Electronic Communications of the EASST
Volume 16 (2009)

Proceedings of the
Doctoral Symposium at the
International Conference on Graph Transformation
(ICGT 2008)

Modelling Clustering of Wireless Sensor Networks with Synchronised Hyperedge Replacement

Mohammad Hammoudeh, Robert Newman, and Sarah Mount

14 pages

Guest Editors: Andrea Corradini, Emilio Tuosto
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Modelling Clustering of Wireless Sensor Networks with Synchronised Hyperedge Replacement

Mohammad Hammoudeh, Robert Newman, and Sarah Mount

{m.h.h, r.newman, s.moutn}@wlv.ac.uk, http://www.wlv.ac.uk/
School of Computing and IT
University of Wolverhampton, Wolverhampton, UK

Abstract: This paper proposes Synchronised Hyperedge Replacement (SHR) as a suitable modelling framework for Wireless Sensor Networks (WSNs). SHR facilitates explicit modelling of WSNs applications environmental conditions (that significantly affect applications performance) while providing a sufficiently high level of abstraction for the specification of the underling coordination mechanisms. Because it is an intractable problem to solve in distributed manner, and distribution is important, we propose a new Nutrient-flow-based Distributed Clustering (NDC) algorithm to be used as a working example. The key contribution of this work is to demonstrate that SHR is sufficiently expressive to describe WSNs algorithms and their behaviour at a suitable level of abstraction to allow onward analysis.

Keywords: Synchronised Hyperedge Replacement, Wireless Sensor Networks, Clustering

1 Introduction

A successfully large-scale Wireless Sensor Network (WSN) deployment necessitates that the design concepts are analysed before they are mapped onto specific hardware platforms. Developing, testing, and evaluating network protocols and supporting architectures and services for WSNs are typically undertaken through test-beds or simulation. The substantial cost of deploying and maintaining large-scale WSNs and the time needed for setting up the network for experimental goals makes simulation invaluable in developing reliable and portable applications for WSNs. For instance, test-beds implementation is not always viable because it does not easily scale to large number of nodes. Nonetheless, simulation in WSNs has its own set of problems. Some simulators are acknowledged to be difficult to use [Cur] (e.g., NS-2 [NS-07]), platform dependent (e.g., TOSSIM [LLWC03]), containing inherent bugs (e.g., NS-2), distributed under commercial licenses (e.g., OPNET [Inc07]), etc.

We contend that abstract models can provide cost effective methods for analysing WSNs algorithms before deployment. It can, for example, help in assessing the scalability of algorithms independently from target hardware platforms and with a certain degree of accuracy; or else, it may simplify the software development process for WSNs applications. One of the main difficulties in modelling WSNs lays in their intrinsic interdependency with the environment in which they operate. It had been observed that WSNs applications are characterised by the tasks involved in a particular application. WSNs applications attribute integration of communication, computation, and interaction with the physical environment.

This inherent environment-application mutuality suggests that a reasonable model for WSNs has to encompass the environmental conditions affecting the performance of WSNs systems. To the best of our knowledge, a suitable abstract model encompassing those peculiarities of WSNs is still missing. Available models, like those defined in [GBT^{+}04, GEOW04, CG04] neglect the environmental issues. We contend that a modelling framework for WSNs to cover this concerns is required. In [Tuo05], a first attempt to specify such a model has been tried showing how Synchronised Hyperedge Replacement (SHR) can suitably represent several aspects of WSNs mechanisms in a unique formal framework.

This paper advocates the use of graph transformations as a suitable modelling framework for WSNs. Specifically, we suggest using SHR to formally specify various coordination aspects of WSNs applications and applications environments at different abstraction levels. We show how SHR can explicitly model very low level aspects, such as wireless communications, along with abstract ones, such as sensors spatial distribution. We also propose a new WSNs clustering algorithm called Nutrient-flow-based Distributed Clustering (NDC) which generates balanced clusters. NDC is used as a working example because it is a challenging problem to solve in a distributed manner. SHR separates the coordination aspects of algorithms, such as NDC, from the data-related aspects. In fact, the metrics used to balance the clusters are parametric with respect to the SHR productions describing sensors coordination. This makes our approach suitable for studying a wide range of algorithms classes obtained by varying the metric criteria.

2 Related Work

In WSNs, clustering is the process of grouping nodes into disjoint and homogeneous groups called clusters. Low Energy Adaptive Clustering Hierarchy (LEACH) [HCB02] is one of the most promising routing algorithms for WSNs. However, LEACH has been based on a number of unrealistic assumptions which in the authors' opinion limit its effectiveness in a number of applications. These assumptions are listed in [HKG07]. MuMHR [HKG07] is an improvement over LEACH. MuMHR provides solutions to some of the limitations of LEACH. Similar to LEACH, MuMHR does not generate balanced clusters. The balanced clustering algorithm proposed in [GSYS02] studies the theoretical aspects of the clustering problem in WSNs with application to energy optimisation. Since the linear separability principle does not hold in balanced problems, this algorithm is not suitable for the optimal balanced clustering. In the balanced clustering algorithm, every node should have knowledge about all other nodes in the cluster. Moreover, this algorithm does not consider the transmission range of nodes. Finally, using their defined diameter metric instead of the distance between the nodes and their cluster heads could result in energy inefficient clustering.

