

Emergent self-organisation of wireless sensor networks

Richard Anthony

Department of Computer Science
The University of Greenwich, London, UK
R.J.Anthony@gre.ac.uk

Julie McCann

Department of Computing
Imperial College, London, UK
jamm@doc.ic.ac.uk

Abstract

This paper describes a protocol for dynamically configuring wireless sensor nodes into logical clusters.

The concept is to be able to inject an overlay configuration into an ad-hoc network of sensor nodes or similar devices, and have the network configure itself organically. The devices are arbitrarily deployed and have initially have no information whatsoever concerning physical location, topology, density or neighbourhood.

The Emergent Cluster Overlay (ECO) protocol is totally self-configuring and has several novel features, including nodes self-determining their mobility based on patterns of neighbour discovery, and that the target cluster size is specified externally (by the sensor network application) and is not directly coupled to radio communication range or node packing density. Cluster head nodes are automatically assigned as part of the cluster configuration process, at no additional cost.

ECO is ideally suited to applications of wireless sensor networks in which localized groups of sensors act cooperatively to provide a service. This includes situations where service dilution is used (dynamically identifying redundant nodes to conserve their resources).

1. Introduction

For many advanced applications of Wireless Sensor Networks (WSN) that can be envisaged, sensors will need to do more than simply sense their environment in isolation. They might need to know their relative position or ‘embeddedness’ – in terms of the number of other sensors within communication range (i.e. their neighbours), and the current roles of those sensors. It may be sufficient to know simply the number of neighbours or perhaps their identity.

Such applications include the automatic configuration of a redundant service [1], so that sensors dynamically take over when a neighbour fails, or to automatically perform service dilution (for efficiency), and coverage configuration, so that

a given service is provided in each geographical area by one sensor only thus saving the resources of the others, yet ensuring that every service is available in every area.

In some applications the physical location of sensors needs to be known (e.g. many WSN routing algorithms require a notion of location to efficiently route their message from the source to the sink [2, 3] or sensed data may require location specific processing or identification). This information can be determined from in-built GPS; which makes sensor nodes more complex and expensive, requires more battery power and is limited to operation where the signal strength is sufficient, ruling out many buildings. Alternatively, location information can be manually provided, especially where sensors are manually positioned (e.g. fire or temperature sensors).

However, there are many applications in which it is sufficient for sensors to know their *relative* location (i.e. their proximity to neighbours). This logical topological information can be used in various ways, for example, sharing responsibility of coverage of a certain area, or election of a cluster head node. Cluster-headed, or hierarchically ordered WSNs, are used in a number of applications. In devolved control and systems maintenance, responsibility is transferred to the cluster-heads who then propagate instructions to leaf nodes [4]. Clusters allow hierarchical communication mechanisms; where the cluster head is elected to receive and pass on messages. Communication is one of the most resource hungry aspects of WSN processing. To lower the energy consumed by communication it is reasonable to reduce the power to the radio (wireless) transmitter so that only local nodes can hear a message. Essentially, clustering allows nodes to transfer data to greater than this local distance as the cluster-heads only relay the messages in a multi-hop fashion, therefore providing a more energy efficient solution [2, 5, 6]. Further, clustering can be used to achieve better scalability and robustness in ‘huge-scale control’ [7, 8]. The cluster-head can be an elected member of a cluster or it can be designated due to having a larger set of resources (such as GPRS communication, wired power, larger processing capacity etc.) than that of

the leaf nodes (which can be relatively simple sensors/loggers).

There are many reasons that the cluster/cluster-head topology should become dynamic. In routing over cluster-heads, the cluster-head elected and in fact the cluster's topology boundaries may shift and change to optimise either message delivery or network wide energy longevity. This may also happen if the system should be required to be adaptive to traffic intensities or traffic hotspotting which would drain the energy of a given set of cluster-heads. Here the cluster-head may become a leaf node and perhaps hibernate and a former leaf node become the newly elected cluster-head accordingly. Likewise, the zone covered by the cluster boundaries may change to overcome the same problems. However mobility inherent to some sensor networking applications may mean that a number of the nodes are mobile and will enter and leave the cluster, making it also dynamic. Finally, dynamic clustering may be caused by periodic hibernation of nodes to preserve energy or even the death of a sensor node due to batteries needing replacement.

Although there is a large body of research focusing on WSN hierarchies and clusters, there has been less work focusing on dynamic clustering where nodes are randomly placed, despite this being expected to be the most common deployment in the near-future. For example [9] employs cluster-heads in the centre of each cluster assuming that the topology and numbers of clusters deployed is known in advance. Further, [10] assumes a uniform distribution of nodes electing cluster-heads based on their residual energy alone. Such a mechanism would not work well if the high energy nodes were concentrated in one part of the network. Hence the need for dynamic cluster boundaries.

Other dynamic clustering mechanisms require all nodes to acquire and store global knowledge, specifically location e.g. [11]. Likewise, [12] presume that all the sensor nodes maintain a database with the location of all other nodes in the network, increasing overheads in terms of storage of the database and the update of its contents therein.

[13] re-elect cluster-heads depending on residual energy thresholds and degree of mobility, and determine cluster boundaries based on node density and not distance (therefore assuming dynamic transmission power) where the distribution of nodes is non-uniform. The dynamism is distributed and decisions made locally, however all clusterheads in the system have to be notified when an update occurs which is potentially costly.

There is also a need for sensors to be aware of their relative positions when tracking movement of objects through the sensor network. The relative positions of the sensors forms a higher-level sort of context that enables the sensors to collectively understand speed of movement and direction, and

also to detect flow trends, for example to identify large movements of people escaping a building during an evacuation, or to enable 'follow me'. An example of this is found in [14].

