insolation pattern have been idealised to either full noon or midnight, i.e. 12 hours of light and 12 hours of darkness. The insolation data is suitably scaled for individual nodes to achieve the shadow effect shown in Fig. 9.

Energy awareness makes a difference

A 30 day view of the simulations S_1 with the low observation rate is shown in Fig. 10. The figure shows the energy available in the worst node with minimum energy in the network. The two algorithms cannot be distinguished the first five days. Thereafter, the energy aware routing starts and DEHAR stabilises at a high level where no node is in any danger of being drained for energy. In the DD case, the energy of worst node is steadily drained at a (rather) constant rate and in an foreseeable future it will stop working.

Energy awareness consumes and stores more energy

The total power consumed and the average energy stored per node in the network are monitored for the same simulations as in Fig. 10. These results are shown for the first 10 days of simulated time in Fig. 11.

The day cycle is clearly visible in Fig. 11(a) where the nodes recharge during day and discharge during night. The first five days of simulation does not show any significant difference between DEHAR and DD. During the last five days the DEHAR algorithm makes the network able to harvest and store more energy.

The next graph (Fig. 11(b)) shows the difference of the two curves from the previous. It shows (in the blow-up) that just before day five ends, the DEHAR algorithm starts to consume

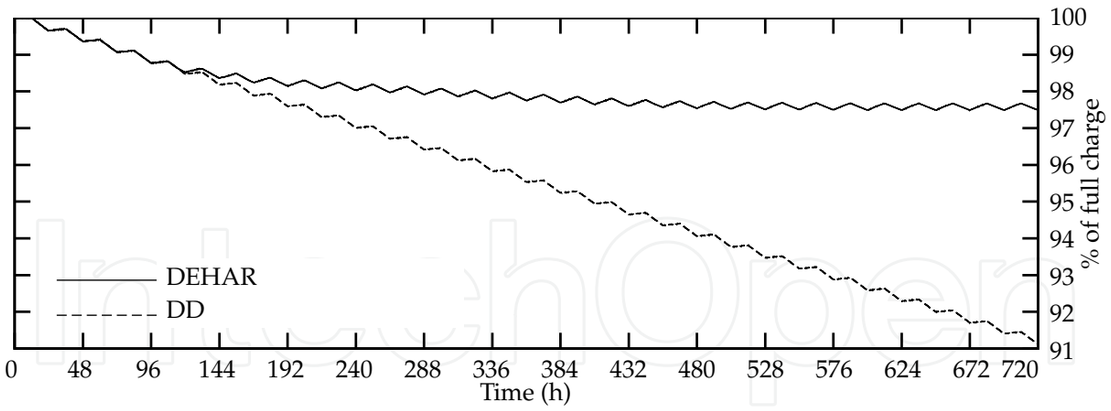


Fig. 10. Results of simulations S_1 for a 30 day simulation. This graph shows the minimum energy in any node in the network.

significantly more energy than the DD algorithm. By looking at the third graph (Fig. 11(c)) which shows the difference in total network energy consumption, it can be confirmed. This extra energy consumption arises from observation packages that travel along longer routes in the network, because the DEHAR algorithm have detected a lower amount of stored energy in some nodes.

Even though the DEHAR consumes more energy due to the longer routes, it can store more energy on average in the nodes. The reason for this is that the extra energy consumption of DEHAR is taken from nodes that are able to recharge fully during daytime. This can be seen in Fig. 11(b) (in the blow-up) at the beginning of day 5 (120h), where the graph shows a sudden rise.