A variety of process algebras to model wireless communications have been recently proposed such as [MS09, God08, God07, LT05]. In the authors' opinion, there is no one model that is universally suitable. Practical choice of a modelling approach depends on the application as well as on the availability of tools for developing, editing, and debugging rewriting approaches. Important features include the readability and logical manageability of rewrite rules, formal properties, and effectiveness of rule application.

The limited availability of development tools limit the practical use of those approaches. In the future, it will become clearer which model is most suitable as the creation new tools and improving the available tools will allow researchers to understand and collect more experience with executable modelling systems.

3 Nutrient-flow-based Distributed Clustering

The aims of the *N*utrient-flow-based *D*istributed *C*lustering (NDC) algorithm are:

(1) To equalise, so far as is possible, the diameter of the clusters.

(2) To equalise, so far as is possible, the membership of the clusters.

The distributed model described here is based around a metaphor of nutrient flow supporting some life-form, such as a mould. The concept is to provide a limited supply of nutrient, and allow the nodes to ally themselves with a cluster head which will provide the largest nutrient supply. If properly regulated, this should lead to clusters broadly equalising their membership. In order to minimise the radius of a cluster, it is arranged that some of the supply of nutrient is lost in transit between nodes - the further the distance travelled, the more is lost. It is important that some advantage be given to nodes the join in a cluster, rather than communicating directly with the sink. For this reason, it is necessary to provide some advantage associated with clustering, as opposed to direct communication. The simplest way to do this is to make the loss of nutrient super-proportional to the distance of a link. Given that in real life, radio propagation obeys an inverse square law, it seems reasonable to make the loss of nutrient proportional to the square of the distance travelled.

Like many distributed route discovery algorithms, this one operates in distinct phases or epochs, with the network reconfiguring itself from epoch to epoch. During each epoch, nodes try to improve the amount of nutrient available to them. They do this by contacting a local node at random and, if clustering with that node will offer a better supply than that which is currently available, then the node changes allegiance. Nodes receiving requests for nutrient from other nodes make an offer back to that node, giving an estimate of the nutrient that would have been available had that node been a member of its cluster. The estimate depends both on the amount of nutrient available, and the number of nodes dependent on that cluster head.

Another consideration is that the algorithm has to give encouragement for clusters to grow, that is that the amount of nutrient available becomes greater as nodes join the network. To effect this, in each epoch the sink has available an amount of nutrient proportional to the total number of connected nodes. The starting conditions are as follows:

(1) Some initial store of nutrient available at the sink, n_{sink}

(2) Current state of all other nodes is to have no nutrient, $n_{av} = 0$

From the initial state, some nodes (N_c) will by chance have direct contact with the sink. These become the initial cluster heads, and each is given an equal share of the nutrient available $n_{av} = \frac{n_{sink}}{|N_c|}$, which is available to them as attenuated by the square of the distance from the sink. Across the network the sequence of events in each epoch is as follows:

i) Each node transmits to its dependents (if any) the total amount of nutrient available to that cluster and the current number of members (including the cluster head) at that level of the hierarchy. Each dependent calculates its share of nutrient, S, for this epoch, which is

$$S = \frac{n}{m \times k \times d^2}$$

Algorithm 1 NDC clustering algorithm.

```
1.  nodes in $N_c$ contact the sink
2.  nodes in $N_c$ become a cluster heads
3.  the sink gives each cluster head nutrient share $n_{av}$
4.  each cluster head send its dependents the value of $n$ and $m$
5.      each dependent node does the following:
6.          calculate $S$
7.          forwarded current cluster head id and received $n$ and $m$ values
8.  nodes receiving the forwarded message do the following:
9.      calculate $S'$
10.     IF $S' > S$ THEN
11.         leave current cluster head
12.         join the cluster head in the forwarded message
13.     END IF
14. all cluster head send there $m$ value to the sink
15. the sink calculate $n_{next}$
16. the sink broadcast $n_{next}$ value
```

where n is the total nutrient available to the cluster, m is the number of members, k is a constant of proportionality for the distance adjustment and d is the distance of the node from the cluster head.
ii) Each node which has a supply of nutrient selects another node (or set of nodes, to speed up the evolution of the system) at random and forwards the above information, along with the identity of the cluster head.
iii) The receiving node calculates the amount of nutrient, S', it could have received in this epoch as a member of that cluster. If the amount is greater than its actual allocation in this epoch it communicates with the cluster head and joins the cluster (also communicates with its old cluster head to leave that cluster).
iv) Cluster heads propagate upwards through the network the number of members. The sink calculates the amount of nutrient available for the next epoch using the formula

$$n_{next} = \frac{n_o \times m_n}{m_o}$$

where n_{next} is the nutrient available for the next epoch, n_o is the nutrient available this epoch, m_n is the number of members reported, m_o is the number of members reported for the previous epoch.

The operation of the NDC clustering is summarised in algorithm 1.

