

Experimental Applications of Hierarchical Mapping Services in Wireless Sensor Networks

Mohammad Hammoudeh
School of Computing and IT
University of Wolverhampton
Wolverhampton, UK
Email: m.h.h@wlv.ac.uk

James Shuttleworth
Cogent Computing ARC
Coventry University
Coventry, UK
Email: j.shuttleworth@coventry.ac.uk

Robert Newman, Sarah Mount
School of Computing and IT
University of Wolverhampton
Wolverhampton, UK
Email: {r.newman, s.mount}@wlv.ac.uk

Abstract—The construction of accurate and efficient visualisations of sense data gathered from wireless sensor networks is an open problem. This paper presents a novel solution in which groups of network nodes cooperate to produce local maps which are cached and merged at a sink node, producing a low-resolution map of the global network. The sink node receives periodic map updates from each cluster-head used to refine an up-to-date global map. The global map gives a low cost interface used to target queries for generating detailed maps from a subset of the sensors in the network.

The key contribution of this work is to combine the mapping and routing services of the network, by utilising a cluster-based routing algorithm and selecting cluster heads as the caches for local maps. This technique performs favourably against similar techniques which do not exploit cluster-based routing.

I. INTRODUCTION

Ever since Descartes introduced planar coordinate systems, visual representations of data have become a widely accepted way of describing scientific phenomena. Modern advances in measurement and instrumentation have required increasingly sophisticated visual representations, to ensure that scientists can quickly and accurately interpret increasingly complex data. Most recently, wireless sensor networks (WSNs) have emerged as a technology which can provide high fidelity, multi-modal sense data over large geographic areas and long periods of time. This presents a number of challenges for developers of visualisation techniques which seek to “map” the data sensed by a network. Traditionally, such applications as mapping would be performed at a single host computer, situated at the sink of the sensor network. In this case, the sensor network is simply a data gathering tool, with all of the information processing being centralised within the computer. Increasingly WSNs are being seen as a more open and retargetable resource, possibly with multiple or mobile users and consequently multiple sinks. In this situation it is necessary for information processing to occur within the network, rather than in an external computer.

Graphics algorithms are often computationally expensive which can cause problems for low-power, long-term deployments of sensor networks. Enough data must be cached in the network to generate maps, although memory may be a scarce resource. Nodes must cooperate to interpolate between data

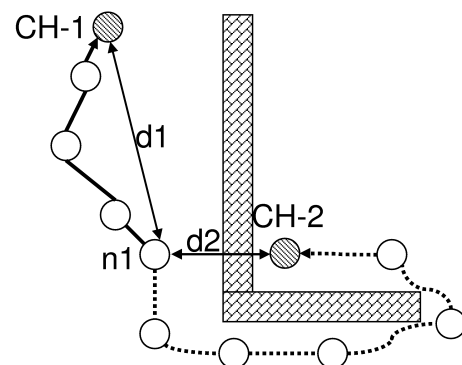


Fig. 1. Routing adaptation to obstacles using the number-of-hops. The filled nodes are cluster-heads. Horizontal and vertical bars forming “L” shape are obstacles. The solid lines are path from node n1 to CH-1 and the dashed lines are path from node n1 to CH-2.

gathered at different points in the network, but radio communication may need to be minimised for energy efficiency.

This paper presents a novel solution to these problems, in which groups of network nodes cooperate to produce local maps which are cached and merged at the sink node defined for the mapping transaction, producing a low-resolution map of the global network. The sink node receives periodic map updates from each cluster-head used to refine an up-to-date global map. The global map gives a low cost interface used to target queries for generating detailed maps from a subset of the sensors in the network. The key contribution of this work is to combine the mapping and routing services of WSNs, by utilising a cluster-based routing algorithm and selecting cluster heads as the caches for local maps.

