All Roads Lead to Rome: Data Highways for Dense Wireless Sensor Networks

David Lowe[1,2] and Daniele Miorandi[2]

[1] Centre for Real-Time Information Networks, UTS, Ultimo, NSW 2007, Australia
`david.lowe@uts.edu.au`
[2] CREATE-NET, v. alla Cascata 56/C, 38100 – Povo, Trento, IT
`{daniele.miorandi,david.lowe}@create-net.org`

Abstract. The design of efficient routing algorithms is an important issue in dense ad hoc wireless networks. Previous work has shown that benefits can be achieved through the creation of a set of data "highways" that carry packets across the network, from source(s) to sink(s). Current approaches to the design of these highways however require a–priori knowledge of the global network topology, with consequent communications burden and scalability issues, particularly with regard to reconfiguration after node failures. In this paper we describe an approach to generating these data highways through a distributed reaction-diffusion model that uses localised convolution with activation-inhibition filters. The result is the distributed emergence of data highways that can be tuned to provide appropriate highway separation and connection to data sinks. We present the underlying models and the algorithms for generating the highways, as well as preliminary simulation results.

Key words: wireless sensor networks, routing, data highways, activation–inhibition mechanisms, reaction–diffusion patterns

1 Introduction

A key issue in dense ad hoc wireless networks is the design of efficient routing algorithms. This can affect the performance of the resultant system in terms of power efficiency, communication latency and robustness. For example, an efficient routing algorithm can lead to reductions in both the number of network nodes that need to remain awake to route traffic and the total transmission power required for the multi–hop communication along the routing path from data source to data sink.

Previous work [10] has shown that the creation of a set of wireless "backbones" or data highways that carry packets across the network, from source to sink, can provide a network capacity that follows the same pattern for randomly located nodes as can be achieved for arbitrarily placed nodes. In effect, the highways are constructed such that every source node is within range of at least one highway (implying it can access it in a single hop). The highways then drain packets to the sinks along a series of shorter length hops, with correspondingly lower power requirements and hence a lower interference footprint.

In previous work, these highways have been constructed based on approaches such as percolation theory [10]. This has the disadvantage that it requires an a–priori analysis of the entire network structure, with the consequence that the approaches cannot readily accommodate randomly placed nodes unless there is a mechanism for determining and communicating node location — a constraint that adds a layer of complexity and a performance burden. It also typically makes the network less robust, as any change (such as a failure or location change of a highway node) requires a global recalculation of the routing pathways.

In this paper we discuss an approach to addressing this problem through distributed determination of the data highways based on an activation-inhibition diffusion that generates optimal highway separation. We argue that this approach represents a significant contribution, insofar as it will improve robustness and allow localised self-healing of the data highways – an important characteristic of dense networks with randomly placed nodes.

In section 2 we discuss previous work in this area and in particular on the application of distributed diffusion model for engineering the emergence of patterns in large–scale networks. Following this, in section 3, we consider the models that underpin the highway generation and the mechanisms that we have used for their distributed construction. Section 4 discusses the way in which data is then routed within this data highway system, and section 5 presents preliminary results and analysis showing the performance of the approach. Finally, we present our conclusions in section 6.

2 Related Work

Wireless sensor networks (WSNs) [1] have become an important tool in many real–world settings. For example, they are being increasingly used for the monitoring of environmental parameters, where nodes are deployed over space, each node sensing a given environmental parameter (temperature, light, level of pollutants etc.). In most real–world settings multiple sinks are present, each sink being connected, usually through some form of long–range wireless communications, with a remote data center where information is processed. When fine–grained information is needed, the net result can be – from a communications perspective – the formation of dense WSNs.

Current state–of–the–art routing schemes for WSNs, for communicating sensory data to the data sinks, are most often based on the construction of trees, a structure that lends itself naturally to perform en route aggregation of data [7]. For dense WSNs, various authors have proposed optimal routing strategies based on the use of a continuum model of node placement over space. In particular, routing strategies based on analogies with physical phenomena have been proposed (optics [12, 6] and electromagnetism [17]), as well as routing strategies built using models inspired by road traffic engineering [2].

In their seminal work, Gupta and Kumar [11] proved that the communication capacity of (dense) ad hoc networks, with n nodes withi na given area, can scale, in terms of per source–destination throughput, as $\frac{1}{\sqrt{n}}$ bit/s in the case of arbitrary node placement and as $\frac{1}{\sqrt{n \log n}}$ in the case of randomly (uniformly) located nodes. The existence of such a gap gave rise to various investigations, aimed at finding suitable strategies for closing it. This is particularly important given that many (indeed possibly most) sensor networks are likely to have nodes that are randomly placed.

In a series of papers, Franceschetti et al. [10] demonstrated, using tools and results from percolation theory, that such a gap could be closed by introducing non–uniform transmission power schemes. The concept of data highways was introduced in [10], where it was shown that it was always possible to build high–throughput paths crossing a given section of the network. Such paths result in high–throughput as they can be operated at very low transmission power, hence limiting self–interference. The optimal routing strategy would then be: (i) from a source reach the closest highway with a single (possibly high–power) hop; (ii) route packets along the highway using low–power hops; (iii) drain packets from the highway to the appropriate sink when in proximity. Such a scheme was shown to be able to attain a throughput per source–destination pair of the order of $\frac{1}{\sqrt{n}}$, thereby effectively closing the gap in the Gupta–Kumar result. A sample representation of a set of data highways is reported in Fig. 1 (taken from [10]).

