Increasing the Lifetime of Roadside Sensor Networks using Edge–Betweenness Clustering

Joakim Flathagen*‡, Ovidiu Valentin Drugan§, Paal E. Engelstad§ and Øivind Kure‡
*Norwegian Defence Research Establishment (FFI), ‡Q2S NTNU, §University of Oslo
Email: joakim.flathagen@ffi.no, ovidiu@ifi.uio.no, {paalee, okure}@unik.no

***Abstract*—Wireless Sensor Networks are proven highly successful in many areas, including military and security monitoring. In this paper, we propose a method to use the edge–betweenness community detection algorithm to determine clusters and to facilitate in-network data aggregation for these applications. To minimize the cost of determining the clusters, the approach is based on exploiting the topology information from the ad hoc routing protocol. Three different schemes are proposed. (1) A distributed clustering scheme using the OLSR routing protocol. (2) A centralized scheme using OLSR. (3) A centralized scheme using an extension to the DYMO-low routing protocol. All schemes support sensor heterogeneity allowing that different data content can use different routing paths. The paper presents simulation results and an analysis of the cluster generation for each of the schemes. The results show that our method is a simple and effective method to improve scalability and lifetime of roadside sensor networks.**

***Index Terms*—Clustering, Data Aggregation, Wireless Sensor Networks**

I. Introduction

Wireless Sensor Networks (WSNs) are proven effective in the fields of perimeter security and military surveillance [1]. In these areas, great benefit can be achieved by using covert miniaturized sensors, as they are difficult to avoid by a possible intruder and less subject to vandalism or theft compared to traditional sensor systems. Further, the redundancy given by ad hoc network protocols improves reliability compared to previous systems. However, WSNs face two basic challenges; *energy efficiency*, due to the battery powered sensors, and *scalability*, due to a potential high number of devices needing to interoperate. The goal of this paper is to provide a method to solve these two issues by the means of in-network data aggregation.

Data aggregation is particularly interesting for roadside surveillance systems. In such systems, the sensor nodes collaborate in detecting events such as movement and particular behavior of objects along the road. Multiple nodes are here likely to sense the same event simultaneously. Conventional routing treats these sensor readings individually and ignores the redundant and highly correlated nature of the data. This leads to ineffective use of the scarce energy and limited channel resources. By employing data aggregation, designated aggregation nodes can wait for multiple reports, either from the same node (temporal redundancy), or from neighboring nodes (spatial redundancy), before reporting about the event to the sink. This strategy not only reduces the traffic considerably, but also reduces the probability of false alarms, as most sensors are likely to be inaccurate and have a small probability of falsely reporting events that are not actually present.

The contributions of this paper include: (1) A data aggregation scheme based on edge–betweenness community detection, (2) three different routing protocol schemes supporting both centralized and distributed clustering, (3) modification and improvement of the DYMO-low routing protocol, and (4), a quantification of the trade-off between cluster-aggregation and traditional routing, and a comparison of our schemes with the well-known K-means clustering. Although we mainly focus on roadside surveillance networks, our protocols, recommendations and results are also viable to other classes of sensor networks that are topologically similar to our scenario.

Before presenting our own scheme and results, it is worth reviewing some of the preceding work regarding data aggregation in WSN.

II. Related work

Different data-aggregation alternatives can be categorized based on the network architecture involved in the aggregation, which can be structured either as a chain, a tree, or by clusters.