For many mapping applications, a complete, high resolution map is not needed. Obtaining the map of an area of interest often suffices. The approach presented here allows the user to obtain a complete or partial map of the environment at desired level of fidelity. A complete map can be obtained by generating local maps at each cluster; the integration of these partial solutions solves the global problem. Indeed, the complete map is calculated by merging sub-maps produced locally at clusters.

The cluster-based technique presented here is a quasi-

distributed solution, as individual clusters of nodes behave as if they were a centralised network. This is an improvement on centralised mapping services which requires all relevant data to be transmitted to the sink node to produce a map, resulting in serious communication bottlenecks.

Hierarchical data distribution is acknowledged as an effective approach to reduce the communication in the network and load balance energy consumption [1]–[3]. The experimental work presented here uses an MuMHR hierarchical routing algorithm [4] that handles various topography types such as forests, or building interiors with obstacles of any shape, without implementation changes. This is achieved by using a number-of-hops metric. In real-world models, packets have to travel around the obstacles, consequently, leading to an increase in the number-of-hops. Figure 1 shows an example of how the number-of-hops metric helps to adapt to obstacles. Node n_1 will join CH-1 despite the fact that the distance d_1 between node n_1 and CH-2 is shorter than the distance d_2 between n_1 and CH-1.

The method proposed here is automatic and does not depend on a specific topology. Maps are successfully handled using no domain specific knowledge. This decentralised approach works very well in domains with a dynamically changing environment. The approach potentially enables the network to adapt to changes in the monitored environment locally (and independently of the other network clusters) which would make the scheme scalable. If desired, more sophisticated application-specific algorithms may be adopted as “plugins” for increased performance.

The following Section describes a number of mapping techniques for WSNs which utilise a range of network topologies, routing protocols and interpolation algorithms. Section III describes the quasi-distributed mapping service proposed by the authors and Section IV gives an experimental evaluation of the technique, carried out in simulation. Section V concludes.

II. RELATED WORK

Map construction techniques have previously been explored in the context of wireless sensor networks [1], [5]. Chang et al. implemented a fault-estimation algorithm which is based on the cluster-based framework to construct a fault map. In this work, each cluster-head uses the received data from nodes within the cluster to estimate the fault probabilities and then build up the fault map. Each cluster-head has a fault rate table to store the fault-estimation rate of each sensor node attached to it. This table is updated dynamically when a new event is detected. The fault map is constructed by calculating the fault estimation rate. The cluster-head estimates the fault rate every time it receives a new packet and compares it with a defined threshold; if the new fault rate of the node is greater than the threshold, it will be stored into the fault rate table. Therefore, the hierarchical fault mechanism via the fault map can be built, maintained, and kept up-to-date.

Event detection based on matching the contour maps of in-network data distribution has been shown effective for event detection in wireless sensor networks [5]. Xue et al. base their

event detection technique on the observation that events in sensor networks can be abstracted into spatio-temporal patterns of sensory data and pattern matching can be done efficiently through contour map matching. Contour maps were found to be an effective solution to the pattern matching problem that works for limited resources networks. Using these contour maps as building blocks, events based on the spatio-temporal patterns exhibited in the contour maps are defined. The contour map is constructed hop-by-hop from bottom up in the network as a special aggregation function instead of transmitting the data to a central point to construct the map centrally. In this map construction scheme, a rectangular $m \times n$ grid with the square cell length i is imposed on the network topology and a multi-path, ring-based routing scheme is adopted for data transmission. Each cell of the grid can have at most one node inside. The grid information is produced at the base station and is disseminated throughout the network. The data is received by and processed on every neighbour of the node that is one hop closer to the sink node. The aggregated data generated and transmitted by a node is the contour map of a sub-network rooted at the node. This data is defined as a partial map. A partial map of a node consists of a set of disjoint contour regions where each contour region is an orthogonal polygon in the 2D plane in the grid setting. A 2D polygonal contour is stored as a linked list in a partial map. Each element in the linked list is an array of vertices with the first array contains the vertices of the outer boundary of the region. The construction starts from each node generating a partial map of its own. The partial map is the grid cell of the node. When a node in the transmission line to the sink receives data from its neighbours it adds each contour region in these partial maps with its own to produce a new set called the working partial map. This process is repeated until the final partial map of the node is generated. This approach works well with grid network topologies and less well with random topologies, which may be more common in real life applications. When a grid is overlaid on top of a random topology some cells in the grid may be empty. These empty cells will not participate in the final partial map construction. Hence, the final map will not cover the entire network area. This makes the scheme sensitive and unsuitable for random sensor networks deployments. Furthermore, the loss of any partial maps will result in an incomplete network map.