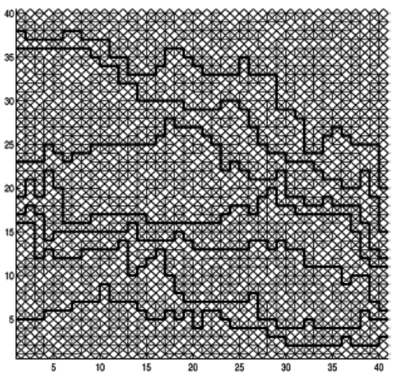


Fig. 1. Typical data highways through a 40x40 grid, obtained using a bond percolation model (from [10]).

The intuition behind such a result is that by using highways it is possible to limit mutual interference among nodes in the network. To the best of the authors' knowledge, however, such a result has not yet found any application to routing problems in realistic environments. The main difficulty is the construction of highways. In order to have a feasible solution (in terms of scalability and complexity), the formation of highways should be achieved by means of a distributed process based on local information only. Rather than a global design of the highways, the nodes should self–organize to achieve the desired spatial structures that can in turn be used to generate the data highways. This raises the question of what form of self-organising process may be relevant in this particular scenario?

The use of self–organizing processes has seen wide consideration. For example, their use for building “spatial” computers was at the heart of the MIT's Amorphous Computing initiative[1]. One of the application scenarios envisioned for the (programming) techniques developed within the framework of such an initiative was to engineer the emergence of structures in sensor–actuator networks [4].

The use of related self–organizing spatial processes for building overlays found applications in various networking fields. In WSNs, Bicocchi et al. proposed to use a field–based mechanism for aggregating WSNs into logical neighborhoods [5]. In peer–to–peer systems, probabilistic distributed mechanisms were proposed by Jelasity et al. for dynamically rewiring links in overlays [14, 13]. In particular, it was shown that, by relying on local interactions only, it was possible to build system–level structures with desired topological properties.

In [16] Saffre and Shackleton propose the use of an embryogenies–inspired mechanism for efficiently allocating 'roles' in an autonomic manner in a peer–to–peer service infrastructure. In such a work, nodes communicate via gossiping techniques, and differentiation decisions are taken at each node based on the nodes's current status and the status of neighbours. The mechanism is reported to be able to build efficient structures (in terms of topology and role assignments).

Possibly the most relevant form of self-organising process to our particular problem are reaction-diffusion processes. These processes are the basis of various natural mechanisms that result in the emergence of patterns, in particular of cell differentiation and morphogenesis [3]. The use of reaction–diffusion processes (and in particular of activation–inhibition mechanisms) has been proposed in the context of ad hoc networks to deal with activation problems. In particular, it has been proposed by Durvy and Thiran for dealing with activation at the MAC level [9] and by Neglia et al. for addressing clustering problems in WSNs [15]. In both cases, the pattern to be created presents isolated activation peaks (corresponding to 'active nodes') divided by large valleys. Such patterns are different from those needed to engineer data highways, which require the creation of zebra–like stripes, which should furthermore converge to one of the sinks present in the network. We will consider these processes in more detail, and how we might adapt them, in the following section.

3 Models and Mechanisms for Data Highways Formation

3.1 Reaction diffusion modeling

As outlined above, we wish to develop self-organising processes that lead to the natural emergence of data highways through a wireless network. These highways

[1] `http://groups.csail.mit.edu/mac/projects/amorphous/`

need to be optimally spaced such that all nodes are within a single hop of a highway, but the highways themselves utilise short-range hops to transport data to any data sink. This concept was shown in Figure 1. Further, the highways should be able to be derived only through local interactions between nodes.

In developing an approach to this problem, we have taken inspiration from mechanisms that utilise activation-inhibition reaction-diffusion [3, 9, 15]. These mechanisms describe how a field strength variable or substance concentration within each cell or node can vary in space and time under a pair of competing influences – a short range positive activation region within which the field of neigbouring cells is strengthened, and a longer range negative inhibition region within which the field of neighbour cells is retarded – with the resultant emergence of specific patterns when the effects are diffused through the network. The resultant models have been used widely to describe behaviours in biological and physical processes (see [8] for a discussion). The simplest formulation of this approach, using a single field variable, is modelled in the discrete time domain as follows:

$$u(\mathbf{k}, t+1) = g\left(\varphi_s u(\mathbf{k}, t) + \sum_{\mathbf{j} \in R_i} \varphi_i(\mathbf{j}) u(\mathbf{k}+\mathbf{j}, t) + \sum_{\mathbf{j} \in R_a} \varphi_a(\mathbf{j}) u(\mathbf{k}+\mathbf{j}, t)\right) \quad (1)$$

where $u(\mathbf{k}, t)$ is the field strength in cell $\mathbf{k}$ at time t, R_i is the region over which the inhibition function φ_i is applied, R_a is the region over which the activation function φ_a is applied, and $g()$ is a limiting function. The activation functions are time invariant, and applied uniformly across the sensor field. Note that this is equivalent to the convolution of $u(t)$ with the sum of φ_i, φ_a and the self-activation value φ_s. Note also that, in general, it is assumed that φ_i takes negative values, while φ_a and φ_s take positive values.

