CLIQUE: Role-Free Clustering with Q-Learning for Wireless Sensor Networks

Anna Förster
Faculty of Informatics, Università della Svizzera Italiana
Via G.Buffi 13, 6900 Lugano, Switzerland
anna.foerster@ieee.org

Amy L. Murphy
FBK-IRST
Via Sommarive 18, 38050 Povo, TN, Italy
murphy@fbk.eu

Abstract

Clustering and aggregation inherently increase wireless sensor network (WSN) lifetime by collecting information within a cluster at a cluster head*, reducing the amount of data through computation, then forwarding it. Traditional approaches, however, both spend extensive communication energy to identify the cluster heads and are inflexible to network dynamics such as those arising from sink mobility, node failure, or dwindling battery reserves. This paper presents* CLIQUE*, an approach for data clustering that saves cluster head selection energy by using machine learning to enable nodes to independently decide whether or not to act as a cluster head on a per-packet basis. We refer to this lack of actual cluster head assignment as being* role-free*, and demonstrate through simulations that, when combined with learning dynamic network properties such as battery reserves, up to 25% less energy is consumed in comparison to a traditional, random cluster head selection approach.*

1. Introduction

Clustering and data aggregation are powerful techniques that inherently reduce energy expenditure in wireless sensor networks while at the same time maintaining sufficient quality of the delivered data. Clustering is defined as the process of dividing the sensor network into groups. Often a single cluster head is then identified within each group and made responsible for collecting and processing data from all group members, then sending it to one or more base stations.

While this approach is seemingly simple and straightforward, efficiently achieving it involves solving four challenging problems. First, the clusters themselves must be identified. Second, cluster heads must be chosen. Third, routes from all nodes to their cluster head must be discovered. And finally, the cluster heads must efficiently route data to the sink(s). This paper focuses on the second and third problems, using existing works for the other two.

The first problem, identifying the clusters, is specific to the application domain. Some solutions generate random clusters [1], while others focus on semantically formed clusters, e.g., grouping all sensors in a geographic area [2] or those with similar data independent of sensor location [3]. Our parallel work identifies non-uniform clusters [4], e.g., small clusters near data sinks and large clusters at a distance. As this is not our focus here, we assume a simple uniform scheme in which a virtual grid overlays the network, forming equal size clusters as shown in Figure 1. This can be trivially accomplished if all nodes know their approximate locations. The key property we rely on is that all nodes know the identifier of the cluster to which they belong.

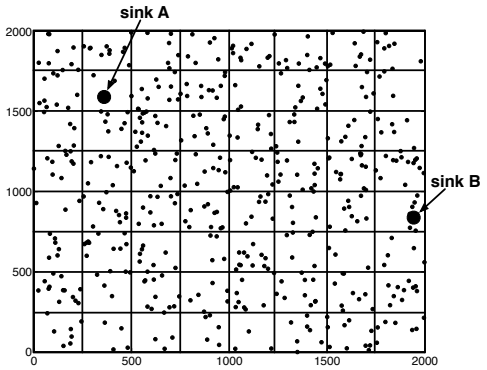

Figure 1. Example of a grid clustered network with cell size 250m, 200 nodes and 2 sinks.

Regarding the fourth problem, routing data from the cluster heads to the sinks, some approaches assume cluster heads communicate directly with the base station, e.g., by boosting transmission power. Such an approach places high energy demands on the cluster heads, and makes unrealistic assumptions about the network size or the position of the sink. Instead, we assume multi-hop communication between the cluster heads and the sinks. Here, we employ our previous work on FROMS [5], a machine learning based, multi-hop, multicast algorithm with very low overhead. This choice allows us to take advantage of cross-layer optimization between FROMS and the work presented here. However, any routing approach can be applied.

This paper focuses on the second and third problems: cluster head identification and intra-cluster routing. Our primary goal is to maximize network lifetime by both minimizing the

energy expenditure at each node and balancing the energy expenditure across all nodes such that no single node is overburdened and thus runs out of energy prematurely. Other approaches, such as randomly selected cluster heads, result in unbalanced intra-cluster communication. Deterministic selection, e.g., based on ID, remaining energy, or other metrics, requires nodes to maintain information about all other nodes in their k-hop neighborhood. In general, cluster head selection requires extensive overhead which must be repeated either periodically in an attempt to balance energy or upon the failure of the cluster head. The key idea in this work is to eliminate the cluster head selection phase all-together and thus its overhead.

Regarding intra-cluster communication, much current research assumes a single hop between all cluster nodes and the cluster head, an assumption that makes selection and propagation of cluster heads easy, but at the same time restricts the domains to which the clustering algorithm is applicable. For example, in scenarios with hundreds or thousands of nodes such as those outlined in Section 2, large clusters are necessary to achieve adequate energy savings, thus requiring multi-hop communication to reach the cluster head. Interestingly, many clustering approaches assume that the cluster head requires more energy, either to reach the sink in a single hop, as previously mentioned, or to perform data collection and aggregation. Upon closer evaluation, we observe that a cluster head actually requires *less* energy than its direct neighbors. Consider that a cluster head gathers data from all other nodes, aggregates it, then sends it out as a single packet. In comparison, its neighbors relay all inter-cluster data packets to it, and thus have higher energy requirements. This is further exacerbated by nodes overhearing messages that they do not need to forward, further consuming energy. Our primary conclusion based on these observations is that cluster head role assignment must take into account not only the current state of the selected cluster heads (e.g., their battery level), but also those of its neighbors and nodes on the paths to the cluster head.