In summary, hierarchical configuration of sensor networks enables a meaningful compromise between local-level control of service quality, coverage, and efficiency savings by service dilution, whilst structure at the inter-cluster-head level provides efficient dynamic routing across the sensor network itself at the higher level. [15] for example, found routing over a specific aggregation tree topology improved performance and decreased energy usage.

This paper presents the ECO protocol for dynamically self-organising a WSN into logical clusters.

The ECO protocol is concerned with the dynamic configuration of clusters. It is not concerned with higher-level routing *between* clusterheads. The ECO approach is close to a small number of schemes therefore we highlight the differences below:

ASCENT [16] is a multi-hop routing protocol that dynamically adjusts the logical topology to conserve power. The core concept is to purposely deploy more sensors than necessary (to cut the costs of subsequent re-deployment), and to only use a small 'active' subset at any time, thus extending the overall lifetime of the network. Once nodes are 'active' they remain in that state until they run out of power. ECO has significantly different goals; clusters are completely reconfigurable and can serve any purpose that an application requires. Further, we do not assume or require the level of redundancy required of ASCENT.

In [17] cluster size is governed by the radius of the radio signal range and is not dynamically controllable as in ECO. Sensor nodes are statically positioned. [18] proposes a dynamic clustering algorithm based upon route efficiency, to achieve load balanced routing. The underlying assumptions concerning the sensor node equipment and the environment are different to those behind ECO in two important ways: 1. the topology is static or changes at a very slow rate allowing time for computation of optimally balanced routing; and 2. their nodes are assumed to be resource rich and have ample computing and memory resources to support complex protocols for proactive load balancing based on previous load profiles. We do not assume this as it does not fit with many of the sensor network capacities used in reality.

[14] is specifically concerned with clustering to achieve target tracking through a sensor network. Key differences from the approach taken in ECO, are that they assume static role allocations and the requirement of a static backbone of sparsely placed high-capability cluster head nodes.

The remainder of the paper is set out as follows: section 2 describes the protocol in terms of its design, operation and its novel features; section 3

presents the simulation model; and section 4 describes the evaluation carried out and analyses the experimental results.

2. The ECO protocol

2.1 Overview

The ECO protocol is totally self-configuring and self-stabilizing. Nodes initially have no external knowledge. This means they are unaware of important aspects including; their neighbourhood, in terms of identities and neighbour density; their own mobility; the extent of dynamism in their environment (including the mobility of neighbours); the required topology that they should form. From this starting point the protocol proceeds in three stages to achieve a globally stable configuration of sensor nodes into logical clusters of a dynamically selectable size. Each cluster has exactly one dynamically selected coordinator node, termed the ‘clusterhead’. In the current protocol this is chosen arbitrarily as part of the third stage of the protocol, but it would be possible to add a further operation stage to adjust the clusterhead within each stable cluster based on some qualitative characteristic such as remaining battery-power (which is very important when the cluster head acts as a link to other clusters in a hierarchical fashion).

The protocol has several novel features:

- Cluster size is not directly coupled to radio communication range or node packing density.
- The cluster head is automatically selected as a by-product of the cluster creation, at no additional cost.
- ‘Logical Triangulation’ is performed to enable nodes to learn of the true size of the cluster they are a member of, and of those that are available for them to join.
- Nodes automatically determine their own mobility, based on the rate of new-neighbour discovery.
- Natural-systems-inspired emergence techniques are employed to yield a totally decentralised system which embraces environmental randomness and, whilst being locally non-deterministic, is globally stable and convergent.

2.2 Natural Systems design paradigm

The protocol mimics the natural systems (specifically insect colonies) in the sense that it operates over a large number of autonomous actors each with only a local view of its environment and with a very simple communication model in which there is no conversation or negotiation. As with the insect colonies, the actors have common goals which they work cooperatively towards. Previous work in which this natural systems paradigm has been successfully applied include the Emergent Election Algorithm [19] and the Emergent Graph Colouring [20]. These protocols clearly demonstrate the

potential benefits of the emergence paradigm for distributed and autonomous computing.

In particular the ECO protocol mimics natural systems in the following ways:

- It embraces randomness. Randomness in the environment contributes to stability, by helping to break symmetry.
- Communication is inherently unreliable, so there is no assumption of, or reliance on, the delivery of any particular message.
- Nodes operate autonomously, and determine their actions based only on their local view (there is no global view).
- The autonomy and asynchronicity of nodes’ behaviour leads to a locally non-deterministic system. However, the system exhibits globally predictable behaviour.
- The internal decision engine at nodes is tuned such that their interactions are self-regulating. Cluster membership decisions are made primarily by a Utility Function (UF).

2.3 Messaging paradigm

Messages are transmitted via range-limited wireless communication, i.e. they propagate a fixed distance and are received by all nodes in range. A node processes all received messages. As these contain the ID of the sender (its *nodeID*), the recipient can discover its direct neighbours (i.e. those in one-hop communication range). At each node, the information contained in messages from neighbours is cached in a neighbour-vector array. This array is the basis of the node’s local view of its neighbourhood and connectivity.

All messages are deemed to have low-value. This concept is taken from the natural systems’ communication in which concepts such as checksums, acknowledgements and retransmissions do not occur. Instead the protocol is designed to be robust with respect to the loss of *any* single message.

Two message transmission techniques are used in combination: state-change-driven; backed up by pseudo-periodic slow-rate updates. The resultant communication complexity is very much lower than that of comparable ‘reliable’ protocols, as it has almost no overheads (almost everything transmitted is valuable information). See [21] for a more-detailed explanation of nature-inspired communication in distributed protocols.