In both Chang et al. and Xue et al. the sink node is required to know the location and the ID of all nodes in the network. The work in both papers is not application-independent and require major lower level modifications if the application is to change. In [1], it is not clear how the hierarchy is built. Besides, it is only suitable for small size networks due to the single hop communication scheme. In [5], the assumptions made on the network topology and the way the grid is formed are not efficient and may dissipate the energy savings achieved by the in-network map construction.

Our quasi-distributed in-network mapping service has been inspired by the work of Ganesan et al. [2]. Their work made the case for a large-scale distributed multi-resolution storage

system that provides a unified view of data handling in sensor networks incorporating long-term storage, multi-resolution data access and spatio-temporal correlations in sensor data. It constructs a multi-resolution hierarchy by each node reducing the time-series data to include only “interesting” events to reduce the communication overhead. For data transmission, Ganesan et al. [2] designed a routing protocol, wavRoute, which minimized the communication overhead and balance communication, computation and storage load. wavRoute used a recursive grid decomposition of a physical space into tiles which makes unsuitable for random topology networks.

Hellerstein et al. [1] proposed the use of TinyDB query processor to construct isobar maps in sensor networks. They showed how in-network merging of isobars could help reduce the amount of communication.

Finally, work already undertaken by the first author [6] presents an initial step towards a mapping service framework, with an example application built upon it. Although this work takes a naïve approach, more sophisticated mapping services were shown to be feasible and a need was identified for further research based on the findings in [6]. A centralized mapping service was constructed and using Shepard interpolation in the reconstruction of a parameter map from sparsely sampled data. This paper follows the work described in [6] and describes the development of a more sophisticated implementation of the mapping service and its applications.

III. QUASI-DISTRIBUTED IN-NETWORK MAP CONSTRUCTION

A. Quasi-Distributed In-network Mapping Service Architecture

The distributed mapping service is made of four modules: application, interpolation, in-network processing and routing. The Routing module is an essential module responsible for data communication. In this scheme, MuMHR [4] is the routing protocol used. The defined routing procedure builds the hierarchy and establishes the path between sensing nodes and their respective cluster-head to enable data transmissions.

The role of Data Fusion module is to process raw data received from various cluster-head nodes in the network. It applies filtering on all the received data to reduce redundancy resulting from overlapping cluster coverage. Moreover, the Data Fusion module manages incremental update messages and merges them into single transaction. Also, it defines two interfaces for the Interpolation and Application modules through which it provides access to the cached data in a suitable format.

All mapping applications use the Interpolation module as a building block to generate maps. The Interpolation module provides has access to the Data Fusion module to obtain the available mapping or update data. In this scheme, we use Shepard interpolation [7] discussed in Section III-B.