We address this challenge with CLIQUE, a Q-learning based algorithm outlined in Section 3 that learns the optimal cluster heads and intra-cluster routes to them, simultaneously coping with multiple mobile sinks and network dynamics such as changing battery levels and node failures. In contrast to most current approaches, cluster head roles are neither explicitly assigned nor do the nodes need to agree on a cluster head. Instead, each node decides on a per-packet basis whether to act as cluster head (aggregating some packets then sending the result to the sinks) or to forward the packet to a better suited neighbor. This role-free scheme makes the algorithm flexible and robust and eliminates the need for multiple cluster head selection rounds. While this approach may result in multiple nodes acting as cluster heads in a single cluster, intuitively increasing the energy expenditure with respect to traditional schemes, the low overhead of CLIQUE, demonstrated through simulation in Section 4, allows it to achieve overall energy savings up to 25%. Enhancements and potential uses of our technique are offered in Section 5.

2. Application Scenario and Related Work

This section details our target scenario and places our work in the context of the current state of the art.

2.1. Application Scenario

We target a traditional, periodic data reporting application with hundreds of nodes spread randomly over a large area. Examples include micro-climate, structural health, network and environmental monitoring, e.g., volcano monitoring, disaster recovery, pipelines, etc. These and related applications exhibit several common characteristics:

- They consist of many unattended energy-restricted nodes, which may fail unexpectedly. Further, new nodes may join the network.
- The nodes communicate over lossy wireless channels.
- The sensory data gathered in the network needs to be delivered to the destinations directly and cannot be stored at the nodes for later access.
- Reasonable delivery delays are acceptable, e.g., in agricultural monitoring.
- The data gathered at individual nodes can be pre-processed in the network, e.g., correlating related data, compressing, or filtering.
- The destinations can be multiple and mobile, e.g., consider scientist with a laptop walking along a volcano monitoring network.
- The potential clusters are often geographically specified and the nodes have some position or orientation information, e.g., room or floor of the building they are installed in, geographic area, etc.

Given this general scenario, we concentrate on the selection of cluster heads or data aggregators. Clustering and cluster membership assignment are beyond the scope of this paper. We assume only that the clustering is well defined and all nodes know the identity of their clusters. This can be achieved using a geographic grid, as illustrated in Figure 1 or with some other cluster identification protocol.

2.2. Related Efforts

We consider two research areas and their relationship to this work: namely clustering and aggregation and applying machine learning to WSNs.

Network clustering in WSNs. Traditional clustering schemes consist of two basic steps: first identifying the clusters and assigning nodes to them and second electing a cluster head responsible for gathering and aggregating data.

In contrast to most research that addresses both problems without differentiating between them, we focus only on the second. Many clustering protocols are modifications of LEACH [1], in which nodes choose to be cluster heads based on an a-priori probability. Subsequently, cluster heads flood a cluster head role assignment message to their neighbors, which in turn select the nearest cluster head as their own.

Random-clustering algorithms have two advantages. First, they are very simple, and second, they avoid multiple rounds of control messages to converge on a single cluster head per cluster. This, however, comes at the price of unpredictable cluster sizes and shapes. Cluster heads can be anywhere in the network and at times will gather data from a large portion of the network, while at other times only a few readings are aggregated. Another disadvantage is their implicit assumption of one-hop communication and in multi-hop networks they perform poorly due to significant control overhead.

K-hop algorithms [6]–[8] represent an evolution beyond random clustering. They first randomly assign cluster head roles to some nodes in the network and then "grow" clusters around them. In case some node cannot find a cluster head at most k hops away, it becomes a "forced" cluster head [6]. The control overhead is significant, however, cluster sizes are bound by the k-hop requirement.

In contrast to random and k-hop clustering, some approaches reverse the steps of the clustering process: they first identify the clusters and their members and then select one to serve as a cluster head. One such protocol is GROUP [2], which builds a geographic grid over the network. Cluster heads become nodes nearest to the crossing points of the grid. A bounded number of messages are needed in each cluster to agree on the cluster head. To load-balance the nodes, GROUP periodically moves the origin of the grid and informs all nodes with a network-wide broadcast. The main advantage of GROUP is that clusters are well defined. Nevertheless, both agreeing on cluster heads and the network-wide broadcasts with the new grid origin incur significant communication overhead.

Another geographic-based clustering approach is applied in [9] for multi-resolution in-network storage of data for WSNs. A hash function is used to map cluster head roles to network locations and nodes nearest to those locations become cluster heads. The approach is similar to GROUP [2] and needs agreement rounds among the cluster members.

Some communication protocols, such as TTDD [10], are a combination of routing and clustering. They build clusters then use only cluster heads for data routing. Their main requirement is that cluster heads can communicate directly to one other to form a routing overlay. Thus, they are more mesh or overlay routing protocols rather than traditional clustering approaches for data aggregation. Further, no load balancing of individual nodes is performed and cluster heads drain their batteries very fast due to the routing overlay.

