On Maintaining Sensor-Actor Connectivity in Wireless Sensor and Actor Networks *

Jie Wu, Shuhui Yang, and Mihaela Cardei
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

Abstract

In wireless sensor and actor networks (WSANs), a group of sensors and actors are connected by a wireless medium to perform distributed sensing and acting tasks. Sensors usually gather information in an event area. They pass it on to actors, which are resource-rich devices that take decisions and perform necessary actions. Therefore, it is vital to maintain connections between sensors and actors for effective sensor-actor coordination. In this paper, we first define several sensor-actor connection requirements, including weak and strong actor-connectivity, and then propose several local solutions that put as many sensors as possible to sleep for energy saving purposes, while meeting different actor-connectivity requirements. We also prove the relationship between the proposed actor-connectivity and the connectivity in regular graphs, which helps with the implementation of the proposed solutions. Comprehensive performance analysis is conducted through simulations.

Keywords: Connectivity, fault-tolerance, local solutions, wireless sensor and actor networks (WSANs).

*This work was supported in part by NSF grants ANI 0083836, CCR 0329741, CNS 0422762, CNS 0434533, EIA 0130806, CNS 0531410, CCF 0545488, and CNS 0626240. Email: jie@cse.fau.edu.

1 Introduction

Recent technological advances have lead to the emergence of distributed hybrid sensor networks consisting of both resource-rich sensor devices (called *actors*) and resource-impoverished sensor devices (called *sensors*). Such a network, called a wireless sensor and actor network (WSAN) [1, 9, 13, 15], is shown in Figure 1. In this figure, sensors are represented as circles and actors as triangles. Actors are connected among themselves and to the sink through special channels.

Typically, when sensors detect a phenomenon, they either transmit data to actor nodes (also called actuators) which then initiate appropriate actions, or route data to the sink which then issues action commands to actors. In this paper, we focus on the former approach, also called automated architecture [1], where actors are deployed to perform distributed actuation tasks upon the environment. For example, a smoke detector (sensor) reports a fire event to one or several nearest water sprinklers (actors) instead of the distant central control system (sink). The water sprinkler(s) then perform an action and report the event to the central system for further processing.

There are two types of coordinations: actor-actor and actor-sensor. In this paper we focus only on the actor-sensor coordination. The number of actors is relatively small and since they are resource-rich devices with a long transmission range, their connection to the sink can be treated in a relatively easy way [9, 18]. For example, a separate wireless interface can be used to communicate with neighboring actors so they can perform long-range communication without any involvement from the sensors.

Existing actor-sensor coordination focuses on energy-efficient connectivity from a sensor to a nearby actor. The approach in [12] borrows the concepts of efficient routing protocols for wireless sensor networks (WSNs). It constructs a tree rooted at the event source sensor to perform a reverse tree-based anycast routing to a nearby actor. Some other approaches construct a cluster structure with each cluster being a tree rooted at an actor, triggered by an on-the-fly event, thus minimizing the routing energy expenditure [17], or constructed only once during the network initialization to route data from a sensor to a nearby actor through a maximum remaining energy path [18]. In [4], energy efficiency is achieved through topology control where each sensor adjusts its transmission range while preserving actor connectivity. However, none of these existing approaches are localized. *In a local solution, unlike the traditional distributed solutions, a decision at each node is purely based on local information and there is no information propagation.*

In this paper, we use a different approach to construct a self-organizing framework for data routing from sensors to actors. We first give a formal graph model for WSANs. We propose several local solutions for maintaining different versions of sensor-actor connectivity by putting as many sensors to sleep as possible, while still considering area coverage and fault-tolerance. In these solutions, only neighborhood information (neighbor set) is required, and location/distance information is not used. In addition, other issues such as sensor

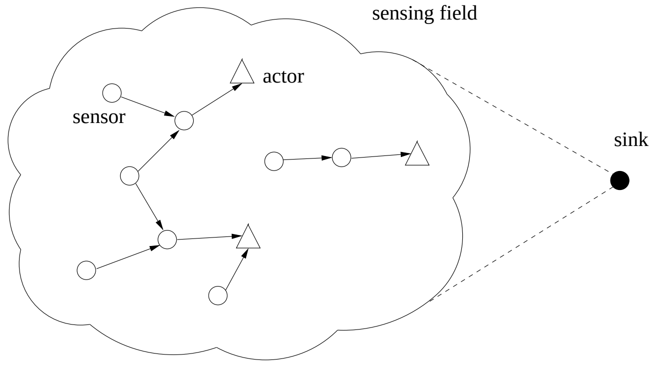


Figure 1: A sample data gathering process in a WSAN with circles representing sensors and triangles representing actors.

energy efficiency and delay sensitivity of individual routing paths are discussed. More specifically, instead of finding efficient routes from sensors to actors in the entire network, we try to reduce the routing space by putting as many sensors to sleep as possible to limit the energy consumption subject to the following two requirements:

- **Coverage**: each sleeping sensor has at least one neighbor that is either an active sensor or an actor.
- **Connectivity**: each active sensor is still connected to the same set of actors as it was before sensors were put to sleep (called *persistent actor-connectivity*) or to at least one actor (called *at-least-one actor-connectivity*).

The coverage requirement is used to ensure the coverage of all the sensors which are discrete points. This point coverage can approximate the area coverage, especially when sensors are densely deployed [5]. The connectivity requirement ensures that information collected by any active sensor can be delivered to at least one actor in at-least-one actor-connectivity. In certain situations, connection to one actor is not sufficient. Multiple actors should be informed in order to make decisions of the most appropriate way of performing actions.

A sufficient degree of connectivity is needed to protect against the loss of sensors and actors due to failure or battery depletion. To ensure a certain degree of fault-tolerance, the network should still meet the coverage and connectivity requirement after removing $k-1$ arbitrary nodes (sleep sensors, active sensors, or actors). The corresponding property is called k-actor-connectivity. In addition, more active sensors and a higher connectivity degree help to find a more efficient route in terms of delay.

We propose several local solutions that put as many sensors to sleep as possible while meeting different coverage and connection requirements. Note that we try to minimize the energy consumption in a single iteration. Network activity is organized as a sequence of iterations, where a new schedule is decided at the beginning of each iteration. Sensors can be scheduled to work in different iterations such that to balance the energy consumption and to prolong the network lifetime. In this paper, we do not deal with the actual routing protocol which can be designed on the active sensors derived from our methods.

In summary, we will focus on the following technical issues in this paper:

1. We give a formal graph model for WSANs and define several sensor-actor connections based on the coordination requirement.
2. We develop two local solutions for the different versions of the sensor-actor connectivity.
3. We prove the relationship between the traditional connectivity in graph and the new defined sensor-actor connectivity.
4. We extend the sensor-actor connectivity and the corresponding local solutions for fault-tolerant consideration.
5. We conduct performance analysis through simulations on the proposed algorithms.

The rest of the paper is organized as follows: Section 2 reviews the related work in WSANs. Section 3 presents a graph model for WSANs and several connectivity requirements. The proposed local solutions for different connectivity requirements are given in Section 4. The fault-tolerance extensions are presented in Section 5, followed by several properties in Section 6. Section 7 discusses several implementation issues and Section 8 presents the simulation results. The paper concludes in Section 9.

2 Related Work

The traditional wireless sensor networks (WSNs) usually contain only a single sink and perform the sensing in a distributed way. However the management is centralized at the sink. WSANs contain actors in addition to a sink and perform both distributed sensing and management. As shown in Figure 1, a WSAN is a two-tier architecture with sensors on the lower layer and actors on the upper layer. Actors provide real-time distributed management, they can coordinate among themselves, and they can communicate with the sink for further instructions. WSANS can be used as an integral part of some novel, low-cost, high-performance systems [1, 2, 7, 11, 20], and can provide the infrastructure of various applications such as battlefield surveillance, nuclear, biological, or chemical attack detection, and environmental or health monitoring. [1] provides a comprehensive survey of WSANs and related research issues.