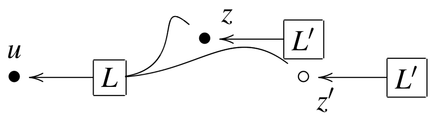
4 SHR at a Glance

We briefly review the main concepts of SHR. For an in-depth treatment of SHR and related literature the reader is referred to [Hir03, Tuo03, FHL+05] and references therein. In the following, the set of tuples on a set X is denoted as X^*; a tuple of length n is written as $\langle x_1, \ldots, x_n \rangle$ ($\langle\rangle$ is the empty tuple); dom(f) is the domain of the function f and $f|_S$ is the restriction of f to S, namely $f\lfloor S(x) = f(x)$ if $x \in S$, $f\lfloor S(x)$ is undefined otherwise. A (hyper)graph consists of nodes, elements of a countably infinite set $\mathscr{N}^*$, and *(hyper)edges* connecting nodes and labelled with elements of a set of edge labels $\mathscr{L}$. If $L \in \mathscr{L}$, $\text{rank}(L)$ is a natural number, called the *rank* of L, that expresses L's *attachment node* (namely the nodes L insists on). A *syntactic judgement* specifies a graph.

Definition 1 (Graphs as judgements). A *judgement* has form $\Gamma \vdash G$ where (i) $\Gamma \subseteq \mathscr{N}^*$ is a finite set of nodes (the *free nodes* of the graph), and (ii) G is a *graph term* generated by the grammar $G ::= L(x_1,\dots,x_n)\ G \mid G \mid \nu y\, G \mid nil$ where $y, x_1, \dots x_n \in \mathscr{N}^*$, $L \in \mathscr{L}$, and $\text{rank}(L) = n$. Also, the restriction operator ν is a binder; $\text{fn}G$ is defined as usual and we demand that $\text{fn}G \subseteq \Gamma$.

Edges are terms of the form $L(x_1,\dots,x_n)$, $\mid$ is the parallel composition operator of graphs, νy is the restriction operator (with priority lower than $\mid$) of nodes and *nil* is the empty graph. Condition $\text{fn}G \subseteq \Gamma$ accounts for having free isolated nodes in G (e.g., $x \vdash nill$ is graph with only the isolated node x). Judgements are compactly written by dropping curly brackets from interfaces and writing Γ_1, Γ_2 for $\Gamma_1 \cup \Gamma_2$ whenever $\Gamma_1 \cap \Gamma_2 = \emptyset$ (e.g., $\Gamma, x \vdash G = \Gamma \cup \{x\} \vdash G$, if $x \notin \Gamma$).

Example 1. The judgement $u, z \vdash \nu z'\ L(u,z,z') \mid L'(z) \mid L'(z')$ represents a graph with three edges labelled by L (of rank 3) and L' (of rank 1); $u, z \in \mathscr{N}$ are free and z' is bound. The graph can be depicted as



where edges are drawn as labelled boxes and nodes are bullets (empty for bound nodes and solid for free nodes). A connection between a node and an edge is called *tentacle*. An arrowed tentacle indicates the first attachment node of the edge; the other nodes are determined by numbering tentacles clockwise (e.g., for L, u is the first attachment node, z the second and z' the third).

Graph terms are considered up to *structural congruence* (not reported here; see [FHL⁺05]) that yields the usual rules for restriction and a monoidal structure with respect to parallel composition and *nil* as neutral element.

The SHR version used here adopts the so called "Milner" synchronisation algebra with mobility [Lan06] that relies on the notion of *action signature*.

Fix a set *Act* of *input actions* with two distinguished actions τ and ε (used below to express synchronisation and idleness, respectively), and let $Act_{Mil} = Act \cup \overline{Act} \cup \{\tau, \varepsilon\}$ where $\overline{Act} = \{\overline{a} : a \in Act\}$ is the set of *output action*. An *action signature* is a triple $(Act_{Mil}, ar, \varepsilon)$ where $ar : Act_{Mil} \to \omega$ is an *arity function* such that $ar(\varepsilon) = ar(\tau) = 0$ and $\forall a \in Act : ar(a) = ar(\overline{a})$).

Mobility is modelled through a *synchronisation function* $\Lambda : \Gamma \to (Act_{Mil} \times \mathscr{N}^*)$ assigning, to each $x \in \Gamma$, an action $a \in Act_{Mil}$ and a tuple $\langle x_{1,}, \dots, x_n\rangle$ of nodes sent to x such that $ar(a) = n$. We let $Act\Lambda x = a$ and $n\Lambda(x) = \{x_{1,}, \dots, x_n\}$ when $\Lambda(x) = (a, \langle x_{1,}, \dots, x_n\rangle)$. Finally, the *set of communicated (resp. fresh) names* of Λ is $n(\Lambda) = \{z : \exists x. z \in n_\Lambda(x)\}$ (resp. $\Gamma_\Lambda = n(\Lambda) \setminus \Gamma$).

Definition 2 (SHR transitions with mobility). Given an action signature Act_{Mil}, a *SHR transition* is a relation of the form:

$$\Gamma \vdash G \xrightarrow{\Lambda,\pi} \Phi \vdash G'$$

where $\pi : \Gamma \to \Gamma$ is an idempotent substitution[1] *s.t.* $\forall x \in n(\Lambda) \cap \Gamma .\ x\pi = x$ and $\Phi = \Gamma\pi \cup \Gamma_\Lambda$. Substitution π induces equivalence classes on Γ (i.e., $[x]_\pi = \{y \in \Gamma : x\pi = y\pi\}$, for $x \in \Gamma$) and selects their representative element (idempotency). Only representatives can be communicated by Λ, while, by definition of Φ, free nodes are never erased and new nodes are bound unless communicated.