There are many clustering algorithms that require full network topology and/or remaining energy information to centrally compute optimal clusters [11], fully disseminating cluster information in each round. However, such approaches do not scale and do not consider fundamental network issues such as failures and asymmetric links.

Machine learning for routing and clustering in WSNs. One of the most fundamental works using machine learning for WSNs is Q-Routing [12], which describes a Q-learning based routing algorithm that learns the minimum latency route from a single source to a single destination. Many works have been inspired by Q-Routing, such as Q-Routing with compression (Q-RC) [13], where the routing objective is not only the lowest delay, but also the maximum aggregation rate. The scenario assumes that all data from all nodes in the network needs to be aggregated on its way to the single sink. This work is similar to ours, since we also use Q-learning to learn the best aggregation (clustering) heads. However, our routing is only intra-cluster, the learning objective is different and, most important, the destinations are multiple and mobile. Additionally, [13] requires global topology knowledge.

Further related works on machine learning for routing and clustering in WSNs are described in [14].

3. Cluster Head Selection and Routing in CLIQUE

The main contribution of this paper is an innovative cluster head identification algorithm with nearly zero communication overhead. It copes with network dynamics such as those arising from the mobility of multiple sinks or node failures. This section first describes our assumptions, then offers a high level overview. It then offers the necessary Q-learning background and protocol details.

3.1. Prerequisites and Assumptions

CLIQUE assumes nodes know the identity of the cluster to which they belong. This can either be established manually (e.g., assigning room or geographic area) or computed on the fly (e.g., using a geographic grid). The shapes and sizes of the clusters are not important to CLIQUE, meaning it does not rely on any specific cluster property. For simplicity, we assume cluster membership is based on a rectangular uniform grid, as shown in Figure 1, and each node knows the identity of the cell to which it belongs. The computation of such a grid is straightforward when even approximate position information is available.

We further assume that cluster members are multiple hops away from one another and while they know their direct one-hop neighbors, they do not have information about all other nodes in the same cluster.

Finally, we assume that the data destinations, the sinks, flood the network with DATA_REQUEST packets, announcing their data interest, a common requirement in WSN systems. We further assume all sinks require the same data, which is periodically produced by all network nodes.

3.2. Overview

Clustering and aggregation in WSNs is typically performed in two steps: first, clusters are identified, and second, cluster head roles are assigned. Sometimes this sequence is reversed: the cluster head roles are first assigned, then nodes join the nearest cluster head, thus forming clusters. Nevertheless, in both scenarios the nodes must agree on the cluster heads and to find routes to them.

In contrast, we propose a cluster head assignment algorithm in which nodes do not need to know the identities of the cluster heads. Given knowledge about their cluster identifier and basic information about their one-hop neighbors, each node tries to route its data directly to *all* sinks while taking the simple decision of whether to act as a cluster head and aggregate data or to route the incoming data to a neighbor better suited for this role.

While straightforward to explain, one major challenge remains: **how do nodes, using only neighbor information, decide both the next hop and whether or not to act as a cluster head**? To address this, we observe that sinks flood the network with DATA_REQUEST packets, with the result that each node knows some routing information regarding each *individual* sink in terms of parameters such as hop count, geographic location, or battery status. However, our challenge is to route to multiple sinks by first routing to the cluster heads most appropriate for each cluster. Therefore, each node must combine the single-sink routing information to identify the cost to route to multiple sinks *and* decide whether or not to act as the cluster head. Unfortunately, the local information from the DATA_REQUEST packets provides only an approximate, upper-bound on the cost, and does not take into account the real multicast cost to the sinks. Therefore, a node can only approximate the total routing costs, and further can only make a best estimate about whether or not to serve as a cluster head. Nevertheless, these approximations are reasonable, and using them eliminates substantial communication overhead found in traditional cluster head assignment.

To improve the localized cost estimates, we employ Q-learning, incrementally learning the real global costs to all sinks through the best cluster head. Additional information is gathered by exchanging feedback among neighbors while routing data packets. We calculate routing cost using a combination of hop counts to reach the sinks (through the cluster heads) and battery status of the nodes on the routes to the sinks. Note that the data is always routed first to the cluster head where it is aggregated. Only after this point can it be duplicated to follow multiple paths to the sinks. Any splitting before aggregation would result in multiple cluster heads aggregating data for each of the sinks and thus significantly increasing the communication overhead both for routing to the cluster heads and to the sinks.

3.3. Q-Learning Model and Solution

Q-learning [15] is a model-free reinforcement learning technique, based on agents taking actions and receiving scalar rewards from the environment in response to those actions. It assigns *Q-values* to each possible action, representing the approximate *goodness* of the action. In the learning process, the agent selects and executes one action, then receives the reward, which it uses to update the Q-value. Over time the agent learns the real action values (costs), which it uses to select the most appropriate route. Q-learning has been widely applied in robotics, wireless ad hoc communications [14], etc. Its main challenge is to properly model the agent and define the Q-values.

In our cluster head routing scenario, each sensor node is an independent learning agent, and actions are routing options using different neighbors as the next hop toward the cluster head. **The cluster head is defined as the cluster node with the best (lowest) routing cost to all sinks**. The following provides detail for our Q-learning solution.

Agent states. The state of an agent is a tuple $\{D, cost_D^{N+self}\}$, where D is the set of sinks and $cost_D^{N+self}$ is essentially all available routing information on this node through all its neighbors N, including the costs of the node itself.