Energy-efficient routing protocols is a major research issue for energy constrained WSNs. Many energy-aware routing schemes that prolong network lifetime have been proposed [3, 14, 21]. WSANs have two unique coordinations compared with WSNs [1]: actor-actor and actor-sensor coordinations. Therefore, routing protocols designed for WSANs should be both energy-efficient and coordination-sensitive. Additionally, the actor-related distributed coordination raises a new research issue. Most of the existing works focus on the design of a self-organizing framework for connecting sensors and actors. The proposed solutions under this framework, however, are distributed but not localized.

Some approaches construct a tree-structure rooted at each actor in a distributed way, and can be viewed as a many (sensor)-to-one (actor) connection. In [17], a sensor-actor connection model based on an event-driven clustering paradigm is proposed, and in each cluster, a data tree is formed. Trees are created on-the-fly and are triggered by an event. This approach considers the tradeoffs between energy efficiency and reliability in the routing procedure. Location information via GPS is necessary in this approach. However, creating a routing framework on-the-fly may require a high overhead. Also, the assumption of each sensor knowing the positions of actors is rather strong. In [18], trees are formed in the initialization procedure and data is collected on the paths with maximum remaining energy. A distributed actor-discovery protocol is developed without the assumption of knowing the actor positions.

The tree-structure rooted at each sensor is also developed, which forms the many-to-many connection. [12] extends the protocols in WSNs for WSANs and proposes a reverse tree-based anycast routing structure. It constructs a tree rooted at the event source where actors can join and leave dynamically as the leaves of the tree. According to the detailed requirement, each sensor can choose to connect to one or more actors. In [19], every sensor finds paths connecting to each actor. Nodes are randomly put to sleep in this approach. Therefore, when constructing the tree to connect to each actor, not only distance but also the status (sleep/active) of the nodes is considered to balance the latency and energy consumption. A power-aware many-to-many routing structure is proposed in [6]. Actors broadcast interests and register in each sensor. When data is collected, a sensor routes data to the corresponding registered actors. Routes are created in the registration procedure. Location information is necessary in this approach.

Some other issues in WSANs are also discussed. For example, [11] discusses control engineering problems and existing technologies in WSANs. In [9, 17], actor-actor coordination is addressed. [4] solves the topology control problem in WSANs considering both energy-efficiency and reliability. The communication range of each sensor is adjusted to reduce total energy consumption while maintaining certain connectivity to the actor(s). Like other topology control methods, this approach requires distance information.

The work proposed in this paper aims at minimizing the entire routing space instead of finding the exact routes from sensors to actors. Our work differs from the other works by considering a qualified minimal forwarding set for all the sensors, that meets the efficiency and reliability requirements. Note that in our paper we measure the routing energy consumption in terms of hop counts as opposed to distance. We also consider fault tolerance. That is, the selected forwarding set can tolerate the failures of up to $k-1$ nodes, including actors. To the best of our knowledge, our work is the first to deal with fault tolerance in WSANs. We focus on the development of localized solutions which rely only on local information, i.e., properties of nodes within their vicinities. In addition, there is no sequential propagation of any partial computation result. In the proposed algorithms, neither location nor distance information is needed. Only neighborhood information by exchanging "Hello" messages is necessary.

3 Model

A WSAN is represented as an undirected graph $G=(V,E)$. $V=S\cup A$, where S is the sensor set and A is the actor set. $E\subset(S\times S)\cup(S\times A)$ is the edge set for sensor-sensor and sensor-actor connections. There is no direct connection between any two actors. They are connected indirectly through other means (such as special channels).

Figure 2 shows several sample WSANs. Each sensor in G_1 is connected to one actor while each sensor in G_2 and G_3 is connected to two actors. A graph G is actor-connected if each sensor is connected to an actor through nodes in G. Note that an actor-connected WSAN does not imply that the whole graph is connected. For example, $G=G_2\cup G_3$ in Figure 2 is not connected, although it is actor-connected. Now suppose a subset $S^{'}$ of S is put to sleep (for energy saving). We denote $G^{'}=G[V-S^{'}]$, i.e., the network after removing $S^{'}$.

Definition 1 Given an actor-connected network G,

- $G^{'}$ is persistent actor-connected if it maintains the same actor-connectivity as G, i.e., if a sensor, sleep or active, is connected to an actor through nodes in G, then it is still connected to the actor through nodes in $G^{'}$.
- $G^{'}$ is at-least-one actor-connected if each sensor, sleep or active in G, is connected to at least one actor through nodes in $G^{'}$.

Note that all of the above conditions imply coverage of sleep nodes. That is, each sleep node has at least one neighbor that is an active sensor or an actor. Suppose sensors are densely deployed such that the given area is fully covered by sensors and actors[1]. Due to the coverage requirement, each sleeping node has at least one active

[1]For simplicity, we assume each actor also has sensing capability with the same sensing range as a sensor.

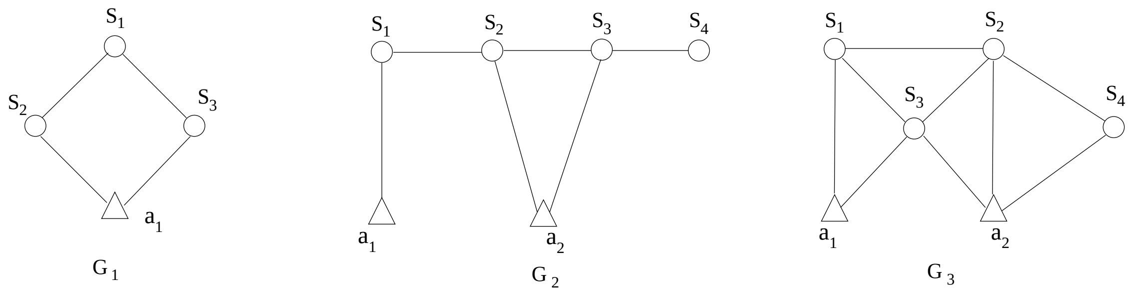


Figure 2: Sample WSNAs.

neighbor to cover it. Therefore, the set of active sensors and actors still cover the whole area approximately. Connectivity varies depending on the required degree: persistent or at-least-one.

In Figure 2 G_2, if $S^{'} = \{s_4\}$, $G_2^{'}$ is persistent actor-connected since all sensors are still connected (through active nodes) to both actors. If $S^{'} = \{s_1, s_2, s_4\}$, $G_2^{'}$ is no longer persistent but it is at-least-one actor-connected, since all sensors are connected to at least one actor (e.g., s_4 to a_2 via active sensor s_3). Note that traditional clustering approaches [8, 16] can be used to meet the coverage requirement, where only clusterheads are active. Such approaches, however, are not localized (there is information propagation). In addition, gateway nodes need to be selected to ensure connectivity among clusterheads. Clustering approaches will not be discussed further here.

Let us consider now a WSAN that is initially actor-connected. How can we remove some sensors (i.e., put them to sleep) while ensuring that the resultant graph is still at-least-one or persistent actor-connected? We consider local solutions in which each node uses neighborhood information to determine its status: active or sleep. Unlike traditional centralized or distributed solutions where some form of sequential propagation of information is required, i.e. broadcasting of link state in central solutions and propagation of node status (active/sleep) in distributed solutions, local solutions relay on h-hop neighborhood (for a small h such as 2 or 3) information without any other form of information propagation. h-hop neighborhood information can be obtained through periodic "Hello" message exchange or link state broadcast within h-hop.