Finally, the Application module contains the user defined applications such as path-finding or isopleth maps. The application module also has direct access to the Data Fusion module

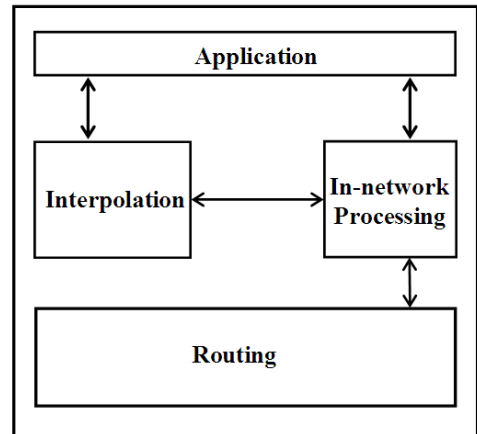


Fig. 2. Architecture of the quasi-distributed in-network mapping service.

to get raw data if required. Figure 2 show the mapping service architecture and interaction between its four modules.

B. Interpolation Algorithm for Map Construction

Map generation is essentially a problem of interpolation from sparse and irregular points [8]. There has been a great wealth of literature on interpolation methods but the challenge here is to seek a method that works for a resource-limited network in a distributed, near real-time, and energy efficient way. In addition, the method should give acceptable interpolation results given the imperfect nature of the collected sensor network data.

Inverse distance weighted interpolation methods, specifically Shepard [7] interpolation, has been found to be an attractive solution. Shepard interpolation is simple and intuitive. It is suitable for large-scale wireless sensor networks because it reduces communication overhead by only considering data points which are significant for the interpolation results. Shepard defines a continuous function where the weighted average of data is inversely proportional to the distance from the interpolated location. This method exploits the intuitive sense that things that are close to each other are more likely to be similar. The method explicitly implies that the further away a point is from the interpolation location P the less effect it will have on the interpolated value. To determine the value at an intermediate point where there no sensor readings exist, the method will use the known values surrounding the interpolated location. Tynan et al. [9] used Shepard interpolation for wireless sensor network power management. The method was also used in [6] and was acknowledged as being energy efficient.

Shepard’s expression for globally modelling a surface is:

$$f_1(P) = \begin{cases} \sum_{i=1}^N (d_i)^{-u} z_i & \text{if } d_i \neq 0 \text{ for all } D_i (u > 0) \\ z_i & \text{if } d_i = 0 \text{ for some } D_i \end{cases} \quad (1)$$

where d_i is the standard distance metric from an interpolation point P to the point numbered i in the N known points set and z_i is the known value at point i . The exponent u is used to control the smoothness of the interpolation. As the distance between interpolation location P and the measured sample point D_i increases, the weight of that sampled point will decrease exponentially. As P approaches a data point D_i , d_i tends to zero and the i^{th} terms in both the numerator and denominator exceeds all bounds while other terms remain bounded. Therefore $\lim_{P \rightarrow D_i} f_1(P) = z_i$ as desired and the function $f_1(P)$ is continuously differentiable even at the junctions of local functions. The exponent, u , is used to control the smoothness of the interpolation. High values lead to sharp edges between regions while low values lead to soft edges.

Shepard [7] devised a number of improvements to this basic algorithm to limit the effect of distant points, make use of the direction of the relationship between known points and the point to be interpolated and to incorporate information on the slope between known points. The algorithm, including the first two of these refinements, has been implemented to interpolate between sensor readings and is discussed further here.

C. Hierarchical Routing for Mapping Service

In the mapping service Routing module we deploy MuMHR [4], Multi-hop, Multi-path, Hierarchical Routing protocol. The routing hierarchy is used for many purposes including data collection, data storage, data extraction and data processing. MuMHR is an improvement over LEACH [10]. It relaxes some of the assumptions made by LEACH such as single hop communication. The main objective of MuMHR protocol is to provide substantially energy-efficient and robust communication. The energy efficiency is achieved by load balancing at two levels: (1) at the network level, which involves traffic multiplexing over multiple paths; (2) at the cluster level, introducing rotation of the cluster-heads every given interval of time. This prevents energy depletion resulting from constantly using the same path for transmission or particular nodes being cluster-heads for a long duration. The multi-paths aspect is not only used for load balancing but also when path failures occur. When a path fails, an alternative path can be immediately used which allows the protocol to dynamically adapt to failures without delays or degradation in the quality of service. At the cluster set-up time, one or more nodes are chosen as cluster-head backup node(s). Backup cluster-head node substitute for the cluster-head in some failure cases or when the current cluster-head decides to reduce its participation in the protocol if its energy level approaches a certain threshold value. For instance, if the current cluster-head decides to hand its role to the backup node, it notifies the respective node and forwards to it necessary information, such as the backup nodes list, to avoid a complete cluster set-up phase.