Productions specify how edges should be rewritten:

Definition 3 (Productions). Let $L \in \mathscr{L}$ have rank n and G be a graph term. A *production* is a transition of the form:

$$x_1,\dots,x_n \vdash L(x_1,\dots,x_n) \xrightarrow{\Lambda,\pi} \Phi \vdash G$$

where $x_1,\dots,x_n$ are all distinct. This production is idle if $\Lambda(x_i) = (\varepsilon, \langle\rangle)$ for each i, $\pi = id$ and $\Phi \vdash G = x_1,\dots,x_n \vdash L(x_1,\dots,x_n)$.

For each $L \in \mathscr{L}$, the set of productions of L are assumed to contain the idle production for L and to be closed under injective re-namings.

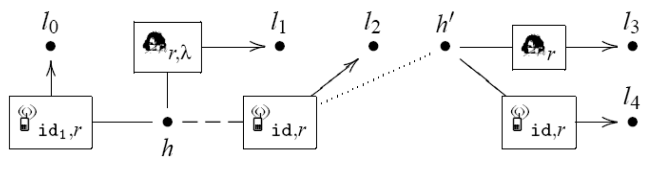
The operational semantics of SHR is omitted for brevity, here we just give an example of how SHR productions are synchronised.

Example 2. If ${}_r$ and ${}_{id,r}$ are labels for cluster heads and sensors respectively, then (ignoring the communication between the sink and the cluster head) the productions for moving a node from a cluster to another at the beginning of a new clustering epoch may be given as

$$l,h \vdash {}_r(l,h) \xrightarrow{h\ \overline{\mathbf{mv}_{\mathtt{id}}}\langle h'\rangle} l,h,h' \vdash {}_r(l,h)$$
$$l,h \vdash {}_{\mathtt{id},r}(l,h) \xrightarrow{h\ \mathbf{mv}_{\mathtt{id}}\langle h'\rangle} l,h,h' \vdash {}_{\mathtt{id},r}(l,h')$$

Namely, the cluster head orders the member `id` to migrate to h' therefore, upon synchronisation, the sensor `id` moves its membership tentacle which models the migration to the cluster headed by another cluster head (the one represented by an edge ${}_r(\_,h')$).

Let G be the graph obtained by removing the dotted tentacle from synchronising the previous productions in G, which yields the graph obtained by replacing the dashed with the dotted tentacle in G.



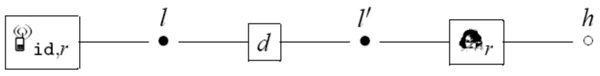
The transition described in Example 2 is obtained by (i) finding in the graph edges labelled by the lhs of each production we want to synchronise; (ii) remove those edges and (iii) replace them with the corresponding rhd of their productions; (iv) collapsing those node that are communicated.

[1] As customary, substitution application is expressed by juxtaposition of the substitution function (e.g., $x\pi$ stands for $\pi(x)$).

5 NDC Modelling in SHR

Following [Tuo05], we give some basic SHR productions aiming to model the environmental conditions of a WSN. We simplify the representation by considering only the geographical information necessary to the coordination among sensors and a sink. Remarkably, extending the present model with further environmental conditions (e.g., obstacles forbidding or deteriorating wireless signals) requires just a few new productions amenable to be synchronised with those presented here. We shall use the labels $☻_r$ and $📱_{id,r}$ to respectively denote cluster head and sensor edges with transmission range r; the different states edges can reach (possibly changing their rank) shall be represented by $☻'_r$ and $📱'_{id,r}$, etc. Geographic information can be modelled in SHR as a group of nodes representing geographical points (ranged over by l, l', etc.) connected by edges representing the Euclidean distance between points.

Example 3. A network where a sensor and a cluster head are respectively located at l and l' separated by a distance d is described by the graph $l,l' \vdash 📱_{id,r}(l) \mid d(l,l') \mid ☻_r(l',h)$ depicted as



In the above example, nodes l and l' are physical "public" locations while node h is the *membership node*, namely the node where sensors joining the cluster headed by the attached cluster head edge will be connected. Though there is no formal difference between location and membership nodes, we want to emphasise their different conceptual distinction. In order to show the suitability of SHR as modelling framework for WSNs, communications at two different levels of abstraction are considered. The lower level shows how wireless communications can be described; in fact, the environmental edges are the carrier of the signal emitted by the sensors or the sink. The higher level instead abstracts from the communication details and focuses on the logical interactions that sensors, cluster heads and sink must perform as prescribed by NDC. The distinction will be clear later, but considering the graph in the Example 3, this can be explained as follows: on location nodes (l and l'), communications are mediated by the environmental edges while, on the membership nodes, they are assumed to be abstract communications.

5.1 Forming Clusters

Once selected, cluster heads signal their presence with the production

$$l,h \vdash ☻_r(l,h) \xrightarrow{l\ \overline{☞_r}\langle h\rangle} l,h \vdash ☻_{r,[]}{}'(l,h)$$

Namely, a cluster head emits a signal $☞_r$ carrying its transmission range r (explicitly represented in the label) and relevant physical signal information $☞_r$ (e.g., direction, signal strength, frequency,... not explicitly represented). The cluster head also communicates the node h to which perspective members have to connect to; afterwards, they move to a state where they wait for instructions from the sink or requests for membership from other sensors. In the state $☻_{r,\lambda'}$, the cluster head records the list λ of the members of its cluster that is initialised to the empty list $[]$ in the previous production. Environmental edges propagate (or stop) signals emitted by sensors. This can be modelled by the following productions

$$l,l' \vdash d(l,l') \xrightarrow{l\ \text{☞}_x\langle h\rangle,\ l'\ \overline{\text{☞}'_{\delta(x,d,\text{☞})}}\langle h\rangle} l,l',h \vdash d(l,l') \tag{2}$$

$$l,l' \vdash d(l,l') \xrightarrow{l'\ \text{☞}_x\langle h\rangle,\ l\ \overline{\text{☞}'_{\delta(x,d,\text{☞})}}\langle h\rangle} l,l',h \vdash d(l,l') \tag{3}$$