As discussed in the previous section, recent work has adapted reaction diffusion models to the design and/or configuration of wireless networks. As an example, Neglia and Reina [15] have used activator-inhibitor diffusion to select active nodes within a dense wireless sensor network. The nodes have deeply overlapping sensing fields, and hence only a small number of nodes are required to be active in order to adequately provide full data on the region to be sensed. The operation of this approach can be seen in Figure 2. A random dense wireless sensor network (Figure 2a) is repeatedly convolved with a symmetric 2-dimensional diffusion filter (Figure 2b). The resultant field strength after 20 iterations of a filter[2] (Figure 2d) is then analysed to determine local maxima (Figure 2e) – which represent the nodes to be activated. All other nodes can be switched to a low-power non-sensing state. The result is a distributed process for identifying a subset of nodes to be activated, such the nodes are suitably distributed.

The repeated filter convolution causes the emergence of the peaks in the activation field by activating localised regions whilst inhibiting the areas between

[2] The filter used in this case was a simplified version, with a central self-activation strength $\varphi_s = 2$, a flat activation ring of strength $\varphi_a = 1$ and radius 1, and a flat inhibition ring of strength $\varphi_i = -0.1$ and radius 6.

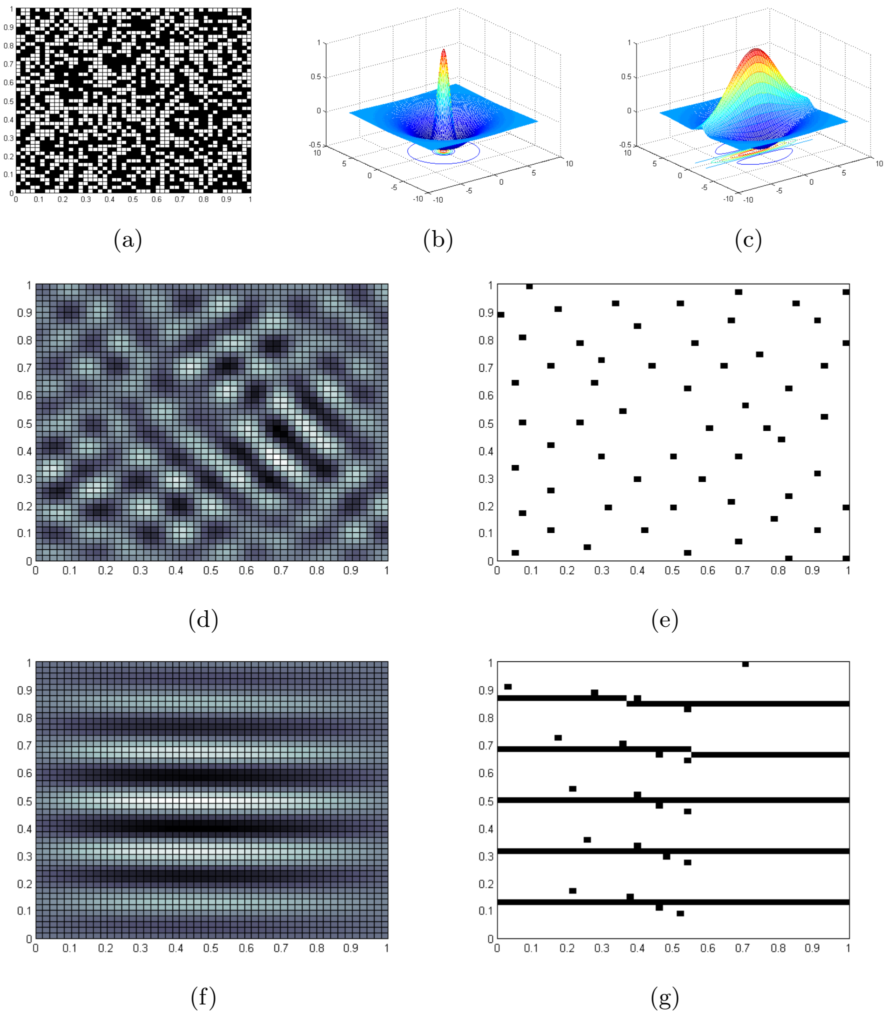


Fig. 2. Sensor activation/inhibition: (a) the sensor field; (b) symmetric filter; (c) rotationally asymmetric filter; (d) activation field resulting from symmetric diffusion filter; (e) detected peaks in field; (f) activation field resulting from asymmetric diffusion filter; (g) detected ridges in field.

these regions. The width of the inhibition zone controls the separation of the resultant peaks and the width of the activation zone controls the kurtosis of the peaks. It is therefore possible to select filter parameters that ensure that an optimal density of active nodes is obtained. The filter used in the activation-inhibition diffusion can be readily implemented in a distributed fashion, provided each node has knowledge of the diffusion filter parameters to be used. Each node communicates with its neighbours, and acts to either strengthen or weaken their resultant activation field. The nodes also can then through comparison with neighbour nodes' current activation strengths, determine whether they are at a local maxima and hence should be active. This means that this approach can be fully distributed within a wireless network, leading to active node selection with no network-wide oversight.