Chain-based aggregation schemes create linear chains for data-aggregation. Each node in the chain only transmits to its closest neighbor, which fuses the data with its own measurements, and retransmits along the chain. In PEGASIS [2], the chains can be made either centrally or distributed. *Tree-based* data aggregation on the other hand, organizes the nodes in an aggregation tree rooted at the sink. Directed Diffusion [3] is one such example. If only a subset of the nodes in the network are sensing nodes, tree based techniques provide better performance than chain-based since the aggregation tree is better than the chain for mere packet routing. For both strategies, the aggregation delay perceived by a node is based on its position in the aggregation tree (or chain). The overall aggregation delay therefore increases drastically with the number of nodes in the network [4]. The challenge is to balance the trade-off between energy efficiency and the delay posed by the aggregation. Both tree-based and chain-based aggregations are best suited for scenarios where *all* nodes in the network produce relevant information *periodically*. For our event-initiated scenario, these proposals are inadequate since the long aggregation delay makes it difficult to uniquely distinguish separate events.

Cluster-based schemes organize the sensor nodes into virtual groups and perform aggregation only at designated cluster-

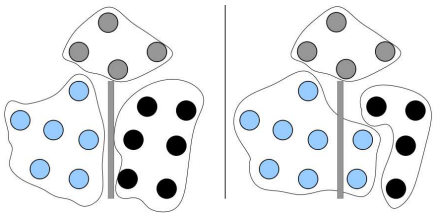

Fig. 1. Edge–betweenness clustering (left) takes the topology into account while K-means (right) use mere geographical positions for clustering.

heads (CHs). This approach drastically reduces the aggregation delay compared to the chain and tree architectures, at the cost of possibly longer routing paths. Notice that cluster schemes are not limited to aggregation only. LEACH [5] for example, uses clustering both as a tool to aid data aggregation and to coordinate access of the wireless channel within the cluster. LEACH only supports single-hop transmission between each cluster-head and the sink, making the approach invalid for our purpose. Lai et al. have recently extended LEACH by allowing multihop transmissions and by better balancing the energy consumption [6]. Gong et al. [7] takes a different approach and propose to use modified K-means clustering, and determines the clusters centrally assuming that the geographical positions of the nodes are known. We describe this method and compare it to ours in the subsequent sections.

While [5]–[7] use explicit control messages to initiate the clusters, our scheme has the ability to passively exploit the underlying routing protocol to gain topology knowledge. Another key difference is that the above methods require that all traffic must pass through the cluster-head, while our approach allows some traffic classes to take an optional (shortest-path) route towards the sink.

III. Clustering

A. K-means

K-means is a classical and simple method for clustering that has been applied to several problem domains, including sensor networks, as demonstrated by Gong et al. [7].

When applied to sensor node clustering, the procedure is as follows: (1) the number of clusters k must be predetermined. (2) k points are placed in the geographical space represented by the nodes being clustered. These points represent the cluster centroids. (3) Each node is assigned to the cluster with the closest centroid (in terms of Euclidian distance). (4) The positions of the k centroids are recalculated as the mass center of each cluster. Then, (3-4) are repeated until the centroids no longer move. In [7], the nodes with the minimum distance to the cluster centroid and highest residual energy are elected as cluster-heads.

While this algorithm outperforms LEACH, its disadvantage is that the number of clusters must be predetermined (or estimated), and that the exact geographical position of the nodes must be known.

B. Edge–betweenness community detection

Edge-betweenness community detection is a method proposed by Newman and Girvan [8]. Community detection algorithms are known from physics literature, (i.e., a community is a region of the network with dense connections) and have been successfully used to capture interactions in ad hoc networks [9]. The algorithm tries to find the communities of the network with the maximum modularity value. The modularity measure is based on the formula $\mathcal{Q} = \sum_{i=0}^{m} \left(e_{ii} - a_i^2\right)$ where m is the number of detected communities, e_{ii} represents the fraction of links in the network that connect the nodes in community i, and a_i represents the fraction of links that connect two nodes in community i. The algorithms proposed by Newman and Girvan [8] all find good approximations for the maximum modularity. The algorithm (*EB*) searches for the division of the network with the greatest modularity value by removing links with high importance in the network (see Algorithm 1).