Two types of message are used: *Status* messages are single-hop neighbour-neighbour communication; *Configuration* messages are external configuration commands to the sensor network and are propagated through the physical mesh network, ignoring logical cluster boundaries (which are explained in the following sections).

2.4 Multi-stage operation of the protocol

The protocol operates autonomously on each sensor node. Nodes are not initially aware of their

locations (either physically, or logically with respect to their neighbourhood). The protocol thus supports a neighbour discovery model in which a physical mesh network is automatically established as the first stage of the protocol. Topology requirements are injected into the network at a single node and are propagated through the physical mesh, in the second stage. Logical cluster configuration takes place in the third stage. The second and third stages are repeated whenever the logical topology requirements (the target cluster size) change.

Stage 1. As soon as the sensor node starts up it begins a process of discovering its neighbourhood. This is achieved by each node broadcasting *Status* messages which are received by all neighbours in range. Each node maintains a table containing neighbour ‘vectors’ (details of each known neighbour). On receipt of a *Status* message, a node either discovers a new neighbour or gets an update of the state of a previously known neighbour. In this way, a mesh topology is built up at a global level. However, each node is only aware of its connectivity with its direct neighbours. The mesh is a physical mapping, as it is based on whether or not a pair of nodes can communicate, due to wireless range.

Status messages are transmitted initially at an interval of 1000 milliseconds less a locally random component to break symmetry. However after only two such transmissions the strategy changes to state-change driven to massively reduce communication overheads.

Where there are long periods of stability at a node, a slow-rate update is sent after a number of simultaneous message slots have been omitted (actually 50, minus a locally random component of between 0 and 10) to ensure that this mechanism does not introduce an aspect of synchronised behaviour.

Each node continues the state-change based transmission of *Status* messages and monitors updates received from its neighbours after the first stage has established a stable local view of its physical connectivity. At this point the system is stable and relatively inactive.

Key content of a *Status* message includes:

- SenderID The sender’s *NodeID*.
- ClusterID Identifies which cluster the node is a member of.
- NeighbourList List of *NodeIDs* of all *known* neighbours in the same cluster.
- MobilityStatus Sender’s self-determined Boolean mobility flag.
- RandomWeight A 64-bit random value.

Stage 2 initiates the configuration of the sensor network into clusters of some externally decided size. This involves the injection of a *Configuration* message into a single arbitrary node. This message is then propagated through the mesh network. To prevent a broadcast storm, a randomly generated 64-bit *MessageID* is included, which is cached at each node. Unknown *Configuration* messages are propagated by broadcast. The original *MessageID* is retained across hops so that duplicates can be

recognised and ignored.

Stage 3 is the actual formation of clusters, in accordance with the requirements of a *Configuration* message.

Clusters are identified by the *NodeID* of the clusterhead of that cluster. Nodes are always a member of exactly one cluster. At initialisation, and at other times when no other clusters are available to join, nodes join their own cluster, i.e. they set their *ClusterID* to the value of their *NodeID*.

Cluster formation requires that nodes collate information received from their neighbours into a format that can be fed into an UF which operates independently at each node. The UF determines which of the available clusters the local node should join (or remain in) based on the ‘utility’ of doing so. The UF thus has to take account of a variety of factors which enable balanced behaviour. For example, some factors increase the ‘utility’ of clusters of the appropriate size requested, whilst others are concerned with stability; inhibiting changing from one cluster to another too frequently, or for marginal gain. Each of the factors has a ‘weight’ associated with it. This enables the UF to dynamically select the ‘best’ cluster in an optimal fashion, i.e. balancing the need for forming clusters of the correct size (some minor variation is inevitable due to the non-deterministic nature of the protocol, and the fact that the physical topology is randomised and thus exact division might be impossible), whilst ensuring stability (such as the avoidance of oscillation between clusters), and also keeping the configuration latency low and the number of messages low. The UF is ‘tuned’ by adjusting the weights of the various terms. The UF is described in section 2.8.

2.5 Dealing with dynamic neighbourhoods

The protocol is designed to operate in dynamic neighbourhoods in which nodes fail (and possibly recover), new nodes are added, and nodes move in and out of communication range with one another.

Stage 1 automatically deals with neighbour discovery arising from node initialisation or recovery, and where existing nodes come into communication range through the mobility of one or both.

It is also necessary for nodes to remove cached neighbour vectors for nodes that have either failed, or moved out of radio communication range. This is to ensure that cluster membership decisions are only based on current neighbourhood information.

Neighbour vector entries have a lifetime timestamp which is renewed each time a status update message is received from the neighbour. If a node does not hear from a known neighbour within the lifetime period, it assumes that the neighbour has failed or moved away, and drops the appropriate neighbour vector entry. If a neighbour recovers after failure (a solar panel re-charges a battery for

example), it is simply rediscovered and a neighbour vector entry is created.

The rate of each node's state-change-based *Status* message transmission activity is automatically adjusted to suit the extent of dynamism detected in its local environment. This is achieved by setting the node's state-change flag if any of the following occur: discovery of a new neighbour; loss of contact with an existing neighbour; the UF decided to change cluster membership; or a *Configuration message* is received.

2.6 Self-detection of mobility

The protocol is designed to operate in systems containing a mix of static and mobile nodes. However, nodes are not explicitly aware of their mobility. A node may be mobile sometimes and static at others; for example it may be attached to an object that is moved around inside a hospital (e.g. patient or drug tracking), factory (e.g. for stock control and positioning), airport etc. but is also left in a fixed position for lengthy periods.