4 Proposed Methods

Let us assume that each node is equipped with its h-hop neighborhood information (for $h = 2$ or 3). Also, each node s has a priority $p(s)$ and such a priority is totally ordered within its h-hop neighborhood. In addition, all actors have the same priority which is higher than any sensor priority. Let $p(a)$ be the actor priority. In Figure 2 G_3, 1-hop neighborhood of s_1 includes a_1, s_2, and s_3, but no connections among 1-hop neighbors.

2-hop neighborhood of s_2 covers the whole network.

Local rule for persistent actor-connectivity: *The default status of a sensor is active. A sensor u is in the sleep mode if, for any two of its neighbors w and v, w and v are connected by a path with all intermediate nodes (sensors or actors if any) having higher priorities than u.*

The above path is called a *replacement path* for node u. The intuition behind this rule is that a sensor u can be put to sleep if any two neighbors can be re-connected through nodes on a replacement path. Note that nodes on a replacement path can also be put to sleep. To avoid inconsistencies and a possible iterative process of putting sensors to sleep, a global priority is defined on each node. Note that if a sensor does not have two neighbors, then the replacement path condition is satisfied and the status of the sensor is sleep. The neighbor could be a sensor or an actor.

Suppose that in the Figure 2 the priority of sensors is the following: $p(a_1) = p(a_2) > p(s_1) > p(s_2) > p(s_3) > p(s_4)$. Using 2-hop neighborhood information, s_1 and s_3 are put to sleep in G_1 in persistent actor-connectivity; s_4 is in sleep in G_2 and s_3 and s_4 are in sleep in G_3.

Suppose $S^{'}$ is the set of sleeping sensors and $G^{'}$ is the subgraph after removing $S^{'}$. $V^{'}$ is the vertex set of $G^{'}$.

Theorem 1 Suppose $S^{'}$ is the set of sleeping sensors after applying the local rule for persistent actor-connectivity.

- (Coverage) For each sensor in $S^{'}$, there is a neighbor in $V^{'}$.
- (Connectivity) $G^{'}$ has the same actor-connectivity as G.

Proof: Suppose $S(a)$ is a subset of S connected to actor a in G. We show that $S(a)$ is still connected to a through nodes in $G^{'}$. We prove this by contradictions. Suppose W is a subset of $S(a)$ not connected to a. Note that nodes in W can be sleep or active. Let $U = N(W) - W$ be the sleep neighbors of W (see Figure 3) that are connected to a. $U \neq 0$ since W is connected to a in G. Let u be the node in U with the highest priority. From the assumption, u has two neighbors, w and v, from W and $V - U - W$, respectively. Any replacement path for u must contain at least one node $u^{'} \in U$. That contradicts the assumption that $p(u) > p(u^{'})$. Therefore, all nodes in $S(a)$ are still connected to a and all sleep nodes in W have neighbors that are active sensors or actors. $\square$

To provide a local rule for at-least-one actor-connectivity, we define an *extended replacement path* as follows:

1. it is regular replacement path for u connecting two neighbors w and v, or
2. w and v are each connected to an actor. These two actors can be distinct and all intermediate nodes in these two connections have higher priorities than u.

Local rule for at-least-one actor-connectivity: *The default status of a sensor is active. A sensor u is in sleep mode if for any two of its neighbors w and v, there exists an extended replacement path for u connecting w and v.*

The intuition behind the above rule is that sensor u can be put to sleep as long as any two neighbors can be either connected through a regular replacement path or each of them is connected to an actor.

In Figure 2, using 2-hop neighborhood information, s_3 is asleep in G_1 for at-least-one actor-connectivity; s_1, s_2, and s_4 are asleep in G_2 and all sensors are asleep in G_3.

Theorem 2 Suppose $S^{'}$ is the set of sleeping sensors after applying the local rule for at-least-one actor-connectivity.

- (Coverage) For each sensor in $S^{'}$, there is a neighbor in $V^{'}$.
- (Connectivity) Each node in $G^{'}$ is connected to at least one actor.

Proof: We use a similar proof as in Theorem 1. In our model, each node in S is connected to at least one actor. We show that each node in S is still connected to an actor through nodes in $G^{'}$. We prove by contradiction. Suppose W is the subset of S not connected to any actor. Let $U = N(W) - W$ be the sleep neighbors of W that are connected to an actor. Let u be the node in U with the highest priority. From the assumption, u has two neighbors, w and v, from W and $V - U - W$, respectively. Any replacement path for u must contain at least one node $u^{'} \in U$. Such a replacement path connects w via $u^{'}$ to either v or an actor directly as shown in Figure 3. That contradicts the assumption that $p(u) > p(u^{'})$. Therefore, all nodes in S are connected to actors and all sleep nodes in W have neighbors that are active sensors or actors. □

5 Extensions

In this section, we introduce two new notions of connectivity.

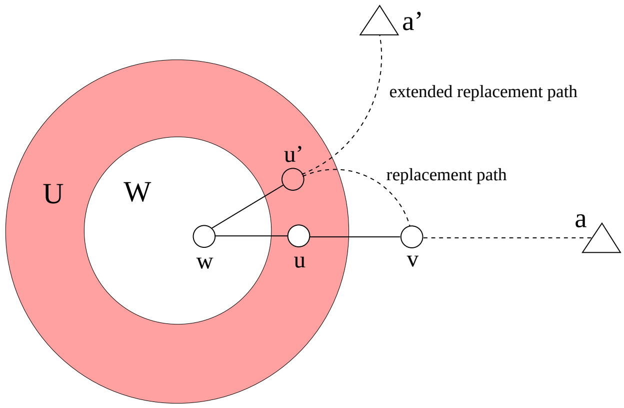


Figure 3: Illustration for the proof of Theorems 1 and 2.

Definition 2 A WSAN, G, is called weak k-actor-connected if each sensor is connected to k actors. A WSAN, G, is called strong k-actor-connected if each sensor is connected to at least one actor after removing any $k-1$ nodes (sensors or actors) from G.

Based on Definition 2, strong k-actor-connectivity implies weak k-actor-connectivity, i.e., connection to k actors. The k-actor-connectivity is used for reliability. Weak k-actor-connectivity can tolerate $k-1$ actor failures, while strong k-actor-connectivity can tolerate $k-1$ failures of any nodes, sensors and actors.

Figure 2 shows several sample WSANs. G_1 is 1-actor-connected, although each node has two node-disjoint paths to actor a_1. G_2 is weak 2-actor-connected but not strong 2-actor-connected. G_3 is strong 2-actor-connected. To simplify the notation, we use k-actor-connected for strong k-actor-connected.

We now consider maintaining k-actor-connectivity while putting some sensors into the sleep mode. Let G be a k-actor-connected network and $G' = G[V - S']$, where S' is a set of sleeping sensors.

Definition 3 Given a k-actor-connected network G,

- G' is persistent k-actor-connected if it maintains the same actor-connectivity as G after removing any $k-1$ nodes (sensors or actors).
- G' is at-least-one k-actor-connected if each sensor, sleep or active, in G is connected to at least one actor through nodes in G' after removing any $k-1$ nodes.

Again, the actor-connectivity means that if a sensor in G is connected to an actor through nodes in G, then this sensor (which might be in the sleeping mode) is still connected to the actor through nodes in G'. At-least-one k-actor-connected is the regular k-actor-connected and is simply called k-actor-connected, while persistent k-actor-connected requires a stronger condition. Here we use the general case of removing $k-1$ nodes, which includes both sensors and actors. In this case, the persistent connectivity means the existence of a path from G' to previously connected actor (before the removal of $k-1$ nodes) even if that actor has been removed. Next, we give local rules that ensure k-actor-connectivity. Two paths are called node-disjoint if they do not share any intermediate nodes.