MuMHR reduces the energy expenditure by shortening the distance between the node and its cluster-head and by reducing the setup communication overhead. This is done through incorporating the number-of-hops metric together with the back-off waiting time. The back-off waiting time helps to

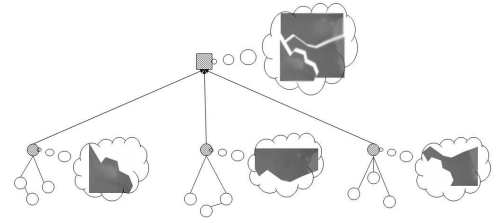


Fig. 3. The problem of generating a complete network map is solved by merging sub-maps produced and maintained locally at clusters-head nodes.

decrease the number of set-up messages and aid the formation of more geographically uniform clusters. During the back-off waiting time, sensor nodes receive advertisement messages and only consider the message with the smallest number-of-hops received during that time. This allows blocking of the advertisement flooding at the edges of neighbouring clusters. Both the number-of-hops metric and the back-off waiting time allow energy efficient cluster formation.

The operation of the proposed routing protocol can be split into two phases: the setup phase and the data transfer phase. During the set-up phase cluster-heads are selected and hierarchy is created. For every node to communicate with its cluster-head it only needs to know the next hop neighbour nearest to the cluster-head. During the data transmission phase, sensing nodes transmit data to their cluster-head. The cluster-heads aggregates the received data before transmission to the sink or immediately multiplex messages over multiple lines in time critical applications. Each member node transmits data on its assigned time slot scheduled by TDMA.

D. Algorithm of the Quasi-Distributed In-network Map Construction

The highly data-driven nature of sensor networks and their limited resources impose many requirements on data storage and processing infrastructure. A fully centralized data collection, storage, and extraction are infeasible due to many reasons including: high energy cost involved in transmitting the data to the sink; it introduces a single point of failure; performance bottlenecks especially around the sink; sensor data has significant redundancy; a big portion of the returned data is not useful; among others. However, many queries of the sensor network of a spatio-temporal nature that requires centralised data collection. Mining for characteristics of such nature where local processing is not enough should be feasible. Furthermore, a sensor network user will regularly require an abstract global view of the large sensor data. The system proposed here responds to these two requirements and integrates them into a semi-distributed, hierarchical, in-network mapping service. It includes a hierarchical, multi-resolution storage and a distributed processing to data extraction paradigms. Scalable, low-resolution, and load-balanced data extraction is sought as a solution to fulfil these requirements. The sink generates a low cost, low-resolution map for the network region before deciding to get more detailed and more expensive detailed datasets. At a lower hierarchical level, cluster-head nodes

maintain high-resolution maps from larger number of nodes to obtain an accurate image from a smaller region. Figure 3 illustrates how local cluster-head nodes maintain a high-resolution accurate map of the regions they manage; these maps are merged at the sink to generate a complete map of the network that contains lower-level of details from larger region.

The major steps of the distributed mapping service are as follows:

- 1) Cluster-head node collects data from its vicinity.
- 2) Cluster-head node computes its partial map and forwards it to the sink.
- 3) The sink computes complete network map.
- 4) Dynamically update the partial and complete maps.