Algorithm 1 Edge-betweenness

EB $(\mathcal{G})$

1) $\mathcal{G}' = \mathcal{G}$
2) $for\ i = 1 \ldots |\mathcal{L}|$
   a) $\mathcal{G}' = \{\mathcal{G}' \setminus \{e_{b_i}\} \mid b_i = max\,(betweenness\,(\mathcal{G}'))\}$
      $\mathcal{P}_i = \{connected\,(\mathcal{G}')\}$
   b) $\mathcal{Q}_i = \mathcal{Q}\,(\mathcal{P}_i)$
3) **return** $\{\mathcal{P}_l \mid l = max\,(\mathcal{Q}_i)\}$

The algorithm recursively computes the betweenness score of each link in $\mathcal{L}$ defined by the number of shortest paths going through a link. The link with the highest betweenness score is removed from the graph, and the modularity value is recomputed. The algorithm is applied until there are no more links left. The communities are determined by the partitioned network obtained in the step with the maximum modularity value.

As opposed to most existing clustering methods, EB-clustering does not put any a priori constraints on the cluster structures (e.g., cluster diameter, number of nodes in a cluster or number of clusters). While K-means requires that a localization scheme is present in the network, EB-clustering only relies on the network topology. Notice that K-means assumes that geographically adjacent nodes also are 1-hop neighbors. This is not always the case for sensor networks. As shown in Fig. 1, this assumption can lead to suboptimal clusters and excessive paths between cluster members and the cluster-head.

C. Fetching topology information

The prerequisite for EB-clustering is to have an updated view of the network topology. Such information can either be obtained *actively* by exchanging explicit control messages between the nodes, or *passively* by taking advantage of information available by consulting the underlying routing protocol. Our approach belongs to the latter category, and performs the topology fetching without the need for extra

messages. Consequently, the overhead of enabling clustering in the network can be drastically reduced.

In our approach, the cluster construction is separated from the routing layer, and standard routing is therefore maintained. The approach taken by [5]–[7] on the other hand, forces all traffic to be routed via the cluster-heads, which is not always in the shortest path between an arbitrary node and the sink. This is a suboptimal solution for heterogeneous networks containing several sensor types. In surveillance systems for example, all sensor nodes can contain passive IR, sound and vibration sensors to detect and track a target, while a few nodes are equipped with a digital camera or active IR for target verification. Our approach supports such applications using policy-based routing. Alarms and measurements are considered easy to aggregate (homogeneous data) and can be routed directly to the designated cluster-head, which is responsible for data aggregation (to reduce data transmissions) or filtering (to reduce the false alarm rate). Meanwhile, data from special purpose sensors, such as imaging sensors, can not be aggregated and should therefore follow the shortest path to the sink.

In this paper, we study both centralized and distributed clustering methods and examine the use of two different routing protocols to obtain the topology information. The proposed schemes are:

1) OLSR distributed scheme.
2) OLSR centralized scheme.
3) DYMO-low centralized scheme.

In the next two sections we describe how to combine EB-clustering with these routing alternatives.

IV. OLSR Schemes

A. Introduction

Optimized Link State Routing (OLSR) [10] is proposed by the IETF aiming at Mobile Ad-hoc Networks (MANET). Although OLSR is seldom considered viable for sensor networks due to its proactive behavior and the possibly large routing table, we argue that some classes of WSNs may benefit from the use of OLSR. OLSR has gained considerable popularity because of its versatility and extensibility, and simple extensions can provide several attractive features, such as e.g., multicast, multiple interfaces and service discovery. If such features are needed in the WSN, using OLSR may simplify the design compared to adding these features on top of a less advanced protocol.

For our purpose, OLSR provides the attractive feature that each node keeps an updated view of the network topology. This feature can be used to determine clusters in the network in a distributed fashion, as described in the next section.

B. OLSR distributed scheme

The *OLSR distributed scheme* employs the OLSR routing protocol repositories on each of the nodes to gain information about the network topology. This information is then used to determine the network clusters locally using EB-clustering. The challenge with this approach is that it relies on consistent cluster calculation in the network. To ensure that all the nodes determine exactly the same clusters, each node needs to obtain accurate topology information. However, if default OLSR settings are used, only partial link-state can be obtained.