Local rule for persistent k-actor-connectivity: *The default status of a sensor is active. A sensor u is in sleep mode if for any two of its neighbors w and v, there exists k node-disjoint replacement paths for u connecting w and v.*

Likewise, the at-least-one version of k-actor-connectivity also uses the extended replacement path.

Local rule for k-actor-connectivity: *The default status of a sensor is active. A sensor u is in sleep mode if for any two of its neighbors w and v, there exists k node-disjoint extended replacement paths for u connecting w and v.*

Note that both w and v can be actors and in this case there is no connection of intermediate nodes. Also, the actor can not be shared in two extended replacement paths. In Figure 2 G_3, s_4 is put to sleep based on local rule for persistent 2-actor-connectivity. That is, even if a node is removed arbitrarily from G_3, all sensors are still connected to both actors. For example, when s_2 is removed, s_4 is connected to a_1 via a_2. Sensors s_3 and s_4 are put to sleep from G_3 for 2-actor-connectivity. In this case, s_1 and s_2 in G_3 are both 2-actor-connected after making s_3 and s_4 sleep. s_3 is put to sleep by checking all neighbor pairs, for which each has 2 node-disjoint paths. For example, for neighbors s_1 and s_2 of s_3, one path is (s_1, s_2) and the other is from s_1 to a_1 and from s_2 to a_2.

The correctness of these two local rules in preserving k-actor-connectivity will be discussed in the next section.

6 Properties

In this section, we relate k-actor-connectivity to k-connectivity in regular graphs. Then, we show the correctness of two local rules for k-actor-connectivity.

We first construct a k-connected graph by treating all actors in A as regular nodes. These actors are connected by a complete bipartite graph $\bar{G}$ with node set $A \cup A'$, where $|A| = |A'|$ and each node in A is connected

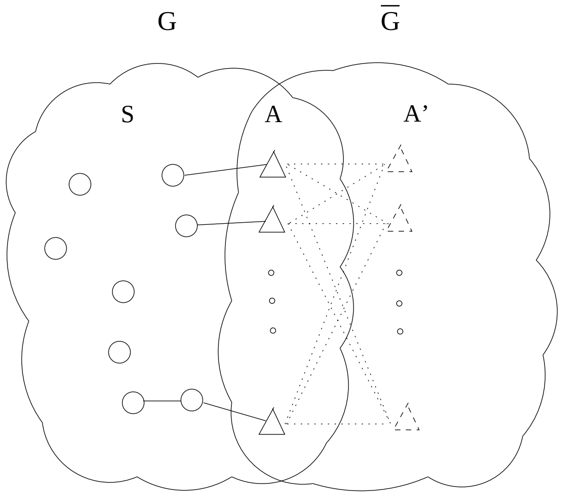


Figure 4: A k-connected graph $G \cup \bar{G}$.

to each node in $A^{'}$. There is no direct connection among nodes in A (and among nodes in $A^{'}$). Note that $A^{'}$ is a set of virtual nodes. Now we first show that $G \cup \bar{G}$ (shown in Figure 4) is k-connected.

Theorem 3 If G is k-actor-connected, then $G \cup \bar{G}$ is k-connected.

Proof: Based on the definition of k-actor-connectivity, we can see that $|A| \geq k$. We arbitrarily select two nodes s and d from $G \cup \bar{G}$, and we have the following three cases:

1. If both s and d are in $A \cup A^{'}$, they are clearly connected after removing $k-1$ nodes, since $\bar{G}$ is a complete bipartite graph with $|A| = |A^{'}| \geq k$.
2. If one is in S and the other in $A \cup A^{'}$, based on the definition of k-actor-connectivity, the one in S is connected to at least one node in A after removing $k-1$ nodes from $G \cup \bar{G}$, which in turn is connected to any node in $A \cup A^{'}$, including d.
3. If both are in S, suppose one is connected to a node a in A and the other is connected to a node $a^{'}$ in A after removing any $k-1$ nodes from $G \cup \bar{G}$. Based on the construction of $\bar{G}$, nodes a and $a^{'}$ are still connected.

Therefore, $G \cup \bar{G}$ based on the definition is k-connected. □

Now we show that the two local rules in the previous section preserve k-actor connectivity.

Theorem 4 Given a k-actor-connected graph G, the graph $G^{'}$ derived using the local rule for (persistent) k-actor-connectivity is (persistent) k-actor-connected.

Proof: Suppose G is the original k-actor-connected graph. Now we arbitrarily remove $k-1$ nodes from G and obtain G^F. By relating k-actor-connectivity to k-connectivity (Theorem 3) and the Menger's theorem [10], G^F still preserves the same actor-connectivity as G. Based on these two rules for k-actor-connectivity, each sleep node in G has k node-disjoint replacement or extended replacement paths for any pair of neighbors. Removing any $k-1$ nodes will leave at least one replacement or extended replacement path. That is, a sleep node using rules for k-actor-connectivity in G will still be a sleeping node using corresponding rules for actor-connectivity in G^F. That is, $G^{'}$ (obtained by applying local rules for k-actor-connectivity on G) has at least the same degree of actor-connectivity as $(G^F)'$ (obtained by applying the corresponding local rules for actor-connectivity on G^F). The rest of the proof follows by applying Theorems 1 and 2 to $(G^F)'$, which shows the relevant actor-connectivity. □

The following are two more properties: one relates k-actor-connectivity to node-disjoint paths to k actors and the other to node-disjoint paths to actors after applying local rules.

Property 1 If G is k-actor-connected, then each node in S has node-disjoint paths to at least k nodes in A.

We can use the following argument to prove this property. Suppose we have a sensor s in S and the other node d in $A^{'}$. Menger's theorem states that in a k-connected graph there are k node-disjoint paths between any two nodes. Using this property, we have k node-disjoint paths between s and d. Among these paths, all neighbors of d are distinct actors in A. Therefore, any node in S has node-disjoint paths to at least k distinct actors.

Suppose $S^{'}$ is the subset of S that is removed (put to sleep) after applying the local rule for (persistent) k-actor-connectivity and again $G^{'} = G[V - S^{'}]$. We have the following result.

Property 2 $G^{'} \cup \bar{G}$ is still a k-connected graph and each node in $S - S^{'}$ has node-disjoint paths to at least k nodes in A.

Based on Theorem 4, the local rule for (persistent) k-actor-connectivity ensures that $G^{'}$ is still k-actor-connected. Based on Theorem 3, we have $G^{'} \cup \bar{G}$ as a k-connected graph. The second part follows directly from Theorem 1. Therefore, the above property holds.

7 Implementation Issues

We discuss several issues related to implementation, including actor-initiated dynamic implementation solutions which will be used for baseline comparison in simulation.

7.1 Selection of priority

We assume that node priorities within h-hop are distinct. In the actual implementation, this condition can be relaxed. That is, nodes within h-hop can have the same priority. This will not cause errors because a node can go to sleep only if any two neighbors are connected by other k paths with "higher" priorities. However, it will affect the efficiency of the algorithm. Because in some cases, two nodes can cover each other's neighbors, but neither can go to sleep due to their identical priority.

A natural choice for node priority is node ID, although other metrics can be used, such as energy level. In this way, sensors can rotate their roles (active/sleep) to balance energy consumption. The energy consumption of each sensor is then balanced in the long term.