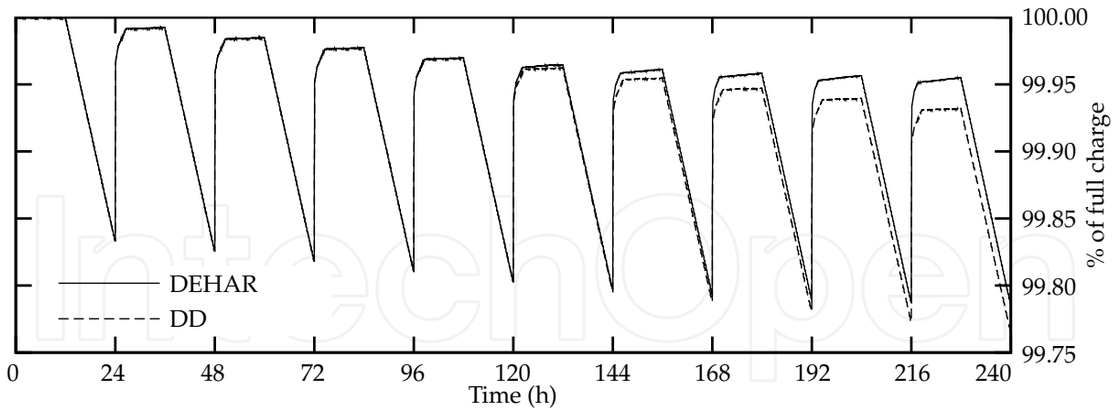
After a short while, the network with the DD algorithm is able to harvest energy at a greater rate than DEHAR. This is due to the fact that the majority of the nodes in the DEHAR network are fully charged. The key point at this time is that the DD algorithm does not allow the network to harvest as much energy as the DEHAR algorithm. This can also be seen through the rest of the daylight during day 5, where the DEHAR network is able to harvest energy at a higher rate than the DD network.

Finally, during night, the DEHAR network again shows a higher energy consumption than the DD network. Hence the graph shows a slow decline.

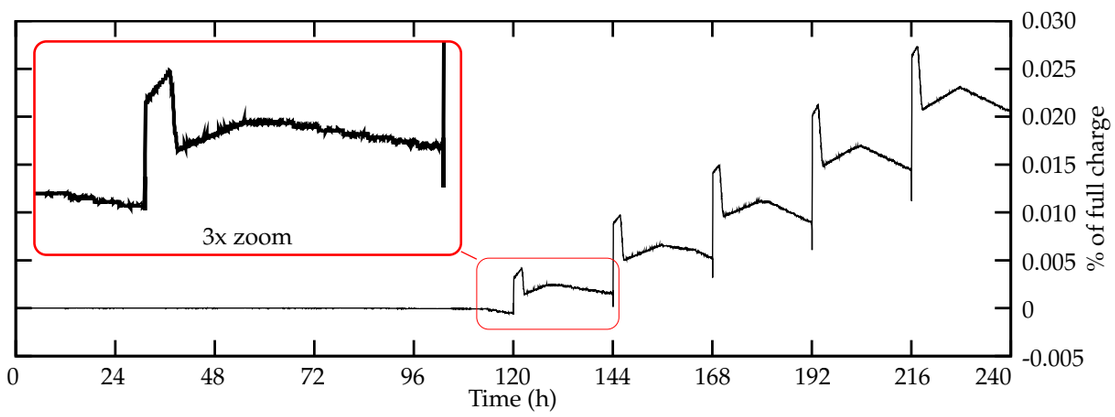
Increasing the rate of observations costs

The next simulations (S_2) have an increased rate of observations and thus an increased radio traffic in the network. The effect of the increased data rate is primarily that the network consumes more power. This extra power consumption speeds up the time from the start of the simulation until the network finds the alternate routing pattern compared to the S_1 simulations.

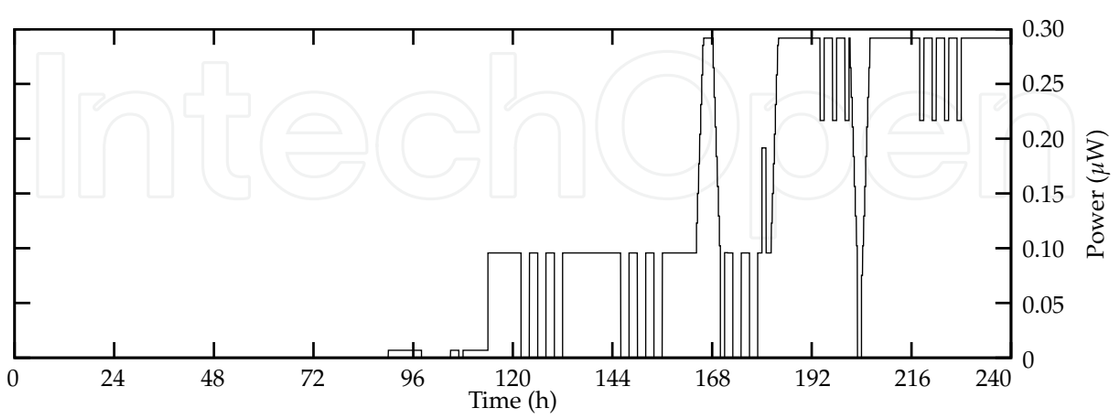
Fig. 12 shows that the minimum energy in any of the nodes in the network stabilises with the DEHAR algorithm. The level at which it stabilises is lower than in the S_1 simulations, which is expectable. The faster observation rate hurts the DD network and a node will already be drained from energy in about 10 days.