3.2 Highway generation

What we are seeking in the design of the data highways is analogous to this – the localised selection of the nodes which form the highways without any global design or control. The natural question to ask is therefore whether or not an activation-inhibition diffusion model can be adapted such that the result is connected nodes that together make data highways, rather than single active nodes.

The solution is based on changing the nature of the diffusion filter. Previous work on wireless networks has used symmetric filters, leading to the emergence of patterns that have isolated peaks in the sensor activation field. Work in other areas (e.g. [8]) has shown that changing the nature of the diffusion filter can lead to changes in the patterns that emerge in the activation field. As an example, consider the bottom row of Figure 2. In this case the random dense wireless sensor network is repeatedly convolved with a rotationally asymmetric filter that has a dominant horizontal activation axis, whilst inhibiting along the vertical axis (Figure 2c). The resultant field strength patter after 20 iterations of a filter that has this structure is shown in (Figure 2f). This has developed a striped pattern of ridges and troughs, with the orientation controlled by the orientation of the activation axis in the filter and the separation determined by the range of the filter inhibition. The ridges in this pattern can then be used to determine local ridge maxima (Figure 2g) – giving the potential data highways that we are seeking. Nodes at ridge maxima become highway nodes, and all other nodes communicate with the highways in order to deliver data to desired data sinks. By tuning the filter parameters appropriately, we can control the separation between the highways, and thereby ensure that all non-highway nodes are within a single hop of a highway.

3.3 Controlling highway orientation and destination

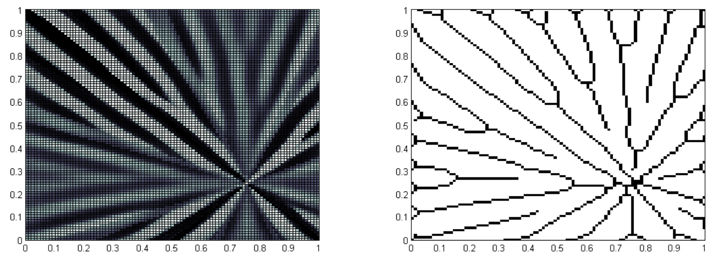
Having demonstrated that it is possible to generate 'striped' patterns that can be used to generate highways, we next consider the question of how we orient these stripes so that the highways converge on data sinks. One possibility is to construct artificial "bridges" between the highways and relevant sinks (taking into account the overall topology as well as load balancing between the sinks). These bridges can be either multi-hop paths, or could be a single high-power connection. In either case, this solution requires the artificial construction of additional routes[3].

[3] This is actually the solution originally discussed in [10]

An alternative is to investigate whether it is possible to control the directions of the highways such that as the ridges are generated during the diffusion process, they naturally converge to the data sinks. Given that the orientation of the ridges are controlled by the orientation of the activation axis in the filter, we should therefore be able to create ridges that converge on a sink by making the direction of the filter activation axis spatially dependent.

Consider the case of a network field with a single sink. Each node in the network determines the direction to that sink, and rotates its local instance of the diffusion filter so that the activation axis is aligned with the direction to the sink. The diffusion filter contains a square inhibition zone R_i of dimension $2R \times 2R$ (with an inhibition level of φ_i, and a wedge-shaped activation zone R_a with a angular size ρ (with an activation level of φ_a. We rotate the filter by rotating just the activation band within the overall square filter.

Applying this filter so that the activation band is always oriented towards the single sink gives an activation field as shown in Figure 3a, with all ridges having a dominant orientation towards the sink. Performing a ridge detection on this then gives the ridges shown in Figure 3b[4]. A simple growth of any ridge point that is at the end of a ridge can be used as a final step in connecting any isolated data highways, to form a connected data highway network.



(a) (b)

Fig. 3. Sensor activation/inhibition with local filters rotated towards a single sink: (a) activation field resulting from rotated filters; (b) detected ridges in field.

If we add additional data sinks then the filter orientation, and hence the activation field orientation, can be derived based on a gravitational attraction model. For each node, the activation field direction should be a weighted sum of the directions to each sink, with the weight inversely proportional to the square of the distance to the sink. In other words, the direction of the diffusion filter at node N_i can be given by the vector D_i:

[4] It is worth remarking that, whereas for the detection of single points a 2-dimensional local maxima was used, in this case the ridge detection is done by looking for local maxima in only one dimension perpendicular to the filter orientation.

$$D_i = \frac{\sum_{j \in \mathcal{S}} (N_i - S_j)|N_i - S_j|^{-2}}{\sum_{j \in \mathcal{S}} |N_i - S_j|^{-2}} \tag{2}$$

where $\mathcal{S}$ is the set of sink nodes S_i.