In dynamic WSN configuration, mobility can be problematic if not detected; especially if the mobile node advertises its local cluster and recruits neighbours to it, and then moves off. This is potentially destabilizing.

When a node is mobile it tends to have higher rates of new-neighbour discovery and of losing contact with previously known neighbours, than a static node (the underlying mobility assumption in this particular version of the protocol being that the majority of nodes are static at any one time). Automatic detection of mobility is based on the ratio of the number of recently discovered neighbours, to the number of long-term known neighbours.

To facilitate quick determination of mobility whilst avoiding false positives; a node considers itself mobile only if the number of recently discovered neighbours is greater than 2 and at least 2/3 the number of established neighbours.

As the sensing of mobility is dynamic, a node can differentiate between periods when it is static and when it is moving.

Detection of mobility impacts on the mobile node's behaviour in several ways which are important in terms of the stability of existing clusters that it comes into contact with, and also its own cluster membership choices:

- The maximum interval between *Status* message transmission is reduced by a factor of 4. This also applies to nodes who have a neighbour claiming to be mobile (via its *Status* messages). The mobile node's higher rate of state transmission serves to allow the clusters it encounters to learn of its presence in a timely fashion, especially since it may only be in range for a short period of time. By having static nodes respond with a higher rate of transmission helps to ensure that the mobile node quickly learns of the cluster's availability.

- The caching period for known neighbour vectors is reduced at the mobile node. During times of mobility a node needs to identify quickly when it has lost contact with its current cluster.
- UF weights are adjusted to discourage a mobile node from forming its own cluster. To prevent instability arising from ephemeral clusters being advertised, the protocol has been tuned such that while a node is mobile it will join *any* available cluster it encounters with higher preference than creating a new cluster using its own cluster ID.
- Nodes who receive a *Status* message from a mobile node adjust their own UF weights to discourage them from joining *whatever* cluster the mobile neighbour advertises, unless other static neighbours also advertise the same cluster. This prevents the mobile node from acting as a fleeting 'bridge' such that a node joins a cluster that it is otherwise not in contact with.

2.7 Logical triangulation

Dynamically forming globally-correct and stable topologies in systems where nodes are completely autonomous and have to work with only a local view on their environment is non-trivial.

Physical connectivity is concerned with the number of direct neighbours (i.e. neighbours in direct communication range). Second-order physical connectivity includes neighbours of neighbours. Logical connectivity discounts physical neighbours who are not in the same logical cluster.

A key step towards achieving convergent behaviour is to enable nodes to accurately determine their current logical connectivity (i.e. the size of their current cluster), and to be able to advertise their cluster size accurately to other nodes that have to consider the relative utility of joining each of the clusters available to them.

Topological configuration requires that nodes know their second-order logical connectivity (it considers logical connectivity with direct neighbours and indirect neighbours reported by direct neighbours).

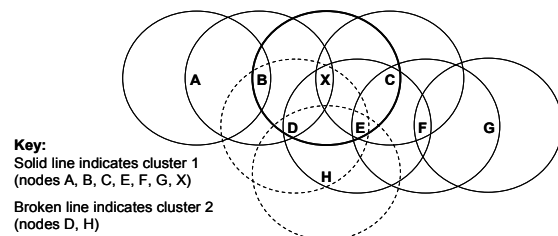


Figure 1. Illustration of the need for logical triangulation.

The following discussion refers to figure 1. The basic problem is that nodes cannot 'see' the actual physical topology, beyond knowing the identities of discovered direct neighbours; so for example, node X cannot know if its neighbours C and E are also neighbours of each other.

Without some form of physical topology information it is not possible for node X to

determine the true size of cluster 1 (of which it is a member). The method used to advertise logical connectivity must avoid two specific problems, *reflection* and *duplication*. *Reflection* occurs for example if node B informs that it has two neighbours in the same cluster. Node X cannot be certain if one of the neighbours that B is reporting is actually itself. *Duplication* occurs for example if both neighbours C and E include each other, as well as nodes G and H, when advertising their logical connectivity. Both problems lead to overestimation of the true cluster size, as seen by X.

The approach taken is to include the identities of all known *logical* neighbours (direct and indirect) in status messages. So for example B would send the vector:

Sender: B; Cluster: 1; Neighbours: A, X;

From which, X learns that A and B are in cluster 1. Note, B does not advertise D, since it is not a logical neighbour (it is in a different cluster).

F sends the vector:

Sender: F; Cluster: 1; Neighbours: C, E, G;

So both C and E will be aware of their logical neighbours F and G. C and E now send, respectively:

Sender: C; Cluster: 1; Neighbours: E, F, G, X;

Sender: E; Cluster: 1; Neighbours: C, F, G, X;

X stores these vectors, because message transmission is asynchronous and state-change based, so two-such messages cannot be compared instantaneously. When they are compared the duplicates are removed and X can determine that the size of the cluster is 7, including itself. D sends:

Sender: D; Cluster: 2; Neighbours: H;

So X is aware that it has a choice between cluster 1 with a total membership of 7; and cluster 2, which would have size 3 (including X).

The Utility Function must be carefully tuned so that X's *local* decision meets a critical compromise between achieving *globally* correct topology, taking into account the current cluster size goal; whilst being locally stabilising (i.e. having a minimal knock-on effect). For example, what should it do if the desired cluster size is 5? This dilemma implies that local deviations from the exact desired cluster size have to be permitted, and that nodes must be able to 'switch off' their optimisation logic to avoid expending too much effort trying to resolve such deviations.

2.8 The Utility Function

A Utility Function (UF) is the core decision engine that determines whether a node should form its own cluster, or join an existing cluster, and if so, which one.