The partial link-state in OLSR is caused by the intention to limit the communication overhead by reducing the number of links advertised and the number of nodes that advertises them. Using default OLSR settings, only nodes chosen as MultiPoint Relays (MPRs) create topology control (TC) messages. A TC message only contains the advertised link set of a node limited to its MPR selector set. Hence, all neighbors will not be reported in the TC message, and for our purpose, this means that the entire topology (including all links) cannot be detected. Consequently, exact and consistent cluster determination cannot be ensured.

Mechanisms to extend the network topology knowledge in OLSR are previously studied in [11] and [12]. In [11], the authors investigate different options by tuning the MPR-Coverage settings and by increasing the amount of information in each TC message. One way to let an MPR report all links is to alter the `TC_REDUNDANCY` parameter from TC_0 to TC_2. By doing this, the advertised link set of the node include the full neighbor link set. However, as pointed out in [12], the nodes generating TC messages are not constrained to MPRs only when using this setting. The authors therefore suggest applying TC generation with full link set only to those nodes that are selected as an MPR by another node. This new proposed setting is named TC_4 (this term is also applied in our research). Notice that if a link exists between two non-MPR nodes, its existence is not reported in any TC messages. This can be resolved by changing the `MPR-coverage` setting, as proposed in [11]. By altering this parameter a node can increase the preferred number of MPRs in its MPR set increasing the probability that all links are reported.

To verify the performance of the distributed clustering scheme, we examine the consistency of the identified clusters while altering the `TC_REDUNDANCY` parameter.

C. OLSR centralized clustering scheme

The *centralized* clustering scheme solves the beforementioned cluster consistency issue. In this case, the clusters are determined by employing EB-clustering at the sink node only. The partial link-state of OLSR is not critical since each node will unambiguously belong to one single cluster. The drawback of this approach is that a separate protocol is needed to elect and inform cluster-heads and to tell each node which cluster it belongs to. However, this can be solved either by creating an OLSR extension, or using a simple application layer protocol.

It is worth noting that distributed protocol designs are traditionally preferred before centralized designs in networking systems due to the fault tolerance and lack of scalability of the latter approach. We argue that in most WSNs, the sink node is already a single point of failure. The fault tolerance is therefore not increased by centralizing the clustering algorithm [13]; in fact, this approach simplifies the protocol design and

the implementation. Further, the scalability of the centralized algorithm is not a big concern compared to a distributed design, as the sink node can be equipped with several orders of magnitude more memory and CPU than the sensor nodes.

V. DYMO-low Scheme

In the *DYMO-low centralized scheme*, we propose a few modifications to the DYMO-low routing protocol [14] to fetch topology information and to facilitate centralized clustering. DYMO-low is intended for use on IEEE 802.15.4 devices and is based on the principle of flooding route requests (RREQ) and unicasting route replies (RREP) as known from AODV and DYMO. However, DYMO-low is considerably simplified to better match the limitations of 802.15.4.

Our extension introduces two new messages to the protocol, Topology Route Request (TRREQ) and Topology Route Reply (TRREP). As with the centralized OLSR scheme, the sink node performs the EB-clustering calculation, and a separate protocol is employed to elect and inform the cluster-heads. The difference between this approach and the OLSR centralized approach is that DYMO-low is a reactive protocol and does not disseminate routing information regularly as is the case with OLSR. This can save considerable bandwidth if the proactive behavior of OLSR is not needed. Another benefit with this approach compared to OLSR is that our DYMO-low implementation can provide *full* topology information, whereas OLSR does not provide this without implementing the beforementioned extensions, with the penalty of increased OLSR overhead. The protocol operation consists of a request phase and a reply phase.