1) *Cluster-head node collects data from its vicinity:* After the paths to the cluster-head are established and every node receives a TDMA schedule, sensing nodes start data transmission to their respective cluster-head in their assigned time slot. The data is sent to the cluster-head is in the form of tuple (x, y, v) where x and y are the 2D Cartesian coordinates of the sensing node and v is the value of the measured parameter. The cluster-heads continuously record and aggregate the each node data. To avoid sending large amount of data across the network, various cluster-head nodes will collect the data from their area and create summary dataset as defined by the data fusion model in the mapping service. The size of the summary dataset is determined with the accuracy level required by the complete network map. These summary messages are then transmitted to the sink.

2) *Cluster-head node computes its partial map and forwards it to the sink:* The cluster-head uses the received data from nodes within the cluster to build an accurate map of its vicinity. To reduce the size of the data set, the cluster-head applies compression techniques to eliminate redundancy due to overlap of nodes coverage area. Given an up-to-date point's data set, the cluster-head can easily generate a partial map using the interpolation module included in the mapping service. This compression process is important as it makes the interpolation step faster. The cluster-head caches two data representations: point's data and graphical map. The former is used to build the local graphical map and forwarded to the sink to be combined with data received from other clusters to build the complete network map. While the latter is used to respond quickly to user queries about events or parameters to be found within the cluster area.

3) *The sink computes complete network map:* The sink fuses the partial map data received from all cluster-head nodes into a complete map of the network. By applying the same interpolation step performed at the local cluster-head nodes, the sink generates a complete map of the network. When requested by the user, this map can be refined by querying more detailed map from the cluster-head managing the area of interest. A user may want to query detailed datasets from a group of sensor in the network. A query may be directed to the cluster-head node nearest to the desired location.

4) *Dynamically update the partial and complete maps:*

In large scale sensor network, map datasets could be large in size. An incremental update of partial and complete maps is a lot if compared to the bandwidth of a typical wireless communication channel. Map dataset size reduction is therefore of vital importance. One method for map dataset size reduction is to split a map in smaller pieces, which can be updated separately. In many real-world applications of WSNs, the sensed modalities are mostly unchanged or only slightly changed over time [5]. Therefore, the successive maps are generated for these modalities are very similar and incremental updates of the maps could save a considerable amount of energy. To reduce the size of an update, only the changes with respect to a previous map version are transmitted. This way you get incremental updates. It is important to have the previous map as a starting point updates to make sure all updates are applied, in the correct order. If several update operations need to be processed all before the map is consistent again, they should be grouped into one update transaction. The users will need an up-to-date map within a certain time after they request for it and within a certain time after the cluster-head nodes respond with new information. The delays experienced by the applications will very much depend on the characteristics of the communication channel.

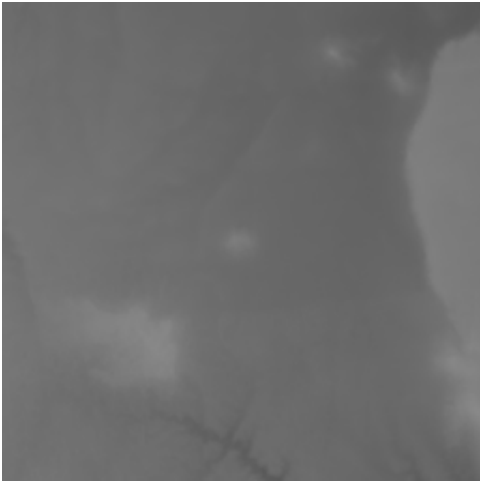
The hierarchical organisation of the data storage, extraction, and processing scheme can be exploited to search efficiently for patterns across the network. This can be done by first extracting and examining a low-resolution map at the sink. Then this map can be used to obtain higher accurate maps for sub-regions of the network efficiently by drilling down the routing hierarchy by eliminating clusters whose maps are not significant to answering a query. In dynamically changing environments, when local changes occur on topological elements, such as fire causing a wall to collapse, the map produced from the modified cluster is recomputed locally and the rest of the map is left unchanged.