As with the previous example, this approach is still able to be implemented in a distributed fashion. The only aspect that unavoidably requires global knowledge is the relative location of the data sinks – or rather, whilst each node does need to know it's own location, it does need to know the direction and (network) distance to each sink. Without this knowlegdge it would be impossible to locally orient the filter, and hence activation bands and the resultant highways. Knowledge of the sinks can however be readily achieved through a broadcast beacon signal from each sink node when it activates. The beacon propagates through the network, with each node recording the number of hops to each sink from each of it's neighbour nodes, and hence the distance and dominant direction. In the next section, we will present a distributed procedure for gathering such information at each node in the network.

4 Routing on Data Highways

Let us now consider the algorithms for implementing the models discussed above, as well as how the resultant highways are then used to route data. We assume the following:

- Nodes are assigned a unique identifier;
- Nodes can tune dynamically their transmission power level P_{tx} in the range $[P_{min}, P_{max}]$;
- The network is connected when all nodes use $P_{tx} = P_{min}$;
- Nodes transmit at P_{min} unless otherwise specified.

Each node maintains the following data structures:

- A database of all data sink(s), called *sinkDB*, having entries of the type $< sinkID, distance >$, where *distance* denotes the (minimum) distance in terms of number of hops from a given sink;
- a database for the state of neighbouring nodes, where the size of the neighbourhood is determined by the filter to be applied, as detailed in Sec. 3. We denote by *nodeDB* such a database, whose entries have the form $< nodeID, distance, state, sinkDB >$, where *nodeID* is the identifier of a node in the neighbourhood, *distance* is its minimum distance (in hops) from the given node, *state* is its current state (activation level) and *sinkDB* is a data structure containing information on distance from the sinks present in the network.

We first detail the algorithms necessary for initializing the network and setting up the highways. The algorithms work in two steps. First, sinks need to announce their presence, and nodes to compute their distance from them. To accomplish such a task, sinks broadcast a beacon message (*sinkBeacon*) containing both their *sinkID* and a *distance* field in the header. The field *distance* is initialised to one. Nodes receiving the beacon check if an entry with the same *sinkID* is present in their *sinkDB*. If there is no entry, or the *distance* field in the received beacon is less than the *distance* in the stored entry, then the *sinkDB* is updated, the *distance* field is incremented by one and the beacon is forwarded on. In such a way, the field *distance* in the sinkBeacon corresponds to the minimum distance (in hops) from the corresponding sink. At the end of the process each node has the complete list of sinks in the network and the corresponding distance from any of them. Such an information is needed for setting up the filter, as outlined in Sec. 3. A detailed description of the algorithm is reported in the App. A (Alg. 1).

The next step is to let nodes construct their neighbourhood map, according to the filter parameters (in particular the dimension of the activation/inhibition neighbourhood, denoted by the filter radius R parameter[5]). In order to do so, they need to collect information about the state of their k–hop neighbours ($k \leq R$) and their distance from the sinks present in the system, as outlined in the previous section. Such a procedure is then repeated periodically to maintain an updated view of the system state. This process is carried out using a gossip–based mechanism for spreading information about nodes' state in a distributed fashion. Each node periodically broadcasts a *nodeQuery* message, where it includes its own ID, current activation state and the information contained in the sinkDB (i.e., distance from any sink). Upon reception of a *nodeQuery* message, one–hop neighbours first update the state field of the corresponding nodeDB entry. Then, they query their own nodeDB for information about the state of nodes that are at distance (in hops) less than or equal to $(R-1)$. This retrieves all information that is at most R hops from the node issuing the original query. The information collected is then included in an acknowledgment message that is sent back to the node originating the query. A detailed description of the algorithm is reported in the App. A (Alg. 2) [6]. Once each node has the relevant filter information they are able to recalculate their activation level, and subsequently compare this to neighbouring values to determine if they are a "ridge" node, and hence on a highway.

Given the mechanisms for building data highways, we now need to introduce mechanisms for routing messages containing sensed data from any node to one of the sinks present in the system. The routing protocol envisioned can be broadly divided in two phases. In the first phase, nodes not belonging to an highway have to find a way to reach one. This is done by simply broadcasting a beacon message

[5] The parameter R, which roughly correspond to the distance among highways, should be chosen in such a way to ensure that, by transmitting at P_{max}, a node will be able to join an highway.

[6] With such a procedure, information on the state of nodes is received incrementally, each round bringing information to nodes located one hop further. When bootstrapping the system, it takes R rounds to acquire the knowledge necessary for building the filter.

with increasing transmission power (and hence communication range) until a node on an highway is reached and sends back an acknowledgment message. Such a node will constitute the entry point to the highway. From that moment on, all messages generated will be forwarded to the entry point. A detailed description of the algorithm is reported in the App. A (Alg. 3). Note that if at any point there is a failure in a highway node, the source node can repeat the above process to locate a new highway entry point.