The UF should minimise the number of cluster membership changes to ensure stability. This has to

be balanced by the need for the node to join the most appropriately sized cluster, and to discourage the node from being logically isolated (forming its own cluster with no other members).

The factors that the UF takes into consideration when choosing which cluster the node should join include:

- Whether the node is a 'leaf' – i.e. does it only have a single neighbour?
- Does the node have attached leaf nodes which are dependent on it?
- What is the desired cluster size and what are the sizes of each available cluster?
- Is the node mobile?
- Is the node the clusterhead of an available cluster?
- Are any of the node's direct neighbours the clusterhead of an available cluster?
- Is the node already a member of one of the available clusters?

In order that the UF is not too complex, it is complemented with a few special rules that deal with certain tricky scenarios. The rules are evaluated *before* the UF, because they adjust weights used subsequently by the UF.

Rule 1 resolves a utility dilemma in which a logically isolated node is neighbouring one or more stable clusters; but in which none of the clusterheads are a direct neighbour of the node. The rule identifies the smallest of the neighbouring stable clusters and adjusts weights to encourage the UF to favour that cluster.

Rule 2 resolves the situation where there are many small clusters that need consolidating into fewer larger clusters. The situation is locally detected when the node is a clusterhead of a stable cluster (it has at least one direct neighbour member), which is undersize by at least 2 nodes. A direct neighbour is a clusterhead with a stable undersize cluster at least as big as the local cluster. If the node were to abandon its current cluster and join its neighbour's cluster, that cluster would be closer to the target size than *either* is at present.

Rule 3 checks if the node is a member of a cluster in which the clusterhead itself is not present.

2.9 Auto-detection and damping of oscillation

The UF has to deal with a large number of different scenarios, and thus tuning it to behave satisfactorily under all conditions is difficult. Inevitably, because of factors which include: the total autonomy of nodes, environmental randomness, propagation lag of cluster information, and the various aspects of non-deterministic behaviour that result, it is not possible for the UF to always decide clearly amongst the possible clusters available to a node. There can always be a marginal situation in which at one time a particular cluster is favoured, whilst a short time later (typically because of receiving a more-recent *Status* message) the cluster preference changes. In isolation such changes

are natural and do not adversely affect stability. However, occasionally it was found that a node would oscillate between a pair of clusters having very similar utility values. This particular behaviour is potentially destabilizing and needs to be damped.

To detect oscillatory behaviour a node keeps a table containing its recent cluster membership decision history. During UF activation the table is examined. If a decision pattern indicating flipping amongst two or more clusters is discovered the UF is sidestepped and the cluster ID is not changed. This technique typically manages to damp oscillations within two or three cycles.

2.10 Randomness employed in the protocol

Several sources of environmental randomness are tolerated, and further randomness has been purposely inserted into the operation of the protocol. This represents the most significant departure from traditionally designed deterministic systems, in which randomness is avoided or removed at all opportunities.

The exploitation of randomness is an extremely important aspect of the protocol's design and is fundamental to achieving stable convergent behaviour despite the totally autonomous behaviour of possibly hundreds of devices with the same goals. Randomness is employed, or occurs naturally, in several ways:

- A random node weight is used during logical triangulation when determining how to associate neighbours-of-neighbours to one of several intermediate neighbours.
- A random cluster weight is used in the UF as a tie-breaker when two or more clusters have the best utility.
- Random timeout components are built into the minimum interval between state-change-driven *Status* message transmissions, and the interval between the slow-rate *Status* message transmissions that occur in times of stability, to avoid accidental synchronization through communication.
- Event timing at nodes is naturally random, due to their complete autonomy and the asynchronous nature of the protocol.

This randomness makes the algorithm non-deterministic, yet also helps it to overcome symmetry and thus cuts out the need for negotiation or voting (and the accompanying heavy message overheads). The symmetry breaking (each node having a unique local view of the system) is key to stability. The resultant algorithm is sufficiently self-regulating that it can tolerate additional randomness in its environment, such as message loss, node movement, and node failure and recovery (see next section).

3. The model

3.1 Assumptions and general configuration

The current model assumes homogenous sensor nodes with respect to their processing and memory capacities, and with respect to their radio communication range. These assumptions hold true for typical current WSN deployments.

The simulation operates in discrete steps, each representing 1 millisecond of elapsed time. This has been found empirically to provide a suitably-fine granularity between protocol-level events which operate at a much coarser granularity of the order of 1000 milliseconds and greater. Each simulated node has its own event queue.

3.2 Topologies and mobility

A number of standard topologies are supported for behaviour evaluation, representing a progressive series from the simplest possible interaction with just two nodes to highly complex symmetrical patterns. The latter were initially highly problematic with respect to stability, and were thus especially useful in tuning the UF so that it is stable over a wide range of scenarios. In some cases particularly difficult modes of interaction were identified that needed to be handled by the introduction of special rules, although the number of such rules has been kept as low as possible. The standard topologies are shown in figure 2.

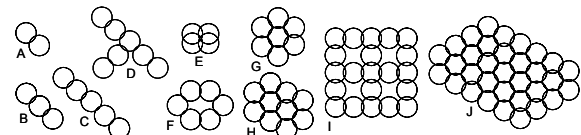


Figure 2. The standard topologies set.

Random topologies containing between 2 and 1000 nodes are also supported, in which node density can be controlled. An example random topology is shown in figure 3.

Much of the current work on sensor network organization and clustering assumes either regular topologies and/or even packing density. For example [17] states “*Most of the existing algorithms are well suited for uniformly deployed sensors in the field.*” See also the discussion in section 1.