7.2 Controlling the path length

In some real-time applications, it is vital for a detecting event to reach the corresponding actor(s) within certain time limits. To avoid having too many sensors on short paths (from the sensor to the actor), we can restrict the length of each replacement path for each sleeping sensor. For example, we can set each replacement path to be bounded by h hops, then globally the shortest path length of each sensor to an actor can be controlled.

7.3 Static vs. dynamic implementation

Local rules can be implemented in a static or dynamic way. In static implementation, each node determines its status based on its h-hop information. In dynamic implementation, each node acts on a message originated from an actor. In such a message, the actor ID or even path information from the original actor to the current node can be piggybacked to assist the status determination of each node. The actor ID indicates the connectivity of a neighbor to a particular actor, even though the actor might be outside h-hop neighborhood. Likewise, path information to an actor can be used for the local rule for k-actor-connectivity. Efficient reduction of active nodes is possible by judiciously selecting an appropriate time-out after receiving the first message at each sensor to gather more path information from actors.

If we allow propagation of node status, an active node can be treated as an actor which is useful for at-least-one actor-connectivity. Note that dynamic implementation resembles distributed implementation which has several simple implementations for at-least-one and persistent actor-connectivity.

7.4 Actor-initiated dynamic implementation

We can use actor-initiated dynamic implementation for two simple cases: at-least-one and persistent actor-connectivity. The direct distributed implementation for at-least-one and persistent k-actor-connectivity are much more involved, since path information needs to be propagated. Note that all dynamic implementations are not strictly local solutions with information propagation. However, they are used as baseline cases for comparison.

At-least-one actor-connectivity:

1. Each actor sends out an invitation message.
2. Each sensor responds to the first invitation only and forwards the invitation to its neighbors.
3. Sensors receiving responses are active and sensors not receiving responses are put to sleep.

Although at-least-one actor-connectivity is simple, it does need some form of information propagation (in this case, an invitation). The number of invitation messages is equivalent to the number of sensors. The distributed implementation for persistent actor-connectivity is much more involved in terms of message complexity: it is the total number of actor-sensor connectivity.

Persistent actor-connectivity:

1. Each actor sends out an invitation message with its ID.
2. Each sensor responds to the first invitation for each ID and forwards the invitation to its neighbors.
3. Sensors receiving responses are active and sensors not receiving responses are put to sleep.

Maintaining k-actor-connectivity is much more involved as each active sensor needs to ensure the existence of node-disjoint paths to k distinct actors. The complete path information needs to be propagated in the network, generating excessive traffic. Hence, we will not discuss this approach further.

Table 1: Simulation parameters.

Network Area	100×100
Transmission Range	25
Number of Sensors	n, 50 to 300
Number of Actors	m, 2 to 8
Number of Hops	h, 2 to 4
Connectivity Requirement	k, 1 to 6
Number of Trials	100

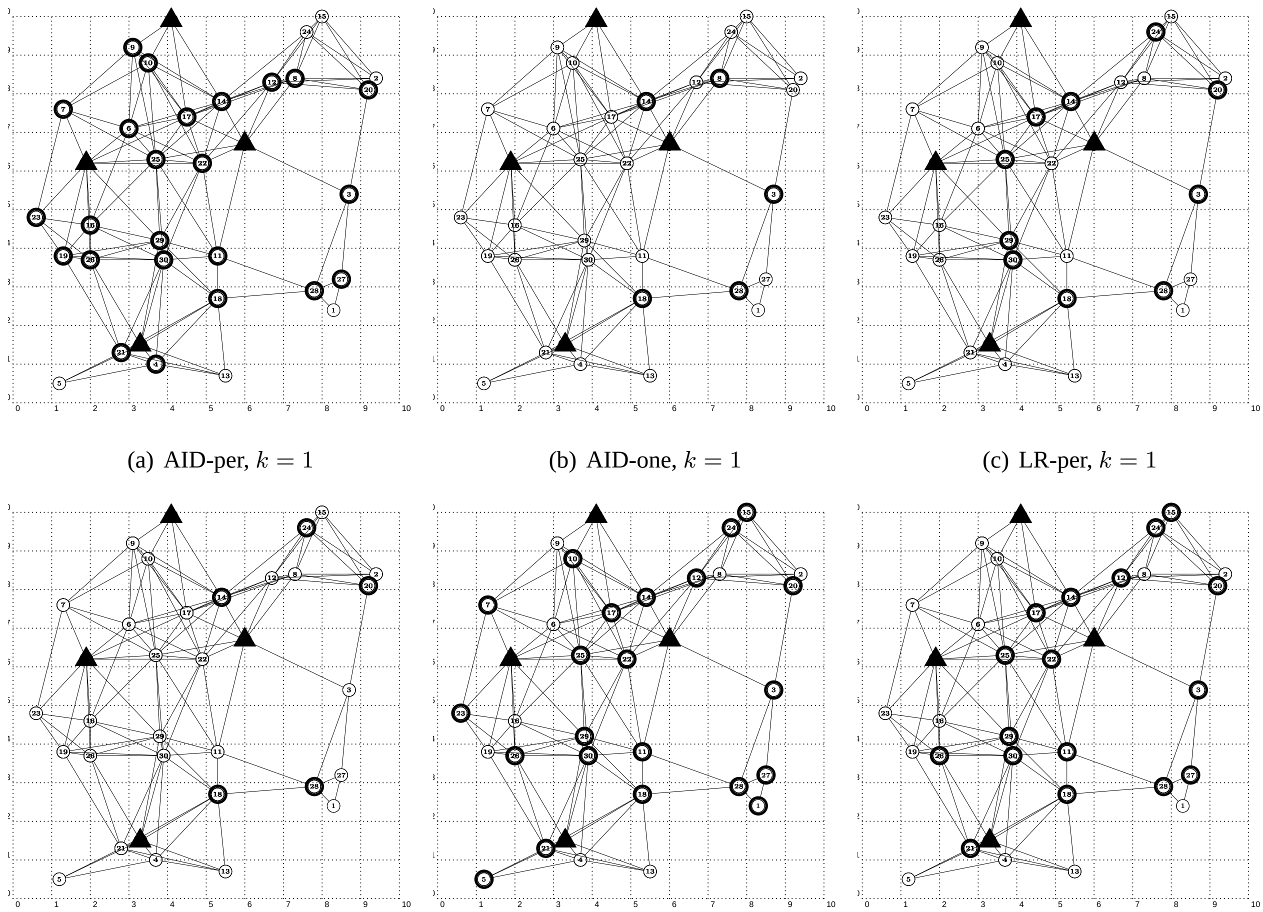
8 Simulations

We evaluate the proposed two algorithms, Local Rule for persistent k-actor-connectivity (LR-per) and Local Rule for at-least-one k-actor-connectivity (LR-one) with different system parameters. We also simulate the Actor-Initiated Dynamic implementation for persistent actor-connectivity (AID-per) and at-least-one actor-connectivity (AID-one) to compare with the proposed local algorithms.

8.1 Simulation Environment

All algorithms have been implemented on a custom simulator. All simulations are conducted in randomly generated static networks. To generate a network, n sensors and m actors are randomly placed in a 100×100 area. The transmission range is 25. Any two sensors, or a sensor and an actor, with distance less than 25 are considered neighbors. Networks that cannot form a k-actor-connected graph are discarded. The tunable parameters in the simulation are as follows. (1) The number of sensors n. We vary the number of deployed nodes from 50 to 300 to check the scalability of the algorithms. (2) The number of actors m. We vary m from 2 to 8. (3) The connectivity requirement k. We use 1 to 6 as the value of k. In each simulation, $k \leq m$. (4) The number of hops, h. The local algorithms use 2-hop neighborhood information in most of the simulations. We also increase h to 3 and 4 to see the effect. When a node collects h-hops information, it gets the network topology within its h-hops neighborhood except for the links between any two h-hop away nodes. For each tunable parameter, the simulation is repeated 100 times or until the confidence interval is sufficiently small ($\pm 1\%$, for the confidence level of 90%). Table 1 lists the simulation parameters.

The following performance metrics are evaluated. (1) *Active nodes*. The number of active sensors, which represents the energy consumption. (2) *Path ratio*. The ratio of the average length of the routing paths in G' to that in G, which represents the routing latency of the system.



(a) AID-per, $k = 1$ (b) AID-one, $k = 1$ (c) LR-per, $k = 1$

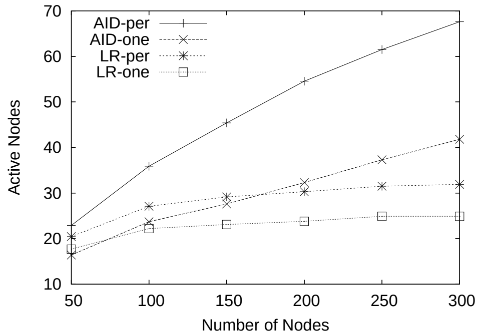
(d) LR-one, $k = 1$ (e) LR-per, $k = 2$ (f) LR-one, $k = 2$

Figure 5: Examples of AID-per, AID-one, LR-per, and LR-one ($n = 30, m = 4$).

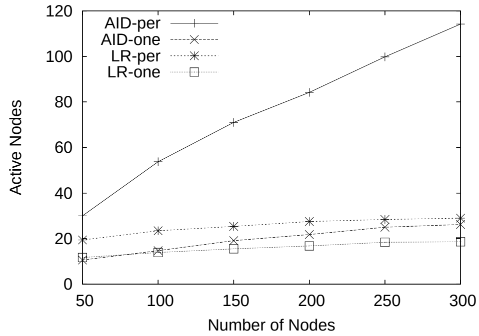
Figure 5 shows the selected active node set in a sample network. There are 30 sensors (shown as circles) and 4 actors (shown as triangles). The active sensors are shown by bold circles, and the numbers in the sensors are the IDs. (a) and (b) are the results of AID-per and AID-one, respectively, with $k = 1$. There are 28 and 9 active nodes, respectively. (c) and (d) are of LR-per and LR-one when $k = 1$. There are 14 and 9 active nodes, respectively. (e) and (f) are of LR-per and LR-one when $k = 2$. There are 26 and 21 active nodes respectively.

8.2 Simulation Results

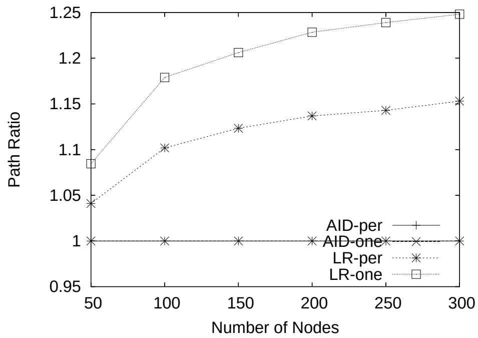
Figure 6 shows the comparison of AID and LR with $k = 1$. (a) and (c) are the results of the number of active nodes and ratio of length of the routing paths with $m = 2$, respectively. (b) and (d) show the results for $m = 6$.



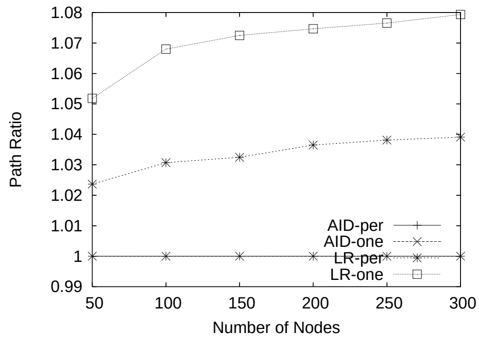
(a) Active nodes ($m = 2$)



(b) Active nodes ($m = 6$)



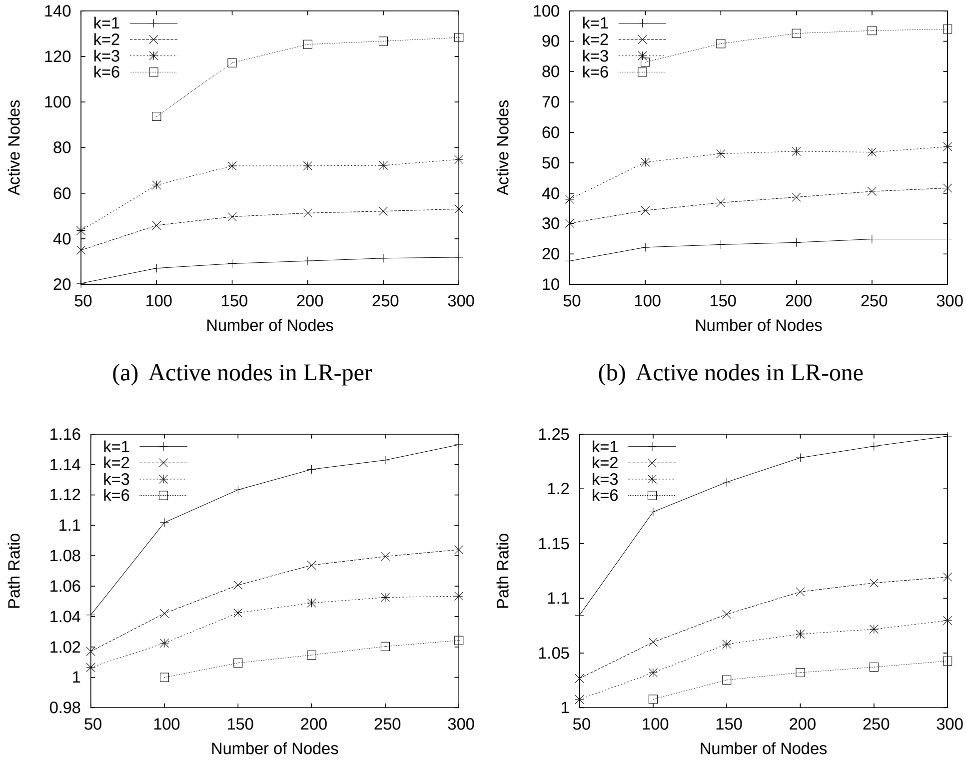
(c) Path ratio ($m = 2$)



(d) Path ratio ($m = 6$)

Figure 6: Comparison of AID-per, AID-one, LR-per, and LR-one ($k = 1$).

In (a) we can see that methods for persistent connectivity have larger numbers of active nodes than those for at-least-one connectivity, which is obvious since more nodes need to be selected in order to keep each node connected to every actor instead of one actor. AID-per selects more active nodes than LR-per. AID-one has less active nodes than LR-one only when the number of sensors is very small (smaller than 75). In (b), when $m = 6$, the comparison results of the four algorithms remain the same with those in (a). However, AID-per has more active nodes with larger m while the other three tend to have less active nodes. This is because in AID-per each node needs to keep a shortest path to every actor with all the nodes on the path being active. Obviously, the greater the number of actors, the more active nodes there will be. In AID-one, a shortest path to an actor is needed for each node, and more actors help to reduce the length of this shortest path. Thus, less active nodes are necessary. For the LR-per and LR-one, since actors are viewed as nodes with the highest priorities, more actors lead to higher probability of (extended) replacement path and hence the non-active nodes. Therefore, less active nodes are selected. In both (a) and (b), more deployed sensor nodes lead to increased number of active nodes. However, the increasing tends to stop when the node density reaches a certain degree in AID-one, LR-per, and LR-one.