Actions. An action identifies the next hop on the route to the cluster head. This could be a neighbor or the node itself if it is acting as cluster head and aggregating packets. We define a possible action as $a_{n_i} = (n_i, D)$ with $n_i \in \{N, self\}$.

Q-values. Q-values represent the goodness of actions and the goal of the agent is to learn the *actual* goodness of the available actions. In our case, Q-values represent an estimate of the cost to route through a neighbor, a value composed of two parts. The first part is the broadcast hop count to reach all sinks from the agent and the second is the minimum battery level among the nodes on the route to the sinks through this neighbor. The first part accounts for energy efficiency by minimizing communication overhead. The second, the minimum battery of the nodes, allows very low powered nodes to be avoided. The two elements of the cost function are united with a *hop count multiplier (hcm)* value, which grows exponentially for decreasing battery levels [16]. This means that when the batteries of the nodes are full, the routing cost of a neighbor is exactly the number of hops to reach the sinks; while with decreasing batteries this cost exponentially grows, giving preference to higher powered nodes on possibly longer routes.

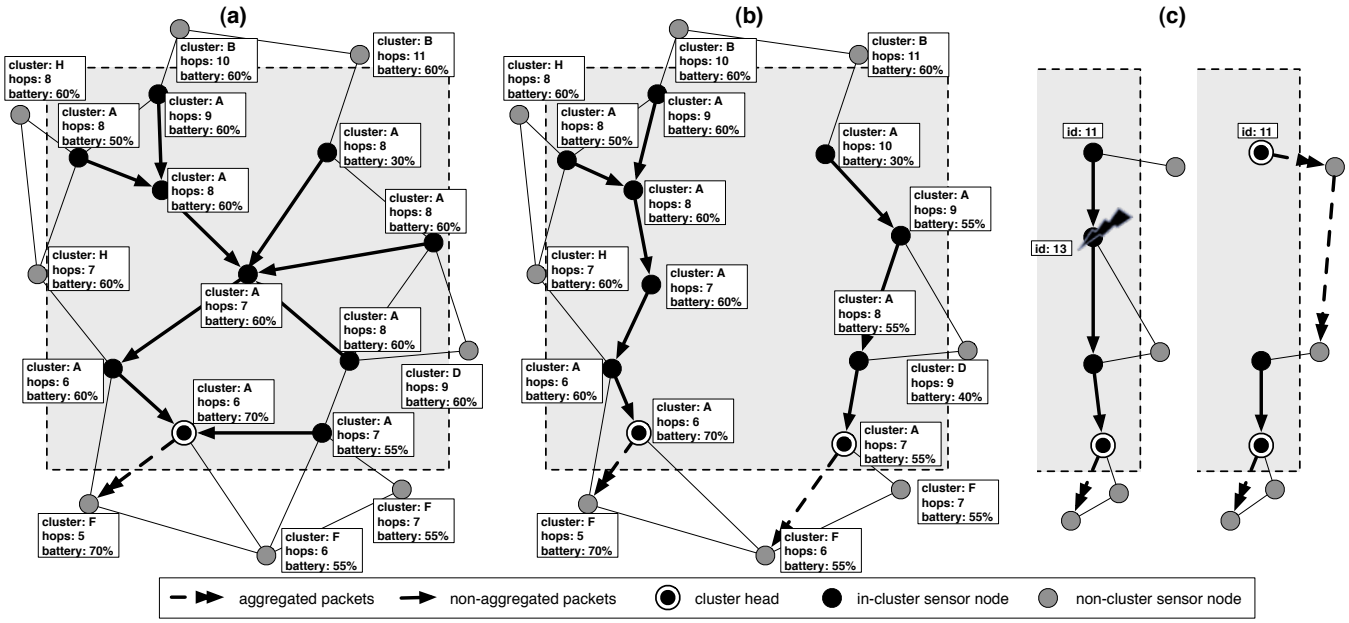

Figure 2. Learned cluster head in a connected scenario (a), in a disconnected scenario (b) and recovery after node failure (c). Data gathering and aggregation is shown only inside the cluster.

To initialize these values, we could use random values, as is common in many learning approaches. However, we use a more sophisticated approach that calculates an estimate based on the hop counts to individual sinks available in a standard routing table, thus speeding up the learning process.

The initial Q-value for an action $a_{n_i} = (n_i, D)$ is:

$$\begin{aligned} Q(a_{n_i}) &= Q_{hops}(a_{n_i}) * Q_{battery}(a_{n_i})) \\ &= \sum_{d \in D} hops_d^{n_i} * hcm(bat_{n_i}) \end{aligned} \quad (1)$$

where $hops_d^{n_i}$ is the number of hops neighbor n_i needs to reach sink d. The initial value of the battery element is set to the battery status of neighbor n_i. Note that the hop-count estimation is an *upper bound* of the real costs, because subsequent hops are expected to be able to share routes to multiple sinks, decreasing the number of transmissions needed to reach the sinks. On the other hand, the battery element is expected to decrease, because battery levels decrease. Thus, the Q-values are expected first to drop, reflecting the learning of the real hop costs to reach the sinks, and then to slowly and constantly increase because of depleting energy on the nodes.