The routing hierarchy can have any number of levels, making it scalable for large problem spaces. When the problem space is large, a larger number of levels can be the solution for reducing the map production effort, for the cost of more localized storage and pre-processing time.

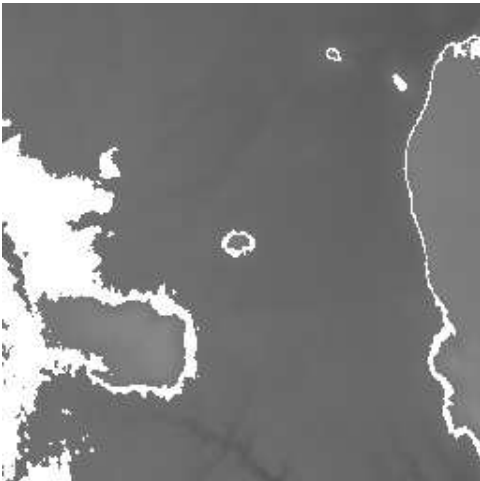
IV. EXPERIMENTAL COMPARISON OF CENTRALISED AND HIERARCHICAL MAPPING SERVICES

In this Section we study the efficiency of the proposed hierarchical mapping service in terms of energy and the quality of the produced map. The results obtained here are compared to the results of the approach proposed in centralised mapping service [6]. This algorithm has been implemented using a mapping API with an in-house simulation software "Dingo" which is an enhanced version of "SenSor" simulator [11], [12] which used to implement the centralised mapping service.

To present the results of generating high quality maps at a low energy cost and minimal network delay we have chosen the determination of isopleths on interpolated fields. Figure 4(f) shows a height map of a section of the Grand Canyon,



(a) Height map derived from USGS Grand Canyon data.



(b) A contour at 116 units drawn on the interpreted field in Figure 4(f).



(c) Interpolated field generated using a hierarchical mapping service running on 150 simulated nodes, randomly distributed on the surface from Figure .



(d) Interpolated field generated using a hierarchical mapping service running on 150 simulated nodes, randomly distributed on the surface from Figure .



(e) A contour at 116 units drawn on the interpolated field from Figure .



(f) A contour at 116 units drawn on the interpolated field from Figure .

Fig. 4. Simulation results

taken from data recorded by the US Geological Survey [13]. We apply the isopleths generation algorithm directly to the

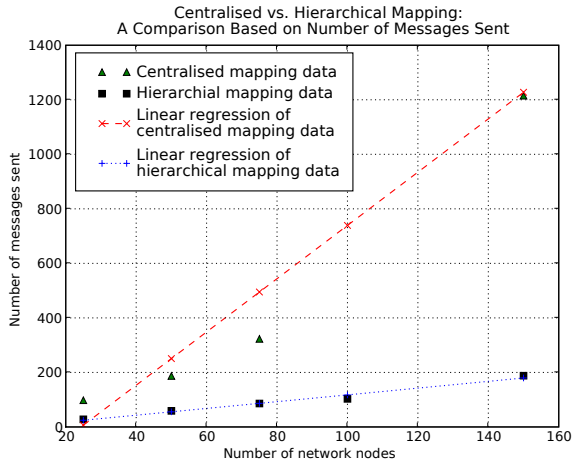


Fig. 5. A comparison of centralised and hierarchical mapping based on number of messages sent

data collected by the centralised and hierarchical mapping services. In this experiment, a 150-node sensor network was randomly distributed over the map described above. The results of generating isopleths based on this data are shown in Figures 4(e) and 4(f). These contours were generated for a height of 116 units and a threshold of 1. The interpolated terrain generated by the two approaches is nearly identical and is similar to the real surface. However, the difference in the cost of generating the two contours is large. Figure 5 shows the difference between the number of messages exchanged using both approaches.