(a) Average energy in nodes for each simulation of S_1.



(b) Difference in the average energy in nodes for simulations in S_1. Given that the two curves in Fig. 11(a) are characterised by the functions $f_{\text{DEHAR}}(t)$ and $f_{\text{DD}}(t)$, then the curve in this figure is characterised by $f_{\text{DEHAR}}(t) - f_{\text{DD}}(t)$.



(c) Surplus energy consumption by DEHAR compared with DD for simulations in S_1.

Fig. 11. Results of simulations S_1 showing the first 10 days. The two blow-ups in (b) and (c) emphasises the first important difference between the DEHAR and DD algorithms.

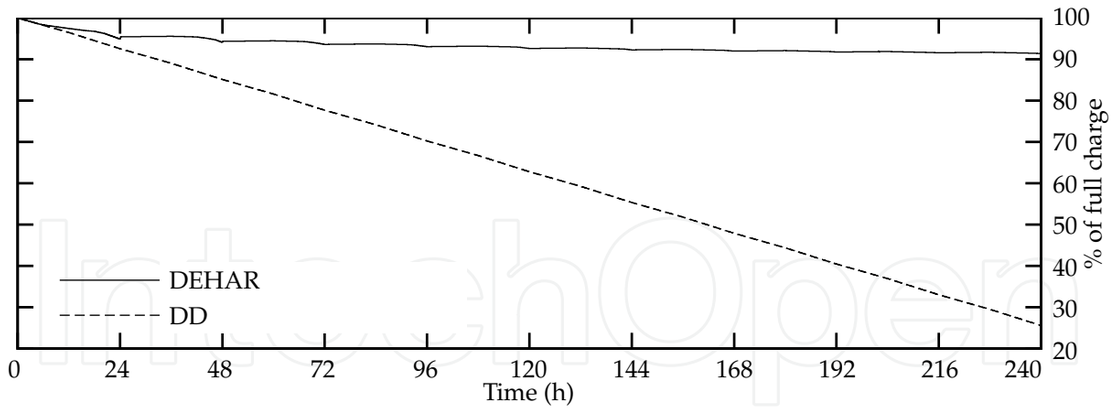


Fig. 12. Minimum energy in any node of the simulations in S_2. The day cycle is barely visible due to the compressed y-scale, compared to the simulations S_1.

The routing trend of the DEHAR algorithm is the same in the simulations S_1 and S_2. The only difference is that the DEHAR algorithm finds this alternative routing pattern faster in S_2 than in S_1.

The energy statistics of the node covered by the strongest shadow (at coordinate (1,3)) can be analysed. A graph of the energy level of this node will look similar to Fig. 12 and (in this simulation) it stabilises at precisely the same energy level. This show that the energy it can harvest closely matches the energy it needs to perform routing updates and performing observations (i.e. refraining from routing other nodes observations).

5. Conclusion

We have presented a new modelling framework aimed at describing and analysing wireless sensor networks with energy harvesting capabilities. The framework comprises of a conceptual basis and an operational basis, which were used to describe and explain two wireless sensor networks with energy harvesting capabilities. One of these network models is based on DD, i.e. it supports energy harvesting; but the routing is not energy aware, as it just forwards observations to the base station along statically defined shortest pathes. The other network model is based on the energy harvesting aware routing protocol DEHAR. Both of these networks were given natural explanations using the concepts of the modelling framework, and this gives a first weak validation of the adequacy of the framework. More experiments are, of course, needed for a thorough validation. Simulation results show that energy awareness of DEHAR-based networks can significantly extend the lifetime of nodes and it significantly improves the energy stored in the network, compared with a network like DD, with no energy aware routing.

There are several natural extension of this work.

First of all, the modelling framework should be validated by establishing its applicability in a broad collection of energy harvesting aware networks. The framework should be extended to include the deployment phase, where the nodes communicate in order to initialize their states. We do not expect principle difficulties in these extensions, but they are, of course, technical.