Updating a Q-value. To learn the real values of the actions, the agent uses the reward values from the environment. In our case, each neighbor to which a data packet is forwarded sends the reward, its best Q-value, piggybacked on its next data packet. The new Q-value of the action is:

$$Q_{new}(a_{n_i}) = Q_{old}(a_{n_i}) + \alpha(R(a_{n_i}) - Q_{old}(a_{n_i})) \quad (2)$$

where $R(a_{n_i})$ is the reward value and α is the learning rate of the algorithm. We use $\alpha = 1$ to speed up learning and because we initialize the Q-values with non-random values. Therefore, with $\alpha = 1$, the formula becomes $Q_{new}(a_{n_i}) = R(a_{n_i})$, directly updating the Q-value with the reward.

Reward function. Intuitively the reward function is the downstream node's opportunity to inform the upstream neighbors of its actual cost for the requested action. Thus, when calculating the reward, the node selects its *lowest Q-value* among all its actions and adds the real action cost:

$$R(n_{self}) = c_{n_i} + \min_{n_i \in N} Q(a_{n_i}) \quad (3)$$

where c_{n_i} is the cost of reaching node n_i and is always 1 (hop) in our model. This propagation of Q-values upstream is piggybacked on usual DATA packets and allows all nodes to eventually learn the actual costs.

Exploration strategy (action selection policy). One final, important learning parameter is the action selection policy. A trivial solution is to greedily select the action with the best (lowest) Q-value. However, this policy ignores some actions that may, after learning, have lower Q-values, resulting in a locally optimal solution. Therefore, a tradeoff is required between exploitation of good routes and exploration among available routes. This problem has been extensively studied in machine learning [15]. Here we chose the standard ϵ-*greedy strategy*, which selects a random route with probability ϵ and the best route otherwise. Our previous work [16] showed that a dynamic cost function whose value changes continuously over time, such the one here based on battery level, results in implicit exploration of the routes. This is because the changing route costs force the protocol to switch to other, less costly routes, thus also learning their real costs.

3.4. Key Properties, Convergence and Optimality

The most important property of CLIQUE is its role-free nature. In contrast to most cluster head selection algorithms, it does not try to find the optimal cluster head (in terms of cost), but incrementally *learns* the best without knowing either where or who the real cluster heads are. As a result, at the beginning of the protocol, multiple nodes in the cluster may act as cluster heads. While this temporarily increases the overhead, it is a short-term tradeoff in comparison to the overhead required to agree on a single cluster head. Later in the protocol operation, after the real costs have been learned, multiple cluster heads occur only in disconnected clusters, where a single cluster head cannot serve all cluster members.

Even in the case of a connected cluster in which two nodes have exactly the same routing costs and are *not* neighbors, there will exist a single node in this cluster who needs to decide to which of the two possible cluster heads to route the data. If both options have exactly the same routing costs, a simple ID-based tie-break will uniquely identify one option. If the routes to both possible cluster heads have different costs (e.g., there is a depleted node on the path to one of them), then the data will be routed automatically to the cluster head with the lower cost route. This intuitively argues that CLIQUE *converges* to a single cluster head (iff the cluster is connected).

Figure 2 shows some cluster head learning scenarios. Scenario (a) presents a simple cluster, where all nodes within the cluster are connected. The optimal cluster head lies in the lower left corner of the cluster because all sinks lie in this direction. A more problematic scenario is presented in Figure 2(b), where the cluster is disconnected. Such a scenario is challenging for traditional clustering approaches as they need a complicated recovery mechanism which requires large control overhead. On the contrary, CLIQUE automatically identifies two cluster heads, as shown in the figure.

Figure 2(c) shows a recovery scenario in which node 13 fails. Node 11 is no longer able to send its data to the cluster head and needs to find a new solution. Instead of searching for a new route to the cluster head it simply becomes a cluster head itself. Because of its learning properties and network status awareness, this requires no control overhead.

It is also worth noting that CLIQUE is a reactive protocol: it only learns the optimal cluster heads when data traffic is flowing. If some part of the network remains silent, it will not spend any energy on learning or clustering. Note also that after aggregation, the packet is routed to the sink via the routing protocol.

3.5. Sink Mobility

CLIQUE also handles sink mobility. When a destination moves, it must re-broadcast its data request to update routing information at all nodes. However, when we combine CLIQUE with the FROMS routing protocol (our multicast reinforcement learning based algorithm), the full-network broadcasts can be avoided. They are compensated by the feedback (reward) mechanism, which updates the Q-values at all nodes while forwarding normal data to the destinations. Thus, the sinks need to broadcast their data request only one hop to their immediate neighbors, who then piggyback the new Q-values (costs to reach the sinks) to the other nodes.

Table 1. Energy expenditure model for our simulations (taken from the MICA-2 data sheet)

mode	energy spent
sleeping	0.054 mW
idle	66 mW
RX	117 mW
TX	117 mW

4. Simulation Results

We validate CLIQUE through simulation, and begin with a sketch of the simulation environment.