In equation 2, when d detects the presence of a cluster head at distance x from l, it echoes on l' the presence of the cluster head updated with the its distance from l' computed by the (monotonically increasing) function δ which takes x, d and the physical attributes of the signal in input. Notice that the forwarded information of the physical attributes can change depending on the geographical positions l and l'. Production (3) symmetrically propagates the signal from l' to l and, as production (2), it is supposed to be applied only if the residual strength of the signal (carried by ☞_x) is enough for the signal to cover d; on the contrary, the productions

$$l,l' \vdash d(l,l') \xrightarrow{l\ \text{☞}_x\langle h\rangle} l,l',h \vdash d(l,l') \quad \text{and} \quad l,l' \vdash d(l,l') \xrightarrow{l'\ \text{☞}_x\langle h\rangle} l,l',h \vdash d(l,l')$$

will stop the signal. Noticeably, productions (2) and (3) are parametric with respect to the used metric (hidden in δ). This shows how orthogonal aspects can be naturally kept separated in SHR so that the model actually specify a class of algorithms (those obtained by instantiating different metrics) and can therefore be used to study and compare them. Eventually, the presence of a cluster head can be detected by a sensor so that the latter can apply for membership to the former according to the production

$$l \vdash \text{📱}_{\mathtt{id},r}(l) \xrightarrow{l\ \text{☞}_x\langle h\rangle} l,h \vdash \text{📱}_{\mathtt{id},r,x}{}'(l,h)$$

stating that a sensor in state $\text{📱}'_{id,r,x}$ records its distance x from the cluster head and connects to the membership node h over which a membership request can be issued:

$$l,h \vdash \text{📱}_{\mathtt{id},r,x}{}'(l,h) \xrightarrow{h\ \overline{\text{♪}}\langle l\rangle} l,h \vdash \text{📱}_{\mathtt{id},r,x}{}''(l,h)$$

notice that in the request ♪ the sensor includes its position l. And by

$$l,h \vdash \text{👤}_{r,\lambda}{}'(l,h) \xrightarrow{h\ \text{♪}_{\mathtt{id}}\langle l'\rangle} l,l',h \vdash \text{👤}_{r,\lambda@(\mathtt{id},l')}{}'(l,h)$$

the cluster head adds the new member to its list on accepting the new member. A sensor can also decide to ignore a signal:

$$l,h \vdash \text{📱}_{\mathtt{id},r,x}{}'(l,h) \xrightarrow{h\ \overline{\text{♪}}\langle l\rangle} l,h \vdash \text{📱}_{\mathtt{id},r,x}{}''(l,h)$$

During the cluster formation phase, further cluster heads can be detected by sensors. Indeed, it may even happen that a device senses the presence of other cluster head after having already joined a cluster. Hence, in order to optimise its energy a sensor can decide to leave its current cluster and join a geographically closer one. This can be obtained by the following two productions:

$$l,h \vdash \text{📡}''_{\mathtt{id},r,x}(l,h) \xrightarrow{l\ ☞_y\langle h'\rangle} l,h,h' \vdash \text{📡}''_{\mathtt{id},r,x}(l,h), \qquad y \geq x$$

$$l,h \vdash \text{📡}''_{\mathtt{id},r,x}(l,h) \xrightarrow{l\ ☞_y\langle h'\rangle,\ h\ \overline{\mathbf{bye}}_{\mathtt{id}}\langle l\rangle} l,h,h' \vdash \text{📡}'_{\mathtt{id},r,x}(l,h'), \qquad y < x$$

Accordingly, cluster heads acknowledge migration with the production

$$l,h \vdash ♚'_{r,\lambda}(l,h) \xrightarrow{h\ \mathbf{bye}_{\mathtt{id}}\langle l'\rangle} l,l',h \vdash ♚'_{r,\lambda\setminus\mathtt{id}}(l,h)$$

notice that the list of members is updated.

5.2 Balancing Clusters

At this stage, cluster heads send their list of members to the sink:

$$l,h \vdash \text{📡}''_{\mathtt{id},r,x}(l,h) \xrightarrow{h\ \mathbf{mv}_{\mathtt{id}}\langle h'\rangle} l,h,h' \vdash \text{📡}'_{\mathtt{id},r,x}(l,h')$$

where signal $\mathbf{ls}_\lambda$ carries position and membership node and is propagated through the environmental nodes (like ☞ signals).