The sink first initiates the network by announcing its address via a TRREQ. This message can be seen as a proactive route request destined to *all* nodes in the network. The TRREQ is flooded similarly as a regular DYMO-low routing request (RREQ), and the nodes which receive the TRREQ, retransmits the packet only once. This means that all nodes will receive a copy of the TRREQ from each of its neighbors. When a node receives a TRREQ packet, it stores the address of its neighbor. As the TRREQ disseminates from the sink to the entire network, all nodes will eventually obtain a list including each of its one-hop neighbors with no more cost than a regular Route Request process.

Upon receiving a TRREQ packet, a node responds back to the sink using a TRREP. This transmission takes place after a random time delay to avoid network congestion and collisions. The response message extends the regular Route Reply defined in DYMO-low, with a list of the one-hop neighbors. The sink will eventually receive TRREPs from all the sensor nodes in the network. EB-clustering will then use this information to determine the clusters. Note that each link in the network is reported twice (once from each link end). The duplicated information can enable reconstruction of missing TRREP information.

It can be argued that this approach cannot be classified as passive clustering since we alter the routing protocol to fetch the topology. However, our extension in fact reduces the number of control messages compared to standard DYMO-low. After a complete request/reply phase, all nodes in the network have a valid route to the sink, making them ready to perform their sensing task immediately. If standard DYMO-low is used, route requests must be initiated from each node in the network to accomplish this. This leads to a tremendous overhead due to the flooded route requests. Our approach, on the contrary, limits this to just *one* sink-initiated route request and considerably reduces the number of messages flooded in the network. The request/reply and EB-clustering process can be initiated either automatically or by a network operator.

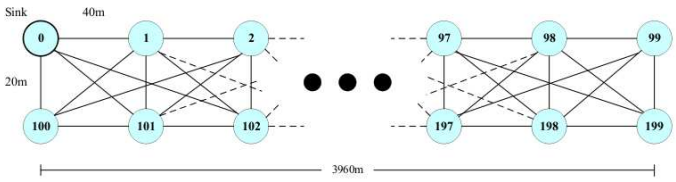

Fig. 2. The main scenario used in the simulation and analysis.

VI. Results and analysis

The motivation behind the simulations in this section is threefold. First, we analyze the accuracy of the topology knowledge in OLSR and how inaccuracies affect the cluster consistency for distributed clustering. Second, we compare the overhead posed by the different schemes. Finally, we study the energy savings by employing the clustering scheme considering different cluster-head election strategies, and different distances between a target (sensed by the WSN) and the sink. We compare EB-clustering with K-means clustering.

We implemented the DYMO-low Internet Draft in the NS-2.34 network simulator and added the proposed extensions to the protocol to enable neighbor detection and reporting. For the OLSR experiments, we used UM-OLSR [15], which we modified to provide extended topology knowledge. The clustering methods were implemented using iGraph. Unless otherwise mentioned, default OLSR settings were used. IEEE 802.11 DCF was used as the MAC protocol.

Two scenarios were created for the testing and analysis. The initial setup (scenario 1) consists of 200 nodes aligned along a virtual road, see Fig. 2. The inter-node distance is 40m horizontally and 20m vertically, covering a total area of 20x3960m. The transmission range is set to 100m. Scenario 2 is a small modification of Scenario 1 made by removing the nodes $4, 8, 12 \ldots 96$. Scenario 2 in this way provides a layout with defined groups of nodes.