4.1. Simulation Environment

Our evaluation employs OMNeT++[1], the event-based network simulator that is gaining importance for both wired and wireless simulations. We combine OMNeT++ with the Mobility Framework, which allows for fast implementation of communication protocols for WSNs, and the Feedback Framework, our implementation supporting feedback-driven protocols. Further, our configuration includes a Nakagami-1 radio probabilistic propagation model, linear battery model with 3000 mA initial energy (2 AAA batteries) with the energy expenditure model given in Table 1, a CSMA MAC protocol, and an application layer that generates a single DATA packet at every node except the sinks every 2 sec.

Unless otherwise stated, our experiments run with 30 random connected networks with 30 random seeds (900 runs in total). The network scenario contains from 100 to 300 nodes in an area of 2000x2000m, with 1 to 5 sinks and cluster sizes from 250m to 1000m.

4.2. Comparative Study

As sketched in Section 2.2, there are many different clustering approaches for WSNs. To make a fair comparison to CLIQUE, we need an algorithm that uses the same grid clustering approach. However, most of the grid-based algorithms require one-hop connectivity between all cluster members, which is not true for CLIQUE. Conversely, algorithms that have k-hop connectivity inside the clusters are usually not

1. http://www.omnetpp.org/

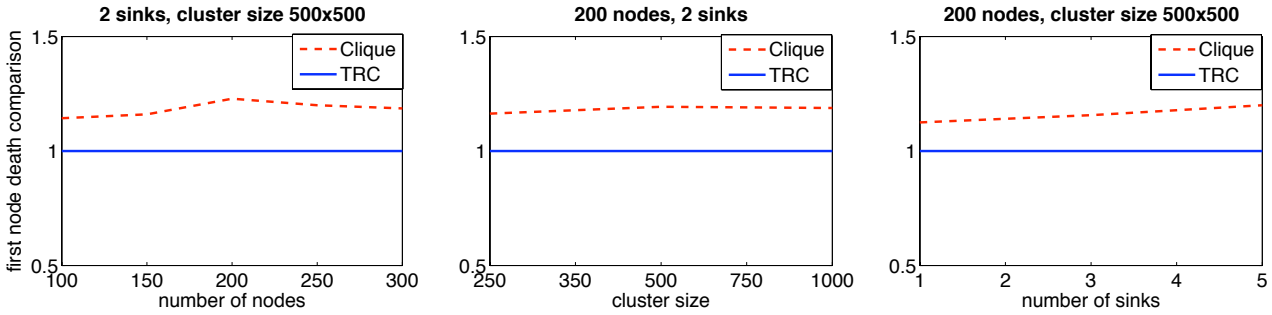

Figure 3. First node death for different number of sinks, cluster sizes and nodes in the network

grid-based. Thus, we have chosen a variation of the work presented in [6], which we call the Traditional Random Clusterhead Assignment (TRC). The original algorithm uses an a-priori probability at each node to decide whether or not to become a cluster head. If it does, the node broadcasts a notification. Non-cluster head nodes simply join the nearest cluster head. Our modification consists of clustering the network exactly as in CLIQUE. Then, the probability-based cluster head assignment from [6] is applied, and nodes join the nearest in-cluster cluster head. If no cluster head is announced after some time, the algorithm is re-run and data packets are buffered. The algorithm runs periodically to spread energy expenditure among the nodes. The probability is derived from the expected number of nodes in one cluster and is different for different network deployments:

$$P_{clusterhead} = \frac{N_{clusters}}{N_{nodes}} \quad (4)$$

With both clustering protocols, cluster heads collect all arriving packets for a predefined interval (200 msec in our evaluation), aggregate them, then forward the aggregate to the sinks using the routing protocol.

For our study, we use the FROMS multicast routing protocol [5] to forward aggregated packets from cluster heads to sinks. Similarly to CLIQUE, FROMS is based on reinforcement learning, and uses a cost metric which is a combination of remaining node energy and number of hops to the sinks. Furthermore, FROMS handles mobility and recovery after node failures. The combination of CLIQUE with FROMS is efficient since both use the same cost metric and are designed for multiple mobile sinks. We also consider FROMS as a good choice to combine with TRC, however any multi-hop multicast routing protocol may be used for either clustering approach.

4.3. Evaluation Results

With our main goal of minimizing network energy expenditure, we consider multiple metrics.

First node death. This is a good indicator for the expected lifetime of network as it shows how well clustering and routing avoid bottlenecks and spread energy expenditure.

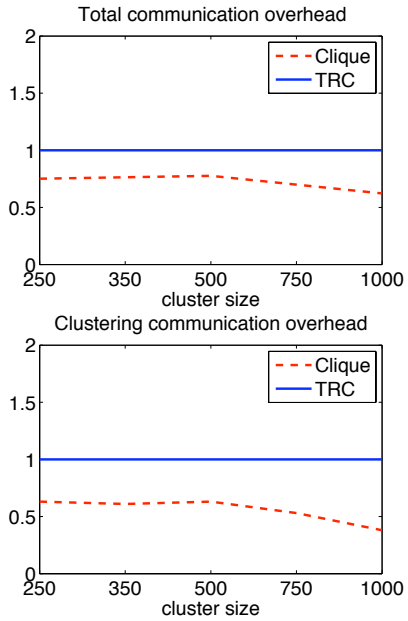

Figure 6. Total (top) and intra-cluster communication (bottom) in number of generated packets of all types

The later the first node dies the better. We evaluate the network lifetime and scalability of CLIQUE in three dimensions: increasing number of sinks, increasing cluster sizes (more nodes per cluster), and increasing density of network nodes.