We now refine the productions in Example 3. On reception of a message from the sink requiring a sensor to migrate to another cluster, a cluster head orders the device to move:

$$l,h \vdash ♚'_{r,\lambda}(l,h) \xrightarrow{l\ \mathbf{mv}_{\mathtt{id}}\langle h'\rangle,\ h\ \overline{\mathbf{mv}_{\mathtt{id}}}\langle h'\rangle} l,h,h' \vdash ♚'_{r,\lambda\setminus\mathtt{id}}(l,h)$$

and, correspondingly, the sensor migrates to the new cluster:

$$l,h \vdash ♚'_{r,\lambda}(l,h) \xrightarrow{h\ \overline{\mathbf{ls}_\lambda}\langle l,h\rangle} l,l',h \vdash ♚'_{r,\lambda}(l,h)$$

Observe that the sensor returns to the state where it has to apply for membership to its (new) cluster head. Finally, sinks are labelled by $\circledS_\lambda$ where λ is the list of nodes and their locations as transmitted by cluster heads; we assume that $nodes(\lambda)$ returns the set of nodes occurring in the list λ and $\lambda \uplus \lambda'$ is the list obtained by appending λ' to λ. Productions for sinks are:

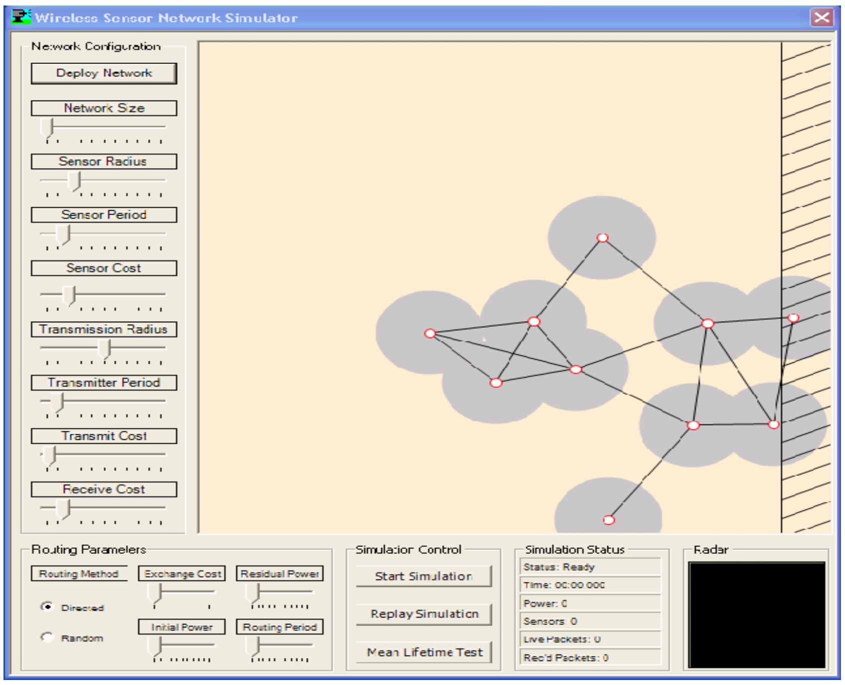
$$l \vdash \circledS_\lambda(l) \xrightarrow{l\ \mathbf{ls}_{\lambda'}\langle l\rangle} l, nodes(\lambda') \vdash \circledS_{\lambda\uplus\lambda'}(l)$$

$$l \vdash \circledS_\lambda(l) \xrightarrow{l\ \overline{\mathbf{mv}_{\mathtt{id}}\langle h\rangle}} l,h \vdash \circledS_{\lambda'}(l)$$

where λ' in the last production takes into account that sensor `id` will migrates to the cluster with membership node h. We remark that the metric criteria determining the choice of the migrating sensors are orthogonal to the coordination productions of the sink. Hence, the SHR model offers another dimension along which NDC can be parateterised.

5.3 A Practical Validation of the Model

In the identification of network properties, the issue of load balancing, in terms of communication, is of paramount importance. The discovery of the most utilised nodes in a large complex



Routing Method: [✓ Directed] Random

Status: Ready
Time: 00:00 000
Power: 0
Sensors: 0
Live Packets: 0
Rec'd Packets: 0

Figure 1: A 10 nodes random topology network.

network is vital to deal with communication bottlenecks and energy depletion problems. In this example we show how SHR can facilitate the ranking of nodes by quantifying their *degree* of utilisation, i.e. the degree to which they have direct relationships with other nodes in a graph. In particular, we define a measure of load, L, on nodes in a network to identify structurally important nodes. A node is said to be important if it has a relatively high rank.

Each node is associated to a sensor. An edge is defined between two nodes when the corresponding sensors can communicate with each other. Individual communication links are associated with energy communication cost. Assume that the size of all message types is constant; the exact values of these communication costs can be estimated according to the distance between the sensors. The amount of energy needed to transmit a message to a destination at distance d from the source can be calculated by the following formula: $e_s = kd^c$ where k and c are constants for a specific wireless system. While the amount of energy needed to receive a message is virtually constant (e_r). Formally, for a graph G, the measure of load of a node $z \in V(G)$ is given by $L(z) = (N_I \times e_s) + (N_O \times e_r)$ where N_I is the number of incoming tentacles and N_O is the number of outgoing tentacles connected with a node.

The usefulness of L was checked in simulation. A random topology network, (Figure 1), that consists of 10 nodes connected by 16 outgoing tentacles was deployed using the general WSN Simulator [Ste05]. All nodes has the same initial amount of energy. When a node detect a moving object it send this information to all nodes within its transmission range. Using L, we predict that nodes have higher number of outgoing tentacles will be the first to die. This prediction was confirmed by the simulation where nodes with $L = 5 \times e_s$ were observed to die first $(e_s > e_r)$.

The number of tentacles also it reflects the congestion level at various parts of the network.