Whilst the work presented in this paper uses such fixed topological constraints to identify specific problems and types of interactions in a repeatable-experiment fashion (see the standard topologies above); the protocol is fundamentally designed to operate in totally random deployments (in terms of the number of nodes, their physical and relative locations, and packing density. Such random deployment is very realistic, at least for the large scale ‘smart dust’ style applications. In these applications the nodes are simply dropped into their environment (possibly literally, from an aircraft for example) and left to get on with it. It is not realistic for such deployments to assume any form of

regularity.

There are also many different models of mobility discussed in the sensor network and mobile computing literature. Extreme examples are from full nomadic computing where people armed with PDAs roam a tourist landscape reading location specific information, to RFID tags on drugs being tracked by sensors so that the correct drug is in the correct container from the production line to the hospital, to military target tracking through a field of smart dust sensors. As this is a very application-dependent aspect, the protocol has been designed to cope with various degrees of mobility. The simulation model is suitably flexible to enable evaluation of the protocol in this respect. The model supports the addition of a mobile node to any of the topologies (including random). The mobility of the node can be intermittent, and its speed of movement and the randomness of its trajectory can be controlled.

3.3 Definition of convergence

Due to the dynamic and non-deterministic nature of the protocol and of the actors over which it operates, full convergence in which every node is completely stable is not always possible; especially in high-density scenarios with significant node overlap. However, even in the most complex topologies, 98% convergence (in which 98% of the nodes are stable over period of 3000 time-steps, which equates to three times the nominal interval between *Status* message transmission) is reached quite quickly. The remaining small number of nodes may take significantly longer to settle down. Waiting for this (which is not guaranteed to actually ever happen), distorts the statistics because the system is essentially stable much sooner.

The experimental results presented in section 4 are thus measured to 98% convergence, with a minimum of *two* nodes allowed to be unstable. The mobile node causes localized disturbances so the overall convergence in large systems is not significantly affected. There is however a larger effect in smaller systems, so when the mobile node is actually moving (its movement can be frozen) measurements are to 98% convergence (with a minimum of *five* nodes allowed to be unstable).

3.4 Message delivery and loss

All messages are sent as local broadcasts. The sender does not know the identity of the recipients. A mechanism in the model determines which nodes are currently in range to receive the particular message; and copies the message into each node's message queue. The time at which the message is actually delivered to each recipient is offset a short random time into the future (between 1 and 10 milliseconds) to reflect the various sources of latency that impact on the end-to-end communication.

Message loss is determined randomly, at the point of message generation. Messages are deemed lost fundamentally through interference and collisions in wireless transmission, so the model enforces that a message is either delivered correctly to all recipients, or is totally lost (not delivered to any recipients).

3.5 An example simulation with a random topology

Figure 3 shows a snapshot of a random topology simulation of a 51 node system, in which one node is mobile. The visual display uses different colours to represent cluster membership; a solid colour indicating membership of that cluster, and a hollow square indicating the cluster head. Thus, to aid clarity (when presented as grey scales), the cluster groups have been encircled with dotted lines.

The snapshot was taken after a stage 2 command to configure into clusters of 4 nodes, and subsequent stage 3 convergence had been reached. Each cluster has exactly one clusterhead. The mean cluster size is slightly larger than the 4-node target, at 4.25 nodes.

The simulation model allows the mobile node's movement to be stopped / started to simulate realistic scenarios where a sensor is attached to a device such as a trolley, vehicle or animal, that has periods of movement and rest. When the snapshot was taken the mobile node was moving and, through analysis of the pattern of neighbour-discovery events, has correctly identified its mobility state as such (the solid black square in the centre of its icon indicates this). The other nodes are all static but do not explicitly know this. They have each also used the pattern of neighbour-discovery events to correctly *determine* that they are static.

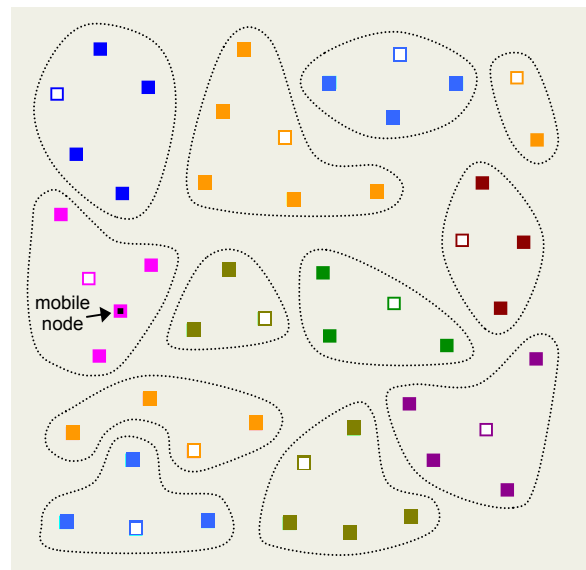


Figure 3. A random topology with 50 static and 1 mobile node.

4. Evaluation

The following notation is used:

- N Number of sensor nodes.
- R Nodes' communication radius.
- V Maximum communication overlap, in terms of percentage of a node's communication diameter.

All experiments were based on random topologies. The model allows topologies of varying density to be created, by changing the values of N , R and V . To ensure consistent neighbourhood density, permitting meaningful comparison across the results, the following $\{N, R, V\}$ tuples were used: $\{50, 45, 30\}$, $\{100, 30, 30\}$, $\{150, 25, 30\}$, $\{200, 22, 30\}$, $\{250, 20, 30\}$, $\{500, 14, 30\}$, $\{750, 12, 30\}$ and $\{1000, 10, 30\}$. Results are based on mean values over ten simulation runs for each configuration.