Figure 3 shows that CLIQUE prolongs network lifetime in terms of first node death by 20-25% and scales well in all load tests. Interestingly, while the gain is constant for different numbers of nodes or different cluster sizes, it increases with increasing number of sinks. This is because of the multicast nature of CLIQUE, which considers the routing costs to all sinks.

Standard deviation of the remaining energy on the nodes. This is closely related to efficiently balancing energy consumption and shows the balance of node usage during the network lifetime (until first node death). Low standard deviation implies good balancing. Figure 4 summarizes the results over the same three dimensions as before (number of sinks, node density and cluster sizes). It is worth noting that

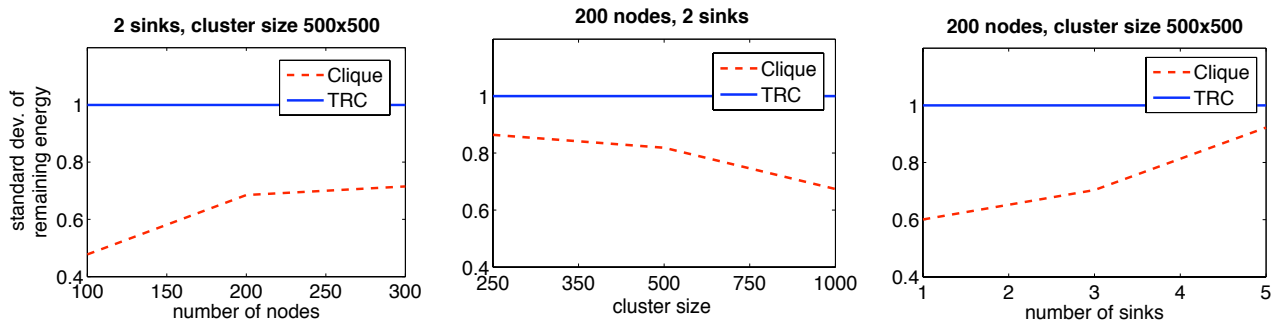

Figure 4. Standard deviation of remaining energies at first node death for different number of sinks, cluster sizes and nodes in the network

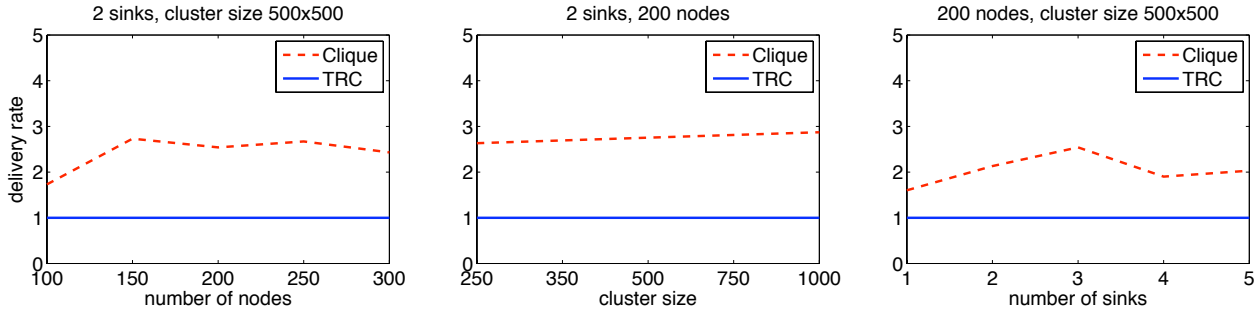

Figure 5. Delivery rate at the sinks for different number of sinks, cluster sizes and node density

with an increasing number of sinks CLIQUE does not achieve much better energy spreading than TRC. This is because the overall communication load increases so much that TRC also uses many nodes, implicitly spreading the load. However, note also that in the same scenario CLIQUE prolongs the network lifetime by almost 25% (Figure 3 rightmost).

CLIQUE's ability to spread energy expenditure is especially clear in Figure 4 center. Here, with increasing node density, CLIQUE makes extensive use of multiple routing options, thus spreading the load among the nodes.

Delivery rate. This is closely related to network communication overhead: with more packets transmitted, the probability of collisions or overflowing MAC layer buffers increases. However, it should be noted that delivery rate is not an absolute metric in a simulation environment and instead simply provides intuition of the expected behavior. Figure 5 summarizes the results. The behavior of CLIQUE can be characterized as stable, as it always shows a delivery rate increase between 20%-30%. It is interesting to consider these results with those in Figure 6 where the total number of generated packets is presented (see below).

Generated packets. Here we count the total number of packets generated in the whole network, including data requests, aggregated and non-aggregated data packets and cluster head selection packets. Figure 6 (top) shows that the overall communication overhead is reduced with CLIQUE by 25%-30%, due mostly to the reduced in-cluster communication, see Figure 6 (bottom). We show these results for increasing cluster sizes (increasing number of nodes in one cluster with constant node density) since it shows clearly how CLIQUE saves even more communication overhead with growing clusters.

Total energy spent. This measures the ability of the clustering approach to minimize network communication as a whole. While longer network lifetimes can be due either to better resource balancing or to less communication overhead, the total spent energy clearly shows the communication overhead. Low energy spent implies less overhead.