A. Topology knowledge and cluster consistency

First we consider only Scenario 1 and employ OLSR routing. Fig. 3 shows the average accuracy of the topology knowledge at each of the nodes. When using default OLSR, topology knowledge accuracy was only 35%. Using TC_2, the accuracy increased to 90%, while with TC_4, the accuracy was 75%. These results correspond to those presented in [12]. The reduced topology knowledge observed at the network ends

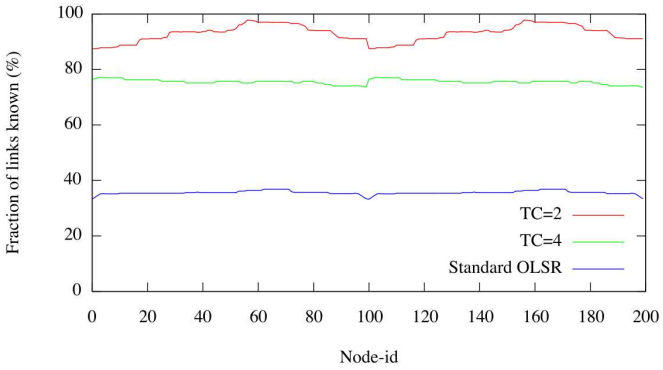

Fig. 3. Accuracy of topology knowledge in the network

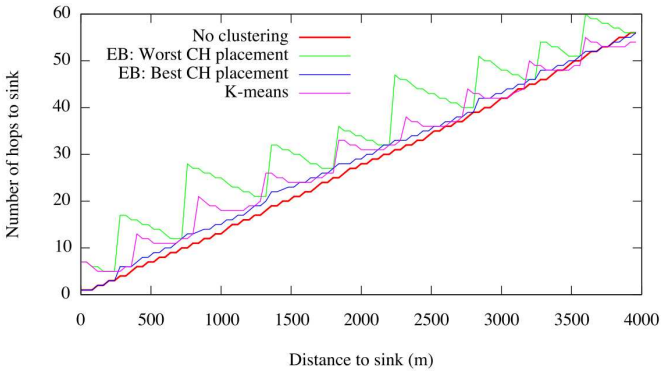

Fig. 4. Number of hops as a function of distance between a detected event and the sink.

TABLE I
PERFORMANCE OF THE SCHEMES. DIFFERENT OLSR DISTRIBUTED CLUSTERING COMPARED WITH DYMO-LOW CENTRALIZED CLUSTERING.

Protocol	Top-know	Overhead	Clust-cons1	Clust-cons2
OLSR TC_0	35%	39.5 KB/s	88.9%	91.1%
OLSR TC_2	90%	76.7 KB/s	96.3%	96.0%
OLSR TC_4	75%	58.1 KB/s	90.6%	93.4%
DYMO_low	100%	483KB/round	(100%)	(100%)

(i.e., the nodes 0,99,100,199) is caused by collisions (and loss of topology information) in the center of the network. As a comparison, when running the centralized DYMO-low scheme 100% accuracy is achieved in the same scenario.

We now examine how the topology inaccuracies affect the consistency of the clusters when EB-clustering is employed on each node. To compare the communities detected at the different nodes, we represent the node-to-cluster memberships in matrixes, and compare the matrixes created at each of the nodes. Due to different topology knowledge, a small percentage of the detected cluster memberships differ among the nodes. The values in table I show the percentage of the detected cluster membership information that is equal among all nodes. There is a tendency that local information is correct, and the membership inconsistencies are on distant clusters only. EB-clustering here works remarkably well even with limited OLSR topology knowledge, but increasing the topology knowledge further improves the clustering consistency. An input scenario with more clearly defined groups (Scenario 2) also leads to more consistent clusters. This is caused by the fact that EB-clustering creates communities based on counting the number of shortest paths going through each link (betweenness-score), and this scenario has more links with salient betweenness-score.

When clustering is employed *centrally*, the reduced topology accuracy is not crucial. Even with standard OLSR, our experiments show that the clusters fit the physical layout of the nodes well, although not as accurate as with increased OLSR topology knowledge or by using DYMO-low.