The comparison of the data shown in in Figure 5 captures the essence of our approach. The figure plots the mapping traffic against the number of nodes in the network. Remarkably, the number of exchanged mapping messages in the hierarchical mapping service is much smaller than that of the centralised version. This suggests that exploiting the routing hierarchy results with substantial energy savings without any degradation in the quality of the produced map. Unlike the centralised mapping service, as the number of nodes in the network increases, the mapping traffic in the hierarchical service remained small which makes it a scalable solution for mapping large-scale WSN data. The hierarchical scheme scales fairly well with respect to the network size and achieves these gains in the efficiency (energy and timeliness) without requiring any extra setup messaging.

V. CONCLUSION

Visualisation of sense data gathered from networks of wireless sensors is a challenging problem in several regards. This paper has presented a potential solution which addresses several issues, namely: energy efficiency, data storage and network organisation. The approach is novel in that partial visualisations are computed and merged in local network clusters. This combination of routing and visualisation is the major contribution of this work. Results of this work, gained in simulation, indicate that significant savings in energy (from

radio transmission) can be made using this quasi-distributed approach.

REFERENCES

- [1] Y.-S. Chang, C.-J. Lo, M.-T. Hsu, and J.-H. Huang, "Fault estimation and fault map construction on cluster-based wireless sensor network," in *SUTC '06: Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing - Vol 2 - Workshops*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 14–19.
- [2] D. Ganesan, D. Estrin, and J. Heidemann, "Dimensions: why do we need a new data handling architecture for sensor networks?" *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 143–148, 2003.
- [3] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2006, pp. 278–287.
- [4] M. Hammoudeh, A. Kurtz, and E. Gaura, "MuMHR: multi-path, multi-hop, hierarchical routing," in *In Proceedings of the 2007 International Conference on Sensor Technologies and Applications (SENSORCOMM2007)*, 2007.
- [5] W. Xue, Q. Luo, L. Chen, and Y. Liu, "Contour map matching for event detection in sensor networks," in *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM Press, 2006, pp. 145–156.
- [6] J. Shuttleworth, M. Hammoudeh, E. Gaura, and R. Newman, "Experimental applications of mapping services in wireless sensor networks," in *In Proceedings of the Fourth International Conference on Networked Sensing Systems INSS 2007*, Braunschweig, Germany, 6-8 June 2007.
- [7] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM national conference*. ACM Press, 1968, pp. 517–524.
- [8] J. Shuttleworth, E. Gaura, and R. M. Newman, "Surface reconstruction: Hardware requirements of a som implementation," in *Proceedings of the ACM Workshop on Real-World Wireless Sensor Networks (REAL-WSN'06)*, June 2006.
- [9] R. Tynan, G. O'Hare, D. Marsh, and D. O'Kane, "Interpolation for wireless sensor network power management," in *International Workshop on Wireless and Sensor Networks (WSNET-05)*. IEEE Press, June 2005.
- [10] W. Heinzelman, A. Chandrakasan, and H. Balakrishna, "Energy-efficient communication protocol for wireless microsensor networks," *Proceedings of the 33rd International Conference on System Sciences*, January 2000.
- [11] S. Mount, R. Newman, E. Gaura, and J. Kemp, "Sensor: an algorithmic simulator for wireless sensor networks," in *In Proceedings of Euroensors 20*, vol. II, Gothenburg, Sweden, 2006, pp. 400–411.
- [12] S. Mount, R. Newman, and E. Gaura, "A simulation tool for system services in ad-hoc wireless sensor networks," in *In Proceedings of NSTI Nanotechnology Conference and Trade Show (Nanotech'05)*, vol. 3, no. 7, Anaheim, California, USA, May 2005, pp. 423–426.
- [13] USGS and C. McCabe, "Grand canyon terrain," Georgia Institute of Technology Large Geometric Models Archive, 1998, <http://www.wstatic.cc.gatech.edu/projects/largemodels/>.