The results in Figure 7 show that not only does CLIQUE reduce the energy spent by 25% (left plot), it also doubles the delivery rate (right plot). This is because the communication overhead is significantly smaller, thus also reducing collisions. The total communication overhead (center plot) is nearly halved: however, this does not result in halving the total energy due to packet overhearing.

5. Summary and Future Work

This paper presented CLIQUE, a novel role-free reinforcement learning based clustering approach that learns the optimal cluster head without control overhead. Simulation results show that approximately 25% less energy is

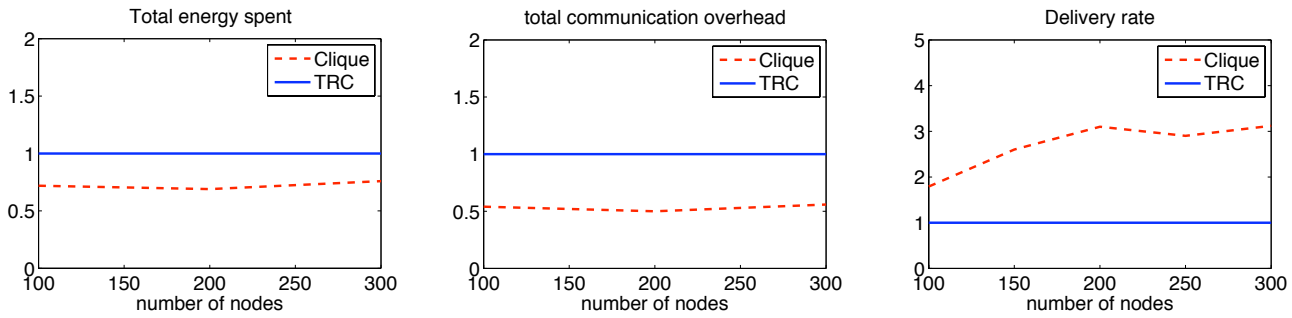

Figure 7. Energy expenditure for networks with unlimited energy for 1000 seconds.

spent compared to a traditional random clustering algorithm. CLIQUE enjoys wide applicability in scenarios where regular data reporting and in-network aggregation are desirable. Its most important properties are its fully distributed, localized role-free nature, its current network status awareness, and its flexibility in case of node and link failures.

We plan to implement CLIQUE in a testbed environment in the near future with multiple mobile sinks to verify the simulation results. At the same time we will continue exploring the non-uniform data dissemination paradigm [4] and evaluate CLIQUE in a non-uniform setting.

References

[1] W. Rabiner-Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Washington DC, USA, 2000.

[2] L. Yu, N. Wang, W. Zhang, and C. Zheng, "Group: A grid-clustering routing protocol for wireless sensor networks," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, Wuhan, China, December 2006.

[3] D. Tulone and S. Madden, "Paq: Time series forecasting for approximate query answering in sensor networks." *Proceedings of the 3rd European Conference for Wireless Sensor Networks (EWSN)*, 2006.

[4] A. Egorova-Förster and A. L. Murphy, "Exploring non uniform quality of service for extending WSN lifetime," in *Proceedings of the 3rd International Workshop on Sensor Networks and Systems for Pervasive Computing (PerSens)*. White Plains, NY, USA: IEEE Computer Society, 2007.

[5] A. Förster and A. L. Murphy, "FROMS: Feedback routing for optimizing multiple sinks in WSN with reinforcement learning," in *Proceedings 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2007.

[6] S. Bandyopadhyay and E. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 3, March 2003, pp. 1713 – 1723.

[7] M. Demirbas, A. Arora, V. Mittal, and V. Kulathumani, "Design and analysis of a fast local clustering service for wireless sensor networks," in *Proceedings of the 1st International Conference on Broadband Wireless Networking (BroadNets)*, 2004, pp. 700–709.

[8] O. Younis and S. Fahmy, "Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[9] D. Ganesan, B. Greenstein, D. Estrin, J. Heidemann, and R. Govindan, "Multiresolution storage and search in sensor networks," *ACM Transactions on Storage*, vol. 1, no. 3, pp. 277–315, August 2005.

[10] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, "TTDD: Two-tier data dissemination in large-scale wireless sensor networks," *Wireless Networks*, vol. 11, no. 1-2, pp. 161–175, 2005.

[11] A. W. Matin and S. Hussain, "Intelligent hierarchical cluster-based routing," in *Proceedings of the International Workshop on Mobility and Scalability in Wireless Sensor Networks (MSWSN)*, San Francisco, CA, 2006.

[12] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," *Advances in Neural Information Processing Systems*, vol. 6, 1994.

[13] P. Beyens, M. Peeters, K. Steenhaut, and A. Nowe, "Routing with compression in wireless sensor networks: A Q-learning approach," in *Proceedings of the 5th European Workshop on Adaptive Agents and Multi-Agent Systems (AAMAS)*, 2005.

[14] A. Förster, "Machine learning techniques applied to wireless ad-hoc networks: Guide and survey," in *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2007.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, March 1998.

[16] A. Förster and A. L. Murphy, "Balancing Energy Expenditure in WSNs through Reinforcement Learning: A Study," in *Proceedings of the 1st International Workshop on Energy in Wireless Sensor Networks (WEWSN)*, Santorini Island, Greece, 2008.