B. Overhead

Table I also show the overhead for the different routing alternatives. As the OLSR protocol exchanges control messages periodically, overhead is almost constant regardless of the lifetime of the clusters and the rotation of the cluster-heads. For DYMO-low, messages are only sent when the clusters are regenerated (referred to as one *round*[1]). For DYMO-low, all TRREQ and TRREP transmissions for one round are included in the measures. We also let CHs flood their existence to the network using TRREQs. The results show that the centralized DYMO-low protocol leads to the best clusters (full topology is obtained) and also the smallest overhead in scenarios with slow CH rotation ($> 12s$). We assume that the distributed clustering scheme may be efficient in mobile networks, which require frequent regeneration of the clusters and where network partitions prohibit centralized control.

C. Data aggregation

Next, we evaluate the clustering scheme considering different cluster-head election strategies, and different distances between a target (sensed by the WSN) and the sink. We extend Scenario 1 to include a vehicle that moves along the WSN, and apply DYMO-low centrally. Sensor nodes detecting the vehicle transmit this information to the cluster-head in its cluster. Most clustering schemes in the literature employ a round-robin scheme to alternate the role of the CH to balance the energy consumption. We find it interesting to examine the performance of EB-clustering with extreme CH placements. The optimal CH placement is found when the elected CH-node is the node in the cluster that minimizes the average number of hops between a node in the cluster and the sink, while the worst is found when this number is maximized.

Fig. 4 shows the number of hops necessary to transmit *one* sensor reading using standard routing, compared with EB-clustering with worst and best CH. This is compared with K-means using centroid CHs (k is predefined to match the cluster number proposed by EB-clustering). We observe that electing the optimal CH nodes hardly increases the path-length compared to standard routing, while the worst placements increases the path length considerably. Uniform rotation of

[1]If the topology has changed between two rounds, a new set of clusters is generated. The optimal round frequency depends on the expected data traffic and link stability and is not studied in this paper.

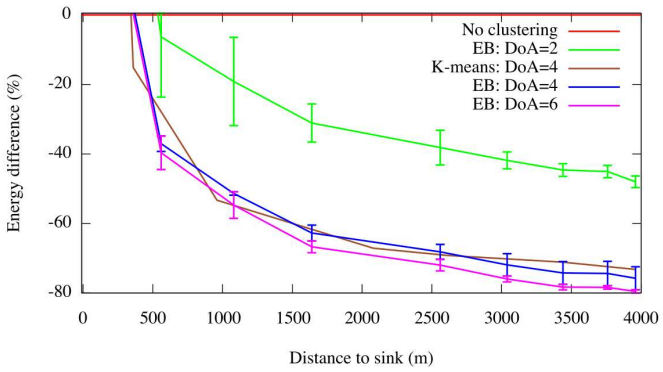

Fig. 5. Energy consumption as a function of distance between a detected event and the sink. Bars indicate worst and best cluster-head placements.

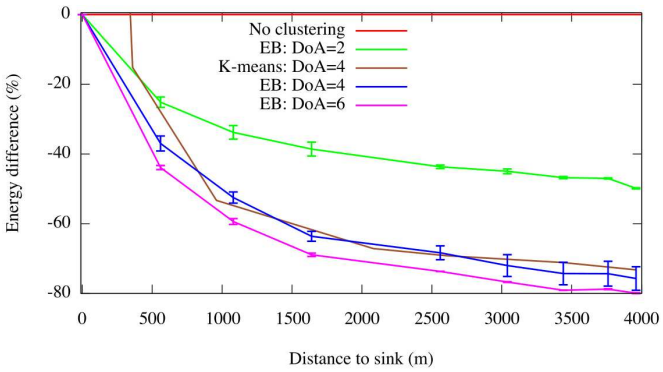

Fig. 6. Energy consumption as a function of distance between a detected event and the sink. Bars indicate min/max.

cluster-heads results in an average of 32% increase in the path length. K-means gives similar average path lengths as EB-clustering. In a real implementation we anticipate that K-means gives longer paths than EB-clustering, since the network topology not always reflect the geographical positions of the sensor nodes (cf. Fig. 1).