5.4 An Example Verification of a NDC Property

This subsection provides an example around which an analytical proof may be attempted. We theoretically verify that by performing the action *mv*, a node can join exactly only one cluster.

In a typical WSN deployment, faulty behaviour can result in a node being a member of two or more clusters. For instance, a join cluster request message can be received by many cluster heads and only the one addressed in that message should respond by sending a notification message that carries the amount of nutrient. The cluster head *id* can be specified in action *m* of a signal edge as defined in the productions of signals in [Tuo05]. If the notification message is lost, the sending node waits for a short period of time and joins the second 'best' cluster head. In this case, the node might be registered as a member in both clusters. Another example is when a node leaves a cluster; it sends a message to the cluster head to be removed from its membership list. If that message is not received by the corresponding cluster head, the node might be registered as a member of the two clusters (previous and current).

Let 📡$_{id,r}(l,h,\Upsilon)$ be a sensor node in a network that consists of $k+1$ clusters. A node can only participate in the network *iff* it is a member of only one cluster at any time. 📡$_{id,r}$ is endowed with a set of attributes, Υ, containing the current clustering information, the amount of available energy amongst others. After a node joins a new cluster or renews its membership with the current cluster, it updates the information stored in Υ. The new *mv* action is described with the transition 📡$_{id,r}(l,h_a,\Upsilon) \xrightarrow{mv_{id}\langle h_b\rangle}$ 📡$_{id,r}(l,h_b,\Upsilon')$. *mv* has the following algebraic properties: non-commutative, transitive, and asymmetric.

Proposition 5.4.1

If 📡$_{id,r}(l,h_a,\Upsilon) \xrightarrow{h_a mv_{id}\langle h_b\rangle}$ 📡$_{id,r}(l,h_c,\Upsilon')$ *where* $a,b,c \in [1,k+1]$ *, then* $h_b == h_c$.

Proof The base case is when the *mv* transition is derived by the application of the rule $\xrightarrow{hmv_{id}\langle h_i\rangle}$ in a network that consists of only one cluster $(k=1)$. In a single cluster network, h_a is said to be identical to h_b and h_c. In this case, 📡$_{id,r}$ is said to renew its membership with the current cluster it operates within.

If 📡$_{id,r}(l,h_a,\Upsilon) \xrightarrow{h_a mv_{id}\langle h_b\rangle}$ 📡$_{id,r}(l,H,\Upsilon')$, then $H = h_b | h_c$. This transition can be derived by applying the rule *mv* where

📡$_{id,r}(l,h_a,\Upsilon) \xrightarrow{h_a mv_{id}\langle h_b\rangle}$ 📡$_{id,r}(l,h_b,\Upsilon')$ or 📡$_{id,r}(l,h_a,\Upsilon) \xrightarrow{h_a mv_{id}\langle h_b\rangle}$ 📡$_{id,r}(l,h_c,\Upsilon')$

The second transition holds only when $h_b = h_c$ because the requirements of the *mv* transition are:

1. h_a must synchronise with h_b using $\xrightarrow{lmv_{id}\langle h_b\rangle h_a \overline{mv_{id}}\langle h_b\rangle}$ (see Example 3).

2. The sensor node synchronises the join request, h♪$\langle l\rangle$, with the new cluster head add member h♪$\langle l'\rangle$ transition.

According to the productions for the environment defined in [Tuo05], an environment edge must coordinate the device action *mv* and ask its neighbour environment in the direction it wants to move toward for the next node. The node will leave h_a as this transition can be applied only if the actions on the external nodes synchronise with actions imposed by the productions of adjacent edges according to the adopted synchronisation policy in [Tuo05]. This guarantees that the node will be removed from the membership list of the previous cluster h_a. In NDC, the membership information (l,h) is kept at the sensor node. Observe that after the synchronisation of the *mv* transition, the node returns to the initial state where it has to apply for membership to a new cluster head. Also notice that the join request ♪ contains the sensor position l. When a notification message or remove node from cluster membership list message are lost, the cluster head removes, *rm*, the node from its membership list if it does not hear from the nodes by the following transition

$$l,h \vdash \text{☻}_r(l,h) \xrightarrow{l\overline{rm}\langle h\rangle} l,h \vdash \text{☻}_{r,[]}{}'(l,h)$$

6 Acknowledgements

Many thanks to Dr Emilio Tuosto for interesting discussions that helped improve this work. Also we thank the anonymous reviewers for their valuable comments and suggestions.

7 Final Remarks

This paper has presented an approach to modelling one of the major practical problems facing designers of large WSNs. Network configuration remains one of the most problematic tasks in WSN design. The largest practical networks so far have been "designed" - that is, the hierarchy has been fixed and built in at design time. This approach has resulted in an inevitable vulnerability to poor surveying of the real environment of the network site and to subsequent equipment failures, which render the designed-in architecture non viable. For this reason, self configuration remains a vitally important area of research, and algorithms such as NDC are of great interest in finding a practical solution.

However, it is very difficult to predict the behaviour of such decentralised protocols when scaled to a very large network. It is not unusual for unexpected behaviours to arise, and protocols must be designed to be resistant to these. As has been argued, the traditional approach of simulation can be limited, both due to the difficulty of building a simulation using current tools, and due to the computational resources needed to analyse a network of substantial size. Also, a simulation is not an intellectually satisfying "proof" of good behaviour. The success of one simulation cannot be taken as indicative of satisfactory function in situations other than the one in which the simulation was performed. Analytical examination of the network offers the possibility of proving at least some of its behaviours, and may therefore be able to reach where simulation can not. One major problem with this approach is finding a suitable modelling framework in which to describe the behaviour of the network. It is argued here that SHR is suitable and appropriate for this type of problem, and this has been demonstrated by the modelling of the NDC clustering algorithm using it.