The generic framework may be instantiated in ways which will not be beneficial for the energy situation in the network. It is desirable and challenging to establish conditions which instantiations should satisfy in order to define an adequate energy harvesting aware network.

Another natural development would be to implement a platform for the modelling framework. The formalized parts of the framework provide good bases for such an implementation; but further formalization concerning the network communication and the medium should be considered prior to an implementation.

6. Acknowledgment

This research has partially been funded by the SYSMODEL project (ARTEMIS JU 100035) and by the IDEA4CPS project granted by the Danish Research Foundation for Basic Research.

7. References

Bush, L. A., Carothers, C. D. & Szymanski, B. K. (2005). Algorithm for Optimizing Energy Use and Path Resilience in Sensor Networks, *Wireless Sensor Networks, 2005. Proc. of the Second European Workshop on*, pp. 391 – 396.

Corke, P., Valencia, P., Sikka, P., Wark, T. & Overs, L. (2007). Long-duration solar-powered wireless sensor networks, *Proc. of the 4th workshop on Embedded networked sensors*, ACM, pp. 33–37.

Faruque, J. & Helmy, A. (2003). Gradient-based routing in sensor networks, *SIGMOBILE Mob. Comput. Commun. Rev.* 7(4): 50–52.

Hassanein, H. & Luo, J. (2006). Reliable Energy Aware Routing in Wireless Sensor Networks, *Dependability and Security in Sensor Networks and Systems, 2006*, IEEE, pp. 54–64.

Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J. & Silva, F. (2002). Directed Diffusion for Wireless Sensor Networking, *IEEE/ACM Transactions on Networking* 11(1): 2–16.

Islam, J., Islam, M. & Islam, N. (2007). A-sLEACH: An Advanced Solar Aware Leach Protocol for Energy Efficient Routing in Wireless Sensor Networks, *International Conference on Networking* 0: 4.

Jakobsen, M. K. (2008). *Energy harvesting aware routing and scheduling in wireless sensor networks*, Master's thesis, Technical University of Denmark, Department of Informatics and Mathematical Modeling.

Jakobsen, M. K., Madsen, J. & Hansen, M. R. (2010). DEHAR: A distributed energy harvesting aware routing algorithm for ad-hoc multi-hop wireless sensor networks, *World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a*, pp. 1 –9.

Jiang, X., Polastre, J. & Culler, D. (2005). Perpetual environmentally powered sensor networks, *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, IEEE Press, pp. 463 – 468.

Kansal, A., Hsu, J., Zahedi, S. & Srivastava, M. B. (2007). Power management in energy harvesting sensor networks, *ACM Trans. Embed. Comput. Syst.* 6(4): 32.

Kansal, A., Potter, D. & Srivastava, M. (2004). Performance Aware Tasking for Environmentally Powered Sensor Networks, *ACM Joint Intl. Conf. on Measurement and Modeling of Computer Systems*.

Lattanzi, E., Regini, E., Acquaviva, A. & Bogliolo, A. (2007). Energetic sustainability of routing algorithms for energy-harvesting wireless sensor networks, *Comput. Commun.* 30(14-15): 2976–2986.

Lin, L., Shroff, N. B. & Srikant, R. (2007). Asymptotically optimal energy-aware routing for multihop wireless networks with renewable energy sources, *IEEE/ACM Transactions on Networking* 15(5): 1021–1034.

Ma, C. & Yang, Y. (2006). Battery-aware routing for streaming data transmissions in wireless sensor networks, *Mob. Netw. Appl.* 11(5): 757–767.

Mann, R. P., Namuduri, K. R. & Pendse, R. (2005). Energy-Aware Routing Protocol for Ad Hoc Wireless Sensor Networks, *EURASIP Journal on Wireless Communications and Networking* 2005(5): 635–644.

Mørk, S., Godskesen, J., Hansen, M. R. & Sharp, R. (1996). A timed semantics for sdl, *in* R. Gotzhein & J. Bredereke (eds), *Formal Description Techniques IX: Theory, application and tools*, Chapman & Hall, pp. 295–309.

Moser, C., Thiele, L., Benini, L. & Brunelli, D. (2006). Real-Time Scheduling with Regenerative Energy, *Proc. of the 18th Euromicro Conf. on Real-Time Systems*, IEEE Computer Society, pp. 261–270.