Now we focus on the same setup, but apply in-network data-aggregation at the CHs. We apply the term Degree of Aggregation (DoA) from [1], representing the number of messages that the CH receives and aggregates before transmitting data to the sink. DoA depends on the sensing range, the network density, and the signature of the tracked object. The effect of manipulating DoA is shown in Fig. 5. As seen in the figure, the benefit of employing clustering is limited when the detected object is close to the sink and the DoA is low. However, assuming that an event can occur (i.e., a vehicle or intruder is detected) at any position along the network, a DoA of 4 and uniform CH placement, 49% energy reduction can be expected. K-means produce comparable results.

Since an EB-clustering node (be it central or distributed) has full knowledge of *all* clusters in the network, the above result can be optimized. Instead of letting the aggregation role rotate among the cluster members only, we instead exploit nodes from the upstream neighbor cluster. This eliminates the problem of routing packets in the wrong direction. In Fig. 6 we apply rotation only among border-nodes. Here, a DoA of 4 gives an average reduction of 58%. Even a modest DoA of 2, gives an average energy reduction of 36% compared to standard routing.

VII. Conclusions

We have proposed a method to use the edge-betweenness community detection algorithm to determine the clusters and to facilitate in-network data aggregation in roadside sensor networks. The method omits the need for exact geographical positions as in K-means. We have presented both centralized and distributed designs, and results show that clusters can be generated in a consistent way, even with reduced OLSR topology knowledge. The best results are obtained using centralized clustering and our DYMO-low routing protocol scheme. The average energy reduction is 20–62% compared to standard routing, and outperforms K-means. Future works include synchronizing idle-listening within the clusters and implementing the protocols in a test bed.

References

[1] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, and L. Gu, "Energy-efficient surveillance system using wireless sensor networks," in *In Mobisys*. ACM Press, 2004, pp. 270–283.

[2] S. Lindsey, C. Raghavendra, and K. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 13, no. 9, pp. 924 – 935, sep. 2002.

[3] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, 2003.

[4] I. Solis and K. Obraczka, "The impact of timing in data aggregation for sensor networks," in *2004 IEEE ICC*. IEEE, 2004, pp. 3640–3645.

[5] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," in *HICSS '00*, 2000, p. 8020.

[6] W. K. Lai, C.-S. Shieh, and Y.-T. Lee, "A cluster-based routing protocol for wireless sensor networks with adjustable cluster size," in *ChinaCOM 2009*, Aug 2009, pp. 1 –5.

[7] Y. Gong, G. Chen, and L. Tan, "A Balanced Serial K-Means Based Clustering Protocol for Wireless Sensor Networks," *WiCOM '08. Proceeding*, pp. 1 –6, oct. 2008.

[8] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, pp. 026 113+, Feb 2004.

[9] O. V. Drugan, T. Plagemann, and E. Munthe-Kaas, "Detecting Communities in Sparse MANETs," *IEEE/ACM Transactions on Networking*, Accepted for publication in 2011.

[10] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626 (Experimental), Oct. 2003.

[11] T. Clausen, P. Jacquet, and L. Viennot, "Investigating the Impact of Partial Topology in Proactive MANET Routing Protocols," in *WPMC*. IEEE, 2002.

[12] P. E. Villanueva-Pena, T. Kunz, and P. Dhakal, "Extending network knowledge: making OLSR a quality of service conducive protocol," in *IWCMC '06*. New York, NY, USA: ACM, 2006, pp. 103–108.

[13] B. Raman and K. Chebrolu, "Censor networks: a critique of "sensor networks" from a systems perspective," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 75–78, 2008.

[14] K. Kim, G. Montenegro, S. Park, I. Chakeres, and C. Perkins, "Dynamic MANET On-demand for 6LoWPAN (DYMO-low) Routing," Internet Engineering Task Force, Internet-Draft, Jun. 2007, Expired.

[15] University of Murcia, "UM-OLSR," no. http://masimum.dif.um.es/, Accessed 2010.