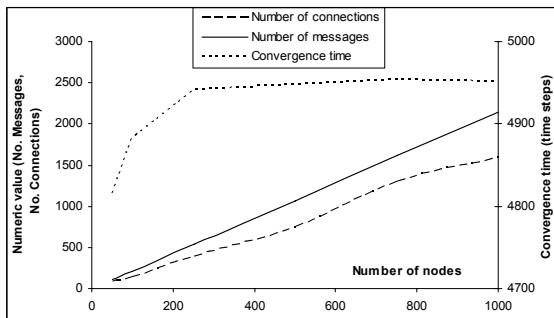


Figure 4. Scalability of stage1 of the protocol.

Figure 4 shows the performance of stage 1 of the protocol as the number of nodes increase. The near-linear relationship between the number of nodes and the number of connections illustrates that node density was indeed consistent throughout the experiments (these are physical connections, i.e. the nodes are in communication range, as opposed to logical connections, in which nodes share cluster membership). Good scalability is demonstrated by the linear relationship between the number of nodes and the number of messages generated.

The remainder of the evaluation is concerned with stage 3 of the protocol, which is the most interesting in terms of emergence and the most important in terms of sensor node configuration.

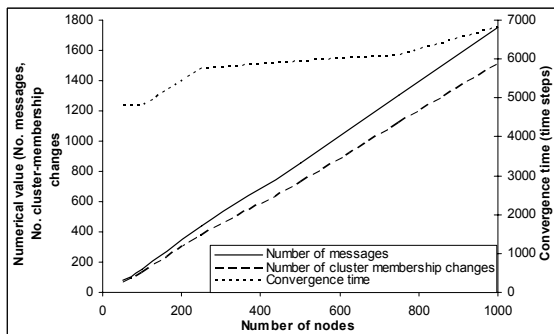


Figure 5. Scalability of stage3 of the protocol.

Figure 5 illustrates the scalability of stage 3 of the protocol: the number of messages generated to converge the configuration increases linearly in the number of nodes. For this experiment the target cluster size was always 4. The number of cluster membership changes is acceptable (with a mean of approximately 1.5 changes per node across all system sizes). To put this into context, even in 1000-node systems the convergence time is below 7 seconds wall clock time (one time-step in the model is equal to 1 millisecond of wall-clock time).

Convergence time increases slowly as the system size grows. This is because the interactions are localised and the UF has been configured to minimise the knock-on effects of local disturbances.

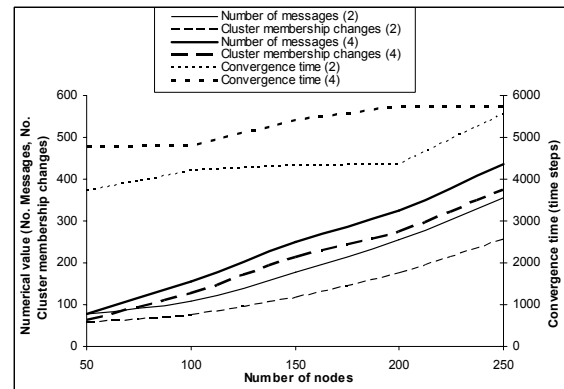


Figure 6. The effect of target cluster size on performance.

Figure 6 compares the performance when configuring clusters of two nodes and clusters of four nodes. The complexity of the cluster to be formed has a measurable, yet affordable impact on convergence time, number of messages sent and the number of cluster-membership changes that occur during convergence.

A key example of the organic behaviour of the protocol is that it is configured to trade some error in the achieved cluster sizes, for efficiency and low latency (which in the context of sensor networks are more important). The effect of this is that the standard deviation of cluster size error tends to be consistently around 1.5. This significantly reduces the optimisation overhead. The UF could alternatively be tuned such that nodes continue to shuffle their cluster memberships until they reach the desired size, but this can lead to significant convergence times and high numbers of messages generated.

The presence of a mobile node is found to slightly increase the convergence time and the number of messages generated, as it causes some localised disturbances. The UF is tuned such that static nodes mostly simply ignore the presence of the mobile node. The mobile node continually changes its cluster membership as its dynamic neighbourhood dictates.

5. Conclusion and further work

The ECO protocol for self-organising sensor nodes into logical clusters has been described and its performance evaluated. The protocol is based on engineered emergent behaviour to arrive at stable cluster configurations of a dynamically and externally specified size (in terms of the number of nodes they comprise). The protocol relies on a mix of environmental and injected randomness to ensure that the totally autonomous nodes have unique local views of the system, which is key to stability.

In addition to clearly demonstrating some fundamental concepts of engineered emergence, ECO introduces some novel features and uses of emergence. These include: self-stabilisation through monitoring of the decision behaviour of the UF and applying internal corrective feedback; the use of a small number of policy rules to deal with specific challenging scenarios by dynamically adjusting the UF weights to retain compromise between stability, fast convergence and topological correctness despite the UF's many degrees of freedom; and dynamic self-determination of node mobility, based on patterns in neighbour discovery.

As the actual configuration of clusters is emergent, it is implicitly non-deterministic. This means that it is not possible to predict or guarantee the exact size of any cluster, or which cluster a specific node will join. However, by careful tuning of the utility function which is the core decision maker (along with the addition of a few rules to handle some special scenarios), it has been possible to ensure that the localised, short-sighted actions of large numbers of independent nodes yields globally convergent behaviour.