Pilegaard, H., Hansen, M. R. & Sharp, R. (2003). An approach to analyzing availability properties of security protocols, *Nordic Journal of Computing* 10: 337–373.

S.D., M., D.C.F., M., R.I., B. & A.O., F. (2005). A centralized energy-efficient routing protocol for wireless sensor networks, *IEEE Communications Magazine* 43(3): S8–13.

Shah, R. C. & Rabaey, J. M. (2002). Energy aware routing for low energy ad hoc sensor networks, *Wireless Communications and Networking Conf., 2002*, Vol. 1, IEEE, pp. 350 – 355.

Simjee, F. & Chou, P. H. (2006). Everlast: long-life, supercapacitor-operated wireless sensor node, *Proc. of the 2006 intl. symposium on Low power electronics and design*, ACM Press, pp. 197–202.

Vergados, D. J., Pantazis, N. A. & Vergados, D. D. (2008). Energy-efficient route selection strategies for wireless sensor networks, *Mob. Netw. Appl.* 13(3-4): 285–296.

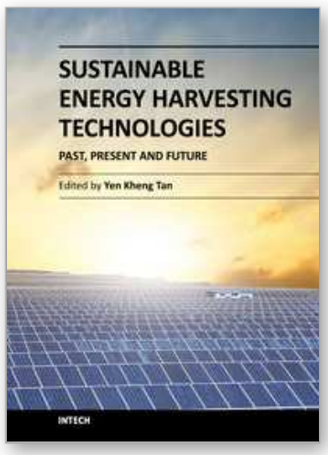
Voigt, T., Dunkels, A., Alonso, J., Ritter, H. & Schiller, J. (2004). Solar-aware clustering in wireless sensor networks, *IEEE Symp. on Computers and Communications* 1: 238–243.

Voigt, T., Ritter, H. & Schiller, J. (2003). Solar-aware Routing in Wireless Sensor Networks, *Intl. Workshop on Personal Wireless Communications*, Springer, pp. 847–852.

Xu, J., Peric, B. & Vojcic, B. (2006). Performance of energy-aware and link-adaptive routing metrics for ultra wideband sensor networks, *Mob. Netw. Appl.* 11(4): 509–519.

Zeng, K., Ren, K., Lou, W. & Moran, P. J. (2006). Energy-aware geographic routing in lossy wireless sensor networks with environmental energy supply, *Proc. of the 3rd intl. conf. on Quality of service in heterogeneous wired/wireless networks*, ACM Press, p. 8.

Zhang, B. & Mouftah, H. T. (2004). Adaptive Energy-Aware Routing Protocols for Wireless Ad Hoc Networks, *Proc of the First Intl. Conf. on Quality of Service in Heterogeneous Wired/Wireless Networks*, IEEE Computer Society, pp. 252–259.



Sustainable Energy Harvesting Technologies - Past, Present and Future

Edited by Dr. Yen Kheng Tan

ISBN 978-953-307-438-2
Hard cover, 256 pages
Publisher InTech
Published online 22, December, 2011
Published in print edition December, 2011

In the early 21st century, research and development of sustainable energy harvesting (EH) technologies have started. Since then, many EH technologies have evolved, advanced and even been successfully developed into hardware prototypes for sustaining the operational lifetime of low?power electronic devices like mobile gadgets, smart wireless sensor networks, etc. Energy harvesting is a technology that harvests freely available renewable energy from the ambient environment to recharge or put used energy back into the energy storage devices without the hassle of disrupting or even discontinuing the normal operation of the specific application. With the prior knowledge and experience developed over a decade ago, progress of sustainable EH technologies research is still intact and ongoing. EH technologies are starting to mature and strong synergies are formulating with dedicate application areas. To move forward, now would be a good time to setup a review and brainstorm session to evaluate the past, investigate and think through the present and understand and plan for the future sustainable energy harvesting technologies.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Michael R. Hansen, Mikkel Koefoed Jakobsen and Jan Madsen (2011). A Modelling Framework for Energy Harvesting Aware Wireless Sensor Networks, Sustainable Energy Harvesting Technologies - Past, Present and Future, Dr. Yen Kheng Tan (Ed.), ISBN: 978-953-307-438-2, InTech, Available from: http://www.intechopen.com/books/sustainable-energy-harvesting-technologies-past-present-and-future/a-modelling-framework-for-energy-harvesting-aware-wireless-sensor-networks

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen