A Kautz-based Real-time and Energy-Efficient Wireless Sensor and Actuator Network

Ze Li and Haiying Shen
Department of Electrical and Computer Engineering
Clemson University, Clemson, SC 29631
Email: {zel, shenh}@clemson.edu

***Abstract*—Wireless Sensor and Actuator Networks (WSANs) are composed of sensors and actuators to perform distributed sensing and actuating tasks. Most WSAN applications (e.g., fire detection) demand that actuators rapidly respond to events under observation. Therefore, real-time and fault-tolerant transmission is a critical requirement in WSANs to enable sensed data to reach actuators reliably and quickly. Due to limited power resources, energy-efficiency is another crucial requirement. Such requirements become formidably challenging in large-scale WSANs. However, existing WSANs fall short in meeting these requirements. To this end, we first theoretically study the Kautz graph for its applicability in WSANs to meet these requirements. We then propose a Kautz-based REal-time, Fault-tolerant and EneRgy-efficient WSAN (REFER). REFER has a protocol that embeds Kautz graphs into the physical topology of a WSAN for real-time communication and connects the graphs using Distributed Hash Table (DHT) for high scalability. We also theoretically study routing paths in the Kautz graph, based on which we develop an efficient fault-tolerant routing protocol. It enables a relay node to quickly and efficiently identify the next shortest path from itself to the destination only based on node IDs upon routing failure. REFER is advantageous over previous Kautz graph based works in that it does not need an energy-consuming protocol to find the next shortest path and it can maintain the consistency between the overlay and physical topology. Experimental results demonstrate the superior performance of REFER in comparison with existing systems in terms of real-time communication, energy-efficiency, fault-tolerance and scalability.**

I. Introduction

Wireless sensor networks (WSNs) is a collection of low-cost, low-power, and multi-functionality wireless sensing devices that can be densely deployed for surveillance purpose. Traditionally, it is a data gathering network where are responsible only for sampling their surroundings and reporting to predefined data sinks. As hardware technology advances, it is now evolving toward service-oriented wireless sensor and actor networks [1]. Wireless Sensor and Actuator Networks (WSANs) consist of sensor nodes capable of measuring stimuli in the environment and actuator nodes capable of affecting their local environment. Similar to sensor networks, WSAN sensors usually are low-cost and low-power devices with a short transmission range that are used for the sensing of a physical phenomena. WSAN actuators are resource-rich devices characterized by higher processing and transmission capabilities along with a longer battery life [1]. When sensors detect events, they process and transmit the event data to their nearby actuators, which take action on the events. WSANs can potentially be used in applications such as real-time target tracking and surveillance, homeland security, chemical attack detection [2] and environment monitoring in battlefields, factories, buildings and cities [3]. For example, smoke detectors (i.e., sensors) are widely and densely deployed in a building to report a fire event to the sprinklers (i.e., actuators) [4]. Similarly, in a battlefield, sensors are widely and densely deployed in fields to report the detection of malicious objects to nearby actuators, which immediately takes action to catch the objects [1]. Since sensors are usually deployed with high density to ensure the coverage and topology connectivity [5], the scenario we considered in this paper is a highly dense and mobile WSAN which consists of densely populated and possibly mobile sensors.

The primary function of WSANs is to enable the actuators to quickly and reliably respond to nearby sensed events. Delay in the actuators' response can lead to disastrous consequences such as a large loss of life. Therefore, *real-time* communication is of great importance in guaranteeing the timely actions. Because of node mobility and resultant routing failures, *fault-tolerance* is crucial to ensure reliable node communication. In addition, *energy-efficiency* is also a critical requirement for WSANs due to limited resources of sensors. Such requirements become formidably challenging in large-scale WSANs (e.g., battlefield monitoring applications) where the number of sensors is in the order of hundreds or thousands [6].

Most of the routing protocols for mobile ad hoc networks (MANETs) and wireless sensor networks (WSNs) treat every node equally and fail to leverage the capabilities of resource-rich devices to reduce the communication burden on low-resource sensors. These protocols are suboptimal for WSANs, if not entirely applicable. Recently, mesh-based [7, 8] and tree-based [2, 9] systems have been proposed for data transmission in WSANs. In the mesh-based methods, physically close sensors form a cluster and the selected cluster head reports events sensed by the cluster sensors to the closest actuator through a multi-hop path. In the tree-based methods, physically close sensors form a tree for data transmission. In both methods, a source node retransmits a message upon a routing failure. Also, both methods employ either geographical routing [10] or topological routing [12, 13]. These routing algorithms consume large amounts of energy because the former relies on position information generated by GPS or a virtual coordination method [11, 14, 15], and the latter relies

on flooding to discover and update routing paths. Therefore, mesh-based and tree-based systems cannot achieve real-time, energy-efficient, fault-tolerant transmissions at the same time in a highly dense and mobile WSAN.

To meet the requirements of WSAN applications, we propose a Kautz-based REal-time, Fault-tolerant and EneRgy-efficient WSAN (REFER). The contributions of this work are:

(1) *A theoretical study of the Kautz graph* for its applicability in WSANs to meet the energy-efficiency and real-time communication requirements in overlay maintenance and routing.
(2) *A Kautz graph embedding protocol* that embeds Kautz graphs to the physical topology of a WSAN and connects the graphs using Distributed Hash Table (DHT) [16] for high scalability and real-time communication.
(3) *A theoretical study of routing paths in the Kautz graph and an efficient fault-tolerant routing protocol* to support fault-tolerant, real-time and energy-efficient data transmission. The algorithm enables a relay node to quickly and efficiently identify the next shortest path from itself to the destination upon routing failure.
(4) *Extensive experiments to* demonstrate the superior performance of REFER in comparison with a tree-based, a mesh-based, and another Kautz-based WSAN.

REFER is advantageous over other Kautz-based works in two aspects. First, REFER is the first work that embeds a Kautz graph into the physical topology of a MANET to keep topology consistency. Previous works on Kautz graphs directly build a Kautz graph overlay on the application layer in peer-to-peer (P2P) networks [17–19] or MANETs [20]. Thus, the overlay is not consistent with the underlying physical topology and multi-hop routing must be used for the communication between two neighboring Kautz nodes in MANETs. This cannot provide fault-tolerance, energy-efficiency and real-time performance. Second, REFER can quickly and efficiently identify the alternative paths and their lengths simply based on node IDs upon a routing failure. In comparison, previous method [21] depends on an energy-consuming routing generation algorithm to find the alternative paths and their lengths.

II. Related Work

WSNs can be regarded as a subcategory of MANETs with the additional constraints of security and energy-efficiency. WSANs are a subcategory of WSNs with higher requirements on real-time, energy-efficient and fault-tolerant transmission. MANETs use either topological routing [12, 13] or geographic routing [10]. In topological routing, a source node broadcasts a query to find a path to the destination. Unfortunately, the limited battery power of the sensors makes such routing unsuitable for WSANs. Geographical routing always chooses the node closest to the destination by relying on the position information generated by GPS or a virtual coordination method [11, 14, 15], both consume a considerable amount of energy, which are also not suitable for WSANs.

To improve the real-time communication in WSNs. Hu *et al.* [22] proposed SPEED that uses a combination of feedback control and non-deterministic quality-of-service (QoS)-aware geographic routing to support real-time communication. Felemban *et al.* [23] proposed a multi-path and multi-speed routing protocol for a probabilistic QoS guarantee. Lu *et al.* [24] proposed a real-time communication architecture and a velocity monotonic packet scheduling policy to deal with the delay and distance constraints in transmission. Ye *et al.* [25] proposed a Two-Tier Data Dissemination approach (TTDD) for sensor data transmission. TTDD uses a grid structure so that only sensors located at grid points need to acquire the forwarding information. All other sensors only need to contact the sensor at grid points for the data transmission.