The second phase is concerned with the routing of messages along an highway, in order to reach an appropriate sink. In general, following the procedure highlighted in the previous section, we will achieve highways connected to one single sink. However, the activation–inhibition mechanism presented in Sec. 3 cannot prevent highways to be connected to two or more sinks. In order to optimize the usage of resources, messages should be directed towards the closest sink.[7] Once the highway setup phase is completed, nodes on the highway(s) will reset their sinkDB data base. Sinks will then broadcast a beacon message (*beaconSinkH*), which will be propagated only by nodes belonging to an highway. Such messages will include a *distance* field that will be initially set to 1. Upon reception of such a message, nodes on the highway(s) will check their internal sinkDB data base; if an entry corresponding to the sink ID is already present, it is updated if the distance contained in the beacon is smaller than that maintained in the corresponding entry of the data base. The beacon message is then relayed, after having increased by one the value of the *distance* field. A detailed description of the algorithm is reported in the App. A (Alg. 4).

As with reconnecting of source nodes to highways, the highways themselves are also able to self-heal. When the failure of a highway node is detected by a neighbouring highway node being unable to route traffic to the failed node, the neighbour can trigger a new localised diffusion process, that leads to the emergence of new highway in the local region of the failed node. Given the local nature of this process, this resultant highway will be operational, but may not be optimal. Whilst the highway is being used, further diffusion can progressively refine the highway route. The result is an inherent self-healing of the network.

5 Numerical Example

In order to illustrate the approach we have developed, we provide a simple example. Figure 4a shows a sample 200×200 wireless node grid. We have then allocated three sink nodes at $(25, 140)$, $(120, 180)$, and $(175, 25)$. We have assumed a maximum communication desired range of 5 units (from a non-highway node to a highway node). Consequently a distributed diffusion filter is implemented by each node, with a radius $R = 5$ hops. Other parameters used were a self-activation factor $\varphi_s = 1.5$, a constant mutual activation factor $\varphi_a = 0.7$, a

[7] It is worth remarking that the ‘distance’ in this case has to be understood as distance along the highway, and not as distance on the connectivity graph of the network as computed at system initialisation phase.

constant inhibition factor $\varphi_i = -0.3$, and an activation band with angular width $\rho = \pm 20\,\text{deg}$. This was then simulated in Matlab. Note that the simulation assumed a simple communication model that focused on routing behaviours, and assumed communication links that varied within a fixed range. Further work will need to investigate the validity of these assumptions. and the consequences when they fail.

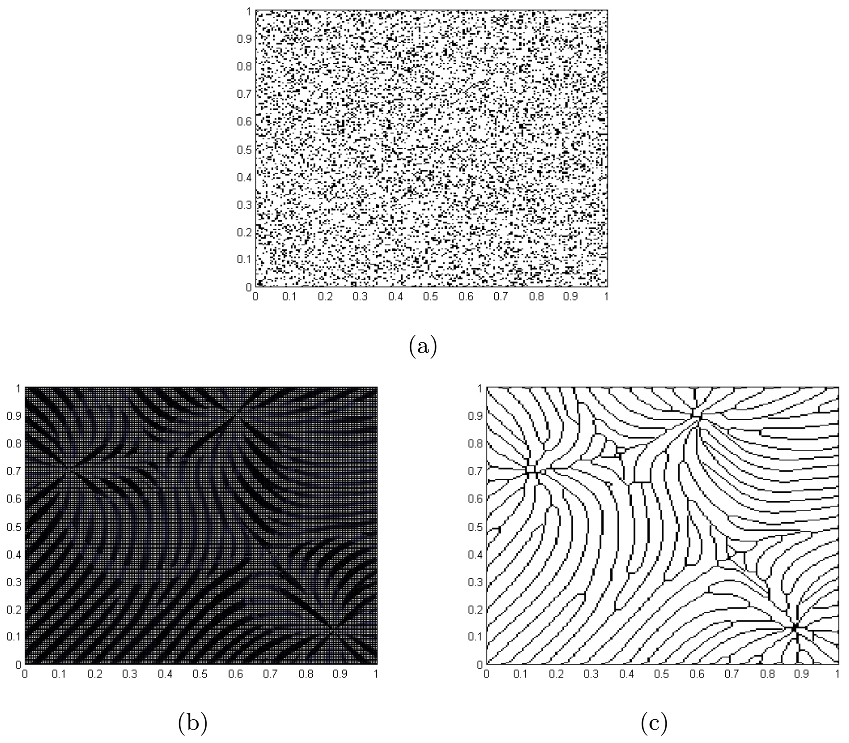


Fig. 4. Determination of wireless network data highways in a multiple sink environment: (a) Example node field (200 x 200 grid); (b) resultant activation field; (c) derived data highways from ridges in activation field.

The resultant activation field after 20 iterations of the simulated convolution is shown in Figure 4b. From this it is possible to see clearly the pattern of ridges that form, and in particular, the way in which the mechanism for calculating the directionality of the diffusion filter has led to ridges that converge on the data sinks. In numerous cases, as the ridges converge on the sinks, two or more ridges merge into a single ridge, thereby keeping the spatial separation of the ridges constant.

It is also worth noting that the resultant pattern exhibits some unusual artefacts. For example, there are several unusual bands that run orthogonally to the main ridges. We are still investigating these, though our current hypothesis is that they arise from the interactions that are occurring when different localised ridge patterns spread, and meet, during the diffusion process. These localised ridge patterns emerge early due to high local node densities. These artifacts do not appear to be problematic, though we are currently investigating this further.

From the ridge patterns we are then able to extract local ridge maxima, and then post-process these ridge maxima to ensure full connectivity is achieved. The result of this is shown in Figure 4c. As can be seen from this figure, we have a connected network of localised highways that can carry data to one or more data sinks. No non-highway node is further than 5 units from the nearest highway node, as desired.

Note that we have not yet evaluated the performance load that this approach places on the network. We are current implementing an OmNet simulation to investigate this issue.

6 Conclusions and Discussion

As outlined in section 1, the design of efficient routing algorithms is a key issue in wireless ad hoc networks. In this paper we have described an approach to the distributed design of data highways for use in routing data within dense wireless sensor networks. The algorithms developed allow these data highways to emerge naturally from localised processing in the network, without requiring network-wide knowledge or oversight, while still ensuring that design criteria are met. In particular, the highways will converge to the data sinks and ensure a maximum highway separation that allows all non-highway nodes to be within a desired maximum distance of a data highway.

As discussed in Section 4 it is expected that this approach will lead naturally to self-healing of the network – in terms of regeneration of the highways in the event of a sink failure, localised recalculation of highway routes in the event of the failure of a highway node, and reconnection of source nodes to the highways when necessary. Ongoing work will explore these self-healing characteristics.

Other questions that remain open, and represent ongoing research, relate to refining the mechanisms for determining the local diffusion mechanisms and on considering the impacts of the sink locations.

In terms of the local filter orientation, our current implementation assumes that each node is aware of it's location and that of each sink, and can hence directly calculate the orientation of the activation band in the diffusion filter. As discussed in Section 3.3 it is possible for local nodes to obtain sufficient information about sink distance and direction through a sink beacon process. We are currently exploring the implementation of a simulation based on this principle.

A further refinement that is yet to be analysed in depth involves a modification to the convolution process that removes the need for local knowledge of the sink directions and only requires distance information. Rather than applying the diffusion as a direct convolution of a fixed (albeit locally rotated) filter, it may be possible to directly diffuse the activation-inhibition data, and the receiving node determines itself whether to treat the diffusion from a neighbour as an activation or an inhibition based on distance parameters. If the distance from the receiving node to the nearest sink is similar enough to the distance from the diffusing node to the sink, then the diffusion is treated as an inhibition (since both need not be on a highway). Conversely, if the distances are sufficiently different, then it is treated as an activation, since they can be on the same highway. Further investigation will explore whether this approach is feasible.

Finally, another key avenue for further exploration is to consider the impact of sink location on the structure of the data highways. In particular, it may be possible to selectively position the sinks in order to allow the highways to be tuned, and the data loads across the highways to be optimally balanced.

References

1. Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.
2. Eitan Altman, Pierre Bernhard, and Alonso Silva. The mathematics of routing in massively dense ad-hoc networks. In David Coudert, David Simplot-Ryl, and Ivan Stojmenovic, editors, *Ad-hoc, Mobile and Wireless Networks, 7th International Conference, ADHOC-NOW 2008, Sophia-Antipolis, France, September 10-12, 2008, Proceedings*, volume 5198 of *Lecture Notes in Computer Science*, pages 122–134. Springer, 2008.
3. Yaneer Bar-Yam. *Dynamics Of Complex Systems.* Westview Press, 2003.
4. Jacob Beal and Jonathan Bachrach. Infrastructure for engineered emergence on sensor/actuator networks. *IEEE Intelligent Systems*, 21(2):10–19, 2006.
5. Nicola Bicocchi, Marco Mamei, and Franco Zambonelli. Towards self-organizing virtual macro sensors. In *Proc. of IEEE SASO*, pages 355–358, 2007.
6. Roberto Catanuto, Stavros Toumpis, and Giacomo Morabito. Opti{c, m}al: Optical/optimal routing in massively dense wireless networks. In *Proc. of IEEE INFOCOM*, pages 1010–1018, 2007.
7. Pietro Ciciriello, Luca Mottola, and Gian Pietro Picco. Efficient routing from multiple sources to multiple sinks in wireless sensor networks. In *Proc. of EWSN*, pages 34–50, 2007.
8. Andreas Deutsch and Sabine Dormann. *Cellular automaton modeling of biological pattern formation: characterization, applications, and analysis*. Birkhauser, 2005.
9. Mathilde Durvy and Patrick Thiran. Reaction-diffusion based transmission patterns for ad hoc networks. In *Proc. of IEEE INFOCOM*, pages 2195–2205, 2005.
10. Massimo Franceschetti, Olivier Dousse, David N. C. Tse, and Patrick Thiran. Closing the gap in the capacity of wireless networks via percolation theory. *IEEE Transactions on Information Theory*, 53(3):1009–1018, 2007.
11. Piyush Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.
12. Philippe Jacquet. Geometry of information propagation in massively dense ad hoc networks. In *Proc. of ACM MobiHoc*, pages 157–162, 2004.
13. Márk Jelasity. Engineering emergence through gossip. In *Proceedings of the Joint Symposium on Socially Inspired Computing, AISB Convention*, page 123126, Hatfield, UK, 2005.

14. Márk Jelasity and Özalp Babaoglu. T–Man: Gossip-based overlay topology management. In *Proc. of Engineering Self-Organising Systems (ESOA)*, pages 1–15, 2005.
15. Giovanni Neglia and Giuseppe Reina. Evaluating activator-inhibitor mechanisms for sensors coordination. In *Proc. of Bionetics*, Budapest, Hungary, 2007. ICST.
16. Fabrice Saffre and Mark Shackleton. "Embryo": an autonomic co-operative service management framework. In S. Bullock, J. Noble, R. Watson, and M. A. Bedau, editors, *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 513–520. MIT Press, Cambridge, MA, 2008.
17. Stavros Toumpis and Leandros Tassiulas. Packetostatics: deployment of massively dense sensor networks as an electrostatics problem. In *Proc. of IEEE INFOCOM*, pages 2290–2301, 2005.

A Detailed algorithms description

```
Procedure at the sink(s)
$distance \leftarrow 1$ {Set distance field to 1}
broadcast($sinkBeacon, sinkID, distance$) {Sink broadcasts beacon}


Procedure at other nodes
receive($sinkBeacon, sinkID, distance$) {Node receives a beacon}
if $sinkID$ already present as $sinkDB(k)$ then
  if $sinkDB(k).distance < distance$ then
    return {Terminate, as already have a shorter path}
  else
    $sinkDB(k).distance \leftarrow distance$ {Store new shorter path}
  end if
else
  $sinkDB.create(sinkID, distance)$ {Add new sinkDB entry}
end if
broadcast ($sinkBeacon, sinkID, distance + 1$) {Node forwards beacon}
```

Algorithm 1: Sink(s) announcement procedure.

```
Announcement procedure at each node
$nodeDist \leftarrow 1$
broadcast($nodeQuery, R, nodeID, state, sinkDB$)


Response to a query message
receive($nodeQuery, R, nodeID, state, sinkDB$)
if $nodeID$ already present as $nodeDB(k)$ then
  $nodeDB(k).state \leftarrow state$ {Update state}
else
  $nodeDB.create(nodeID, 1, state, sinkDB)$ {Add new nodeDB entry, distance field
  set to 1}
end if
create empty list $tmp$
for all entry $k$ in nodeDB do
  if $nodeDB(k).nodeID \neq nodeID$ & $nodeDB(k).nodeDist + 1 \leq R$ then
    $tmp.add(< nodeDB(k).nodeID, nodeDB(k).distance + 1, nodeDB(k).state$
    $, nodeDB(k).sinkDB >)$ {Add entry $k$ of nodeDB to the list}
  end if
end for
send($nodeQueryACK, list$) {Send response}


Update of nodeDB at each node
receive($nodeQueryACK, < nodeID_1, nodeDist_1, state_1, sinkDB_1 >, \ldots,$
$< nodeID_h, nodeDist_h, state_h, sinkDB_h >$) {Node receives an ACK message with
information on state of neighbours}
for all $i = 1$ to $h$ do
  if $nodeID_i$ already present as $nodeDB(k)$ then
    $nodeDB(k).state \leftarrow state_i$ {Update state}
    if $nodeDB(k).distance > nodeDist_i$) then
      $nodeDB(k).distance \leftarrow nodeDist_i$ {Update distance}
    end if
  else
    $nodeDB.create(nodeID_i, nodeDist_i, state_i, sinkDB_i)$ {Add new nodeDB en-
    try}
  end if
end for
```

Algorithm 2: Initialisation and update of filter at each node.

```
**At nodes not on highways**
$nextHop \leftarrow 0$
**while** $nextHop = 0$ **do**
  send($beaconNotHighway, nodeID$)
  **if** no reply within $\tau$ **then**
    increase $P_{tx}$ until $P_{max}$ is reached
  **else**
    receive($beaconNotHighwayACK, nodeHighwayID$)
    $nextHop \leftarrow nodeHighwayID$
  **end if**
**end while**


**At nodes on highways**
receive($beaconNotHighway, nodeID$)
send($beaconNotHighwayACK, ID$)
```

Algorithm 3: Routing along highways.

```
**Initialisation: at the sink node(s).**
$distance \leftarrow 1$
broadcast($sinkBeaconH, ID, distance$)


**Initialisation: at the highway nodes.**
reset sinkDB
receive($sinkBeaconH, sinkID, distance$)
**if** $sinkID$ already present as $sinkDB(k)$ **then**
  **if** $sinkDB(k).distance < distance$ **then**
    **return** {Terminate, as already have a shorter path}
  **else**
    $sinkDB(k).distance \leftarrow distance$ {Update distance}
  **end if**
**else**
  $sinkDB.create(sinkID, distance)$ {add new sinkDB entry}
**end if**
broadcast($sinkBeaconH, sinkID, distance + 1$) {Node forwards beacon to other
nodes on the highway.}


**Highway nodes announcing distance**
$myDistance \leftarrow \arg\min_{sinkDB} distance$ {Computes minimum distance from a sink.}
broadcast($beaconHighway, nodeID, myDistance$)


**Highway nodes updating next hop**
receive($beaconHighway, nodeID, distance$)
**if** $distance = myDistance - 1$ **then**
  $nextHop \leftarrow nodeID$
**end if**
```

Algorithm 4: Building routes: procedures for nodes on highways.