(a) Active nodes in LR-per

(b) Active nodes in LR-one

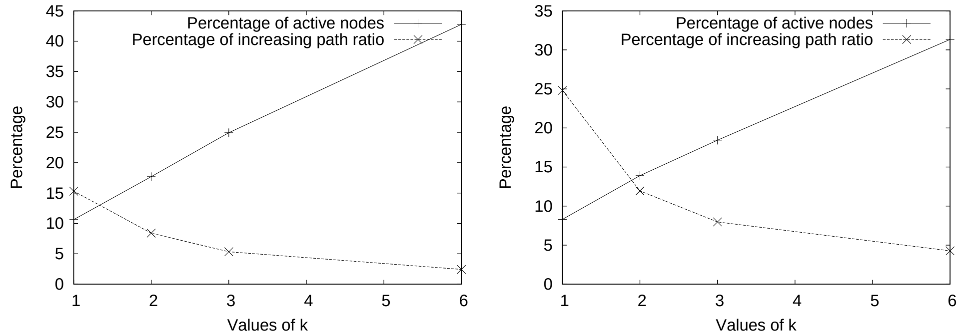
(c) Path ratio in LR-per

(d) Path ratio in LR-one

Figure 7: Performance of LR-per/LR-one with different k ($m = 6, h = 2$).

(c) and (d) are the ratio of the length of the routing path in original graph to that in the resultant graph (via only active nodes). The routing path is the shortest path from a sensor to a nearest actor, and the length is in terms of hop count. Since AID-per and AID-one always keep the nodes on the shortest path from a sensor to a nearest actor, the ratio is always 1. LR-per has smaller ratio than LR-one due to its larger active node set. Comparing (c) with (d) we can see that larger m results in smaller ratio. Both in (c) and (d), more deployed sensor nodes lead to an increased ratio. The increasing tends to stop when the node density reaches a certain degree.

Figure 7 shows the performance analysis of LR-per and LR-one in terms of parameter k with $m = 6$. (a) and (b) are the sizes of the resultant active node sets of LR-per and LR-one, respectively. (c) and (d) are their corresponding path ratios. k is increased from 1 to 6 in these figures. We can see that when k is larger, more active nodes are necessary to achieve higher connectivity. However, compared with that of the deployed sensors, the increasing of the number of active nodes is slight. Also, LR-one needs less active nodes than LR-per, which is consistent with the previous simulation results. (c) and (d) show that larger k helps with a



(a) LR-per (b) LR-one

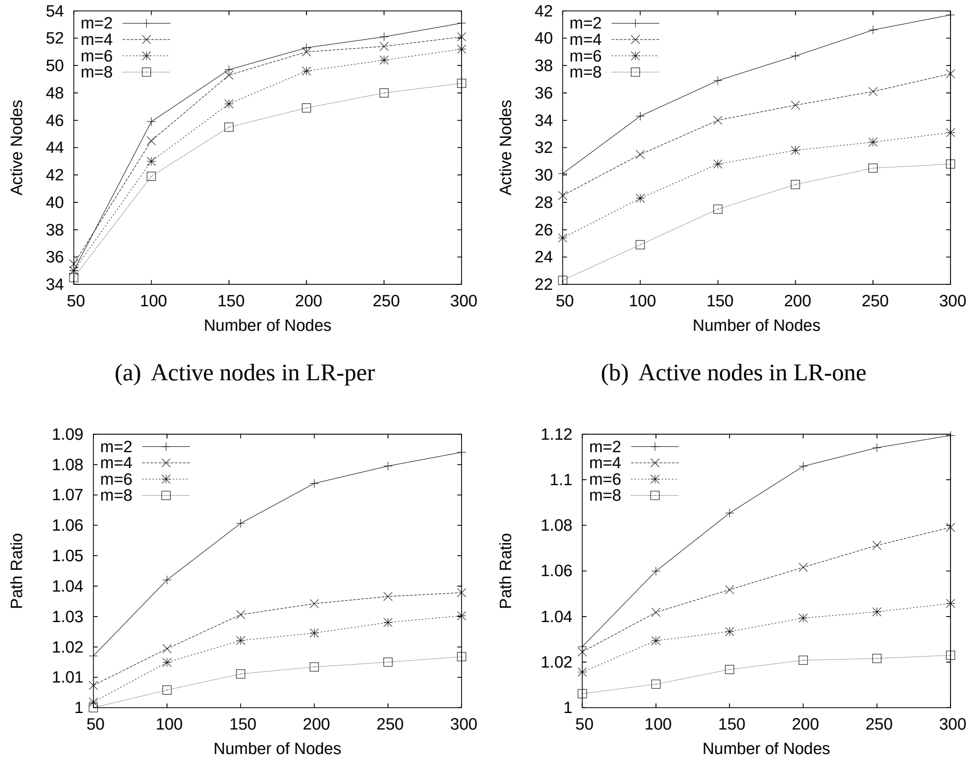
Figure 8: Percentages of active nodes and the increasing of path ratio with different k ($n = 300, m = 6, h = 2$).

smaller path ratio due to more active nodes. However, this decrease of path ratio tends to stop when k is large enough. We can see that when k is 6, the ratio is slightly larger than 1 (less than 1.05). There is little room to further reduce the ratio by increasing k. When the number of deployed nodes is 50, the original graph is hardly 6-actor-connected, thus there are no simulation results in the figures when n is 50 and k is 6.

Figure 8 is generated from Figure 7 to show the percentages of active nodes and the increasing of path ratio in LR-per and LR-one with the increasing of k. We can see that when the value of k decreases, less active nodes are needed in both LR-per and LR-one, and the path ratios are increased as well. However, the increasing of the path ratio is insignificant compared to the decreasing of the number of active nodes until k is small enough, that is, k is decreased to 1. Therefore, most of the time, the decrease of the number of the active nodes in the network will not lead to the significant increase of the path ratio.

Figure 9 shows the results of LR-per and LR-one with different numbers of actors where k is fixed as 2. (a) and (c) are number of active nodes and path ratio in LR-per and (b) and (d) are those of LR-one. From (a) and (b) we can see that larger m results in a smaller number of active nodes. A decreasing of the number of active nodes caused by the increasing of the value of m is more significant in LR-one than in LR-per. This is because in LR-per, although more actors do provide higher probability for (extended) replacement path and hence non-active node status, it also lead to the requirement of an increased connectivity. (c) and (d) show that more actors result in a smaller path ratio. The path ratio of LR-per is smaller than that of LR-one due to its larger size of active nodes. Both the number of active nodes and path ratio increase with the growth of the network density, but tend to get saturated when the density reaches a certain degree under all circumstances. Therefore, the proposed algorithms scale well.

Figure 10 shows the performance of LR-per and LR-one with different h. (a) and (b) are the numbers



(a) Active nodes in LR-per

(b) Active nodes in LR-one

(c) Path ratio in LR-per

(d) Path ratio in LR-one

Figure 9: Performance of LR-per/LR-one with different m ($k = 2, h = 2$).

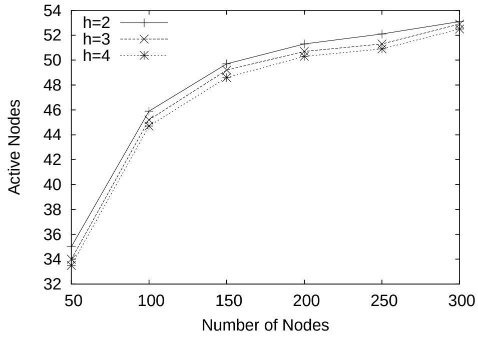
of active nodes in these two algorithms, and (c) and (d) are the path ratios of them. We can see that with more neighborhood information, less active nodes are necessary in both algorithms. However, this increase is relatively slight. When h is 4, a node can achieve almost the entire network topology, the performance can not be increased significantly. Therefore, in application a small value of h is enough. The path ratios are decreased with the growth of the value of h as in (c) and (d). However, the increase of the performance is not significant.