Imperial College, London, UK, have developed custom sensor nodes. These 'Beasties' [1] are towards the very-low-power, low-cost end of the sensor node spectrum, and are designed with a view towards mass deployment 'smart-dust' applications. The characteristics of these devices have influenced the design of the protocol itself, and the simulation experiments. It is intended that the protocol be deployed over a Beastie network.

It is also intended that higher-level applications that work with patterns in data collected across the whole sensor net, as opposed to specific sensor readings, are deployed across the dynamically configured clusters.

References

[1] McCann J.A, Huebscher M, Hoskins A, Context as Autonomic Intelligence in a Ubiquitous Computing Environment, *International Journal of Internet Protocol Technology (IJIPT) special edition on Autonomic Computing*, Inderscience 2007, 2 (1), pp. 30 - 39.
[2] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, Energy-Efficient Communication Protocol for Wireless Microsensor Networks. *Proc. Hawaii Conf. on System Sciences*, Jan. 2000.

[3] Papadopoulos A., Univaq A., McCann J.A., Connectionless Probabilistic (CoP) routing: an efficient protocol for Mobile Wireless Ad-Hoc Sensor, *24th Intl. Performance Computing and Communications Conference*, IEEE, Arizona March 2005.
[4] Q. Fang, J. Li, L. Guiba and F. Zha, RoamHBA: Maintaining Group Connectivity In Sensor Networks, *Proc. 3rd Intl. symposium on Information processing in sensor networks*, Berkeley, CA, USA, ACM, 2004, pp. 151-160.
[5] Ossama Younis and Sonia Fahmy, HEED: A hybrid energy efficient, distributed clustering approach for Ad-hoc sensor networks, *IEEE Transactions on Mobile Computing*, Oct. 2004, 3(4), pp. 366 - 379.
[6] Seema Bandopadhyay, and Edward J.Coyle, An Energy efficient hierarchical clustering algorithm for wireless sensor networks, *Proc. of IEEE INFOCOM 2003*, San Francisco, CA.
[7] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, Next century challenges: Scalable coordination in sensor networks, *Proc. Mobicom 1999*, Seattle, p263-270.
[8] Y. Shang and H. Shi, Flexible Energy Efficient Density Control on Wireless Sensor Networks, *Intl. Journal of Distributed Sensor Networks*, Taylor & Francis, 3 (1), 2007, pp. 5 - 21.
[9] Yuon Guo, Janise McNair, Cost Efficient Cluster Formation for wireless sensor networks, *Proc. of CITSA 2004*, Orlando, FL, July 2004.
[10] Ossama Younis and Sonia Fahmy, HEED: A hybrid energy efficient, distributed clustering approach for Ad-hoc sensor networks, *IEEE Transactions on Mobile Computing*, Oct.-Dec. 2004, Volume: 3, Issue: 4, p.p. 366 - 379.
[11] Ya Xu, Solomon Bien, Yutaka Mori, John Heidemann, Deborah Estrin, and Alberto Cerpa. Topology Control Protocols to Conserve Energy in Wireless Ad Hoc Networks. *CENS Technical Report 0006*, January 2003.
[12] S. Lindsey and C. Raghavendra, PEGASIS: Power Efficient Gathering in Sensor Information Systems, *IEEE Aerospace Conference*, Volume 3, 9-16 March 2002, pp. 3-1125 - 3-1130.
[13] Jyun-Yuan Cheng, Shanq-Jang Ruan, Ray-Guang Cheng, Teng-Tai Hsu, PADCP: Power-Aware Dynamic Clustering Protocol for Wireless Sensor Networks, *IFIP Intl. Conf. Wireless and Optical Communications Networks*, 11-13 April 2006.
[14] Lui Sha, Jennifer C. Hou, Wei-Peng Chen, Dynamic Clustering for Acoustic Target Tracking in Wireless Sensor Networks, *IEEE Transactions on Mobile Computing*, 3 (3), July 2004, pp. 258-271.
[15] Guozhen Tan, Ningning Han, Yi Liu, Jialin Li and Hao Wang, Wireless Network Dynamic Topology Routing Protocol Based on Aggregation Tree Model, in: *ICN/ICONS/MCL 2006. Intl. Conf. Systems and Intl. Conf. Mobile Communications and Learning Technologies*, April 2006, pp. 128- 128.
[16] A. Cerpa and D. Estrin, ASCENT: Adaptive Self-Configuring sEnor Networks Topologies, *Transactions on Mobile Computing*, (3), pp. 272-285, July 2004, IEEE.
[17] Distributed Dynamic Clustering Algorithm in Randomly Deployed Wireless Sensor Networks S. Kher, P. Speck, and A. Somani, Technical report, 2006. <http://www3.ee.iastate.edu/dcnl/Publications/docs/Tech-rep/DCNL-ON-2006-01.pdf>
[18] M. Iqbal, I. Gondal and L. Dooley, An Energy-Aware Dynamic Clustering Algorithm for Load Balancing in Wireless Sensor Networks, *Journal of Communications*, 1(3), pp. 10-20, June 2006, Academy Publisher.
[19] Anthony R, Emergence: A Paradigm for Robust and Scalable Distributed Applications, *First Intl. Conf. Autonomic Computing (ICAC)*, New York, USA, May 2004, pp. 132-139, IEEE.
[20] Anthony R, Emergent Graph Colouring, *Engineering Emergence for Autonomic Systems (EEAS), First Annual International Workshop, at the third International Conference on Autonomic Computing (ICAC)*, Dublin, Ireland, June, 2006, pp. 4-13, CMS press.
[21] Anthony R, An Autonomic Election Algorithm based on Emergence in Natural Systems, *Integrated Computer-Aided Engineering*, 13 (1), pp. 3-22, January 2006, IOS Press.