SHR intuitive, visual, and mathematical precise modelling made it easy to specify and prototype the NDC algorithm by abstracting its intensive computation and clearly presenting communication and relations between different network objects. This made it simple to implement and simulate the algorithm with less errors.

Cluster balancing was chosen as a working example because it is an intractable problem to solve in a distributed manner. In proposing NDC, several tentative claims have been made for it. Not all of these claims can be verified using simulation, in particular, the suitability for an arbitrary number of cluster heads and the open ended scalability may only be proved analytically.

This paper provides a model around which an analytical proof may be attempted. In the process, it has been demonstrated that SHR is sufficiently expressive to describe such a protocol, and can describe its behaviour at a suitable level of abstraction to allow onward analysis, simulation, and implementation. All parts of the protocol have been modelled, from cluster formation, to cluster balancing. These models lay the groundwork for future analysis of NDC and other protocols, and the prospect of implementation of systems which are genuinely scalable, efficient and reliable, by dint of the proof of those properties as part of the design process. This type of analysis will not replace simulation, but it does provide a means for obtaining the kind of open-ended guarantees that simulation cannot give.

References

[CG04] C. Chiasserini, M. Garetto. Modeling the performance of wireless sensor networks. In *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*. Pp. –231. March 2004.

[Cur] D. Curren. A survey of simulation in sensor networks. http://www.cs.binghamton.edu/kang/teaching/cs580s/david.pdf. [Online; accessed 27-August-2007].

[FHL+05] G. Ferrari, D. Hirsch, I. Lanese, U. Montanari, E. Tuosto. Synchronised Hyperedge Replacement as a Model for Service Oriented Computing. In de Boer et al. (eds.). Volume 4111. 2005.

[GBT+04] C. Guestrin, P. Bodi, R. Thibau, M. Paski, S. Madde. Distributed regression: an efficient framework for modeling sensornetwork data. In *Proceedings of the third international symposium on Information processing in sensor networks*. Pp. 1–10. 2004.

[GEOW04] D. Gracani, M. Eltoweissy, S. Olariu, A. Wadaa. On modeling wireless sensor networks. In *Parallel and Distributed Processing Symposium*. Pp. 220–. 2004.

[God07] J. Godskesen. A Calculus for Mobile Ad Hoc Networks (Extended Abstract). In *Coordination Models and Languages*. Pp. 132–150. 2007.

[God08] J. Godskesen. A Calculus for Mobile Ad-hoc Networks with Static Location Binding. In *Proceedings of the 15th International Workshop on Expressiveness in Concurrency*. 2008.

[GSYS02] S. Ghiasi, A. Srivastava, X. Yang, M. Sarrafzadeh. Optimal energy aware clustering in sensor networks. In *Sensors Magazine, MDPI*. Pp. 258–269. January 2002.

[HCB02] W. Heinzelman, A. Chandrakasan, H. Balakrishnan. Energy efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS-33)*. January 2002.

[Hir03] D. Hirsch. *Graph Transformation Models for Software Architecture Styles*. PhD thesis, 2003. http://www.di.unipi.it/˜dhirsch.

[HKG07] M. Hammoudeh, A. Kurtz, E. Gaura. MuMHR: Multi-Path, Multi-Hop, Hierarchical Routing. In *International Conference on Sensor Technologies and Applications (SENSORCOMM2007)*. 2007.

[Inc07] O. T. Inc. The ACM international symposium on mobile ad hoc networking and computing (Mobi-Hoc). http://www.opnet.com/, 2007. [Online; accessed 2-August-2007].

[Lan06] I. Lanese. *Synchornisation Strategies for Global Computing Models*. PhD thesis, Dipartimento di Informatica, Università di Pisa, 2006.

[LLWC03] P. Levis, N. Lee, M. Welsh, D. Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. Pp. 126–137. ACM Press, New York, NY, USA, 2003.

[LT05] I. Lanese, E. Tuosto. Synchronized Hyperedge Replacement for Heterogeneous Systems. In *Coordination Models and Languages*. Volume 3454/2005, pp. 220–235. 2005.

[MS09] M. Merro, E. Sibilio. A Timed Calculus for Wireless Systems. In *3rd International Conference on Fundamentals of Software Engineering (FSEN'09)*. 2009.

[NS-07] NS-2. The Network Simulator. http://www.isi.edu/nsnam/ns/, 2007. [Online; accessed 1-November-2007].

[Ste05] D. Stein. Wireless Sensor Network Simulator v1.0. http://www.djstein.com/projects/WirelessSensorNetworkSimulator.html, 2005. [Online; accessed 18-March-2009].

[Tuo03] E. Tuosto. *Non-Functional Aspects of Wide Area Network Programming*. PhD thesis, Dipartimento di Informatica, Università di Pisa, May 2003.

[Tuo05] E. Tuosto. Tarzan: Communicating and Moving in Wireless Jungles. In Cerone and Di Pierro (eds.), *2nd Workshop on Quantitative Aspects of Programming Languages*. Volume 112, pp. 77–94. Jan. 2005.