Simulation results can be summarized as follows:

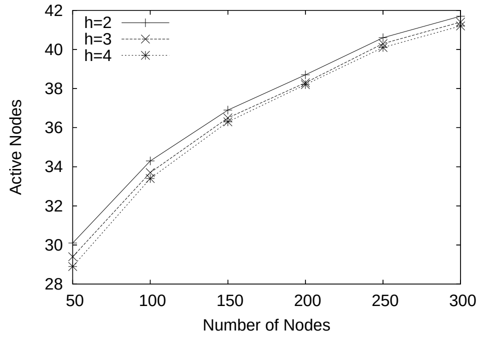
1. LR-per has less active nodes than that of AID-per; AID-one has smaller number of active nodes than LR-one. But AID-per and AID-one are not localized approaches.

2. Although the path ratios of LR-per and LR-one is not 1 (as in AID-per and AID-one), they are not significantly higher than 1, and can be controlled by the value of m.

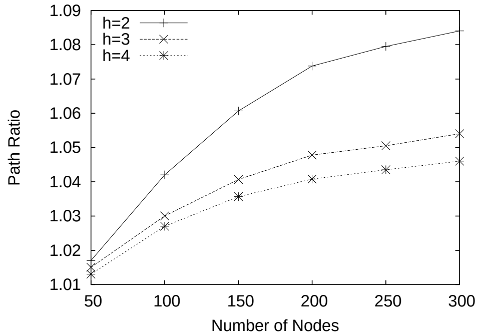
3. When m is fixed, larger k leads to larger active node set and smaller path ratio in LR-per and LR-one.



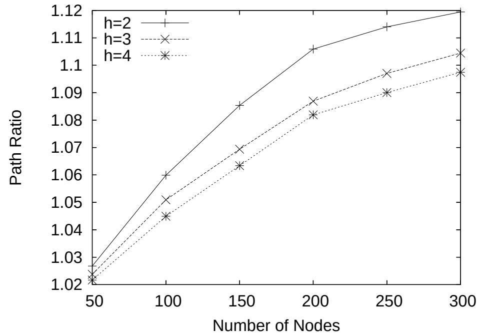
(a) Active nodes in LR-per



(b) Active nodes in LR-one



(c) Path ratio in LR-per



(d) Path ratio in LR-one

Figure 10: Performance of LR-per/LR-one with different h ($k = 2, m = 2$).

4. When k is fixed, larger m helps to reduce the number of active nodes and also path ratio. However, in LR-per the decreasing of number of active nodes by the increasing of the value of $m$is insignificant.

5. The increase of the length of routing path in both LR-per and LR-one is insignificant. Therefore, selecting only a subset of nodes to be active introduces very little data-routing latency.

6. More local neighborhood information results in better performance in terms of both the number of active nodes and path ratio for both algorithms. However, a relatively small value, say 3, of h is enough to avoid overhead.

7. Under all circumstances, with the growth of the number of deployed nodes, the number of active nodes and path ratio increase. However, there is a much smaller increasing ratio and the path ratio tends to get saturated. Therefore, all the proposed algorithms scale well.

9 Conclusion

In this paper, we defined several sensor-actor connection requirements in wireless sensor and actor networks, the persistent actor-connectivity and at-least-one actor-connectivity. We proposed several local solutions to ensure different connection requirements, where each node makes its decision (on its active and sleep mode) purely based on local information and there is no information propagation during the decision process. We also looked at several fault-tolerance extensions, the persistent or at-least-one k-actor connectivity, in which the network is still ensured connectivity in the presence of sensor or actor failures. We also proved the relationship between the regular k vertex connectivity in graph theory and the proposed k-actor connectivity in the WSANs. Simulation results show that LR-per and LR-one can both generate an efficient active node set, saving energy consumption by putting other nodes into sleep states without introducing much routing delay. In the future, we will develop a complete routing scheme based on the proposed connectivity, design a detailed energy consumption model for the WSANs, and compare with other existing energy-efficient routing protocols.

References

[1] I. F. Akyildiz and I. H. Kasimoglu. Wireless sensor and actor networks: research challenges. *Ad Hoc Networks Journal (Elsevier)*, (4):351–367, 2004.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, (4):393–422, 2002.

[3] S. Bandyopadhyay and E. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *Proc. of IEEE Infocom*, 2003.

[4] M. Cardei, J. Wu, and S. Yang. Fault-tolerant topology control for heterogeneous wireless sensor networks. *submitted for publication*.

[5] J. Carle and D. Simplot-Ryl. Energy efficient area monitoring by sensor networks. *IEEE Computer*, (2):40–46, 2004.

[6] E. Cayirci, T. Coplu, and O. Emirogl. Power aware many to many routing in wireless sensor and actuator networks. In *Proc. of Ewsn*, 2005.

[7] A. Cerpa, J. Elson, and M. Hamilton. Habitat monitoring: Application driver for wireless communication technology. In *Proc. of the ACM SIGCOM Workshop on Data Communication, in conjunct with MobiCom'01*, 2001.

[8] A. Ephremides, J. E. Wieselthier, and D. J. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceedings of the IEEE*, (1):56–73, 1987.

[9] B. P. Gerkey and M. J. Mataric. A market-based formulation of sensor-actuator network coordination. In *Proc. of the AAAI Spring Symposium on Intelligent Embedded and Distributed Systems*, 2002.

[10] A. Gibbons. Algorithmic graph theory. *Cambridge University Press*, 1985.

[11] M. Haenggi. Mobile sensor-actuator networks: opportunities and challenges. In *Proc. of 7th IEEE Internatinal Workshop*, 2002.

[12] W. Hu, N. Bulusu, and S. Jha. A communication paradigm for hybrid sensor-actuator networks. *International Journal of Wireless Information Networks*, (1):47–59, 2005.

[13] W. Hu, C. Chou, S. Jha, and N. Bulusu. Deploying long-lived and cost-effective hybrid sensor networks. In *Proc. of the 1st Workshop on Broadband Advanced Sensor Networks (BaseNets 2004)*, 2004.

[14] C. Intanagonwiwat, R. Govindan, D. Estrin, and J. Heidemann. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, (1):2C16, 2003.

[15] R. Kumar, V. Tsiatsis, and M. B. Srivastava. Computation hierarchy for in-network processing. In *Proc. of the 2nd ACM International Conference on Wireless Sensor Networks and Applications*, 2003.

[16] C. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, pages 151–162, 1999.

[17] T. Melodia, D. Popili, V. C. Gungor, and I. F. Akyildiz. A distributed coordination framework for wireless sensor and actor networks. In *Proc. of MobiHoc*, 2005.

[18] M. F. Munir and F. Filali. A novel self organizing framework for SANETs. In *Proc. of the 12th EW Conference*, 2006.

[19] V. Paruchuri and A. Durresi. Delay-energy aware routing protocol for sensor and actor networks. In *Proc. of ICPADS*, 2005.

[20] E. M. Petriu, N. D. Georganas, D. C. Petriu, D. Makrakis, and V. Z. Groza. Sensor-based information appliances. *IEEE Instrumentation and Measurement Magazine*, (4):31–35, 2000.

[21] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proc. of IEEE Infocom*, 2002.