Also, a number of routing protocols have been proposed specifically for WSANs. Melodia *et al.* [2] proposed DaTree in which one actuator (tree root) and its physically close sensors form a tree. Each sensor forwards its detected events to its tree root. DaTree employs geographical routing. Hu *et al.* [9] proposed to build an anytree with leaves as actuators. Each source node builds an anycast tree and sends its detected data along the tree to the actuators. Anytree uses topological routing. The tree structure is not fault-resilient to node mobility since a parent failure will prevent its children from sending or receiving data in time. Ngai *et al.* [7] and Shah *et al.* [8] proposed a distributed protocol to form sensors into clusters. The cluster heads form a backbone mesh network to provide routes toward actuators. These methods need to retransmit a message from the source to the destination upon a routing failure, generating a certain delay. Also, they consume a considerable amount of energy due to their flooding-based topological or geographical routing components. REFER is superior to the previous WSAN routing protocols because it can simultaneously meet the requirements of real-time communication, fault-tolerance and energy-efficiency.

Most previous research on Kautz graphs focus on exploiting the Kautz graph in the application layer of P2P networks [17–19]. Zuo *et al.* [20] proposed to build a Kautz graph overlay on the application layer of a MANET in order to enhance the routing performance. However, due to the topology inconsistency, the method uses MANET multi-hop routing for the communication between two neighboring Kautz nodes. Panchapakesan *et al.* [26] and Li *et al.* [19] studied the shortest and longest path routing. In BAKE [18] and DFTR [21], a node uses the next shortest path when it fails to forward the message along the shortest path. However, a node needs to use a routing generation algorithm (equivalent to the process of building a tree) to find different routes to a destination node and calculate their lengths, which generate high energy consumption. Imase *et al.* [27] identified the bounds of the three possible path lengths in the worst case, but they did not indicate all disjoint paths, the precise path length and the corresponding conditions, which are identified in REFER.

III. REFER: A Kautz-based Real-Time and Energy-Efficient WSAN

Building an overlay on a WSAN for data transmission can avoid data flooding and hence enhance system scal-

ability and transmission efficiency in terms of speed and energy consumption [28]. A well-designed overlay is energy-efficient in topology maintenance, resilient to node mobility, and enables efficient and reliable routing. Below, we present the applicability of the Kautz graph topology to the WSAN overlay (Section III-A), the Kautz graph embedding protocol (Section III-B) and the efficient fault-tolerant routing protocol (Section III-C) with the objective of achieving the properties of a WASN overlay.

A. Is Kautz Graph an Reasonable Topology for WSAN Overlays?

When designing a WSAN overlay structure, we need to consider the tradeoff between network degree and diameter. The degree is the number of neighbors a node maintains and the diameter is the maximum distance between any two nodes. While a smaller degree generates lower maintenance overhead (energy consumption), it leads to a larger diameter and a longer transmission delay. Below, we study whether the Kautz graph is an reasonable overlay topology that achieves an tradeoff between degree and diameter for WSANs.

Definition 1 [29]. In a Kautz graph $K(d,k)$ with degree d and diameter k, nodes are labeled as $(u_1...u_k)$, where u_i belongs to an alphabet of $d+1$ letters $(A=(0,1,\ldots,d))$, and $u_i{\neq}u_{i+1}$ $(1 \leq i \leq k)$. The arc set of the Kautz digraph are $\{(u_1u_2\cdots u_k, u_2u_3\cdots u_ku_{k+1}) |\ u_i \in A; u_i{\neq}u_{i+1}\}$.

The left part of Figure 1 shows an example of K(2,3). For a graph G, $N(G)$ and $E(G)$ denote the number of nodes and edges of the graph, respectively. Graph G's *connectivity* is the minimum number of nodes whose removal results in a disconnected graph. A d-connected graph is a graph whose vertex connectivity is d or greater [30].

Definition 2 [30]. A graph connection optimization problem is to find a d-connected n-vertex graph with the smallest connectivity d given the number of vertices n and diameter k $(k<n)$.

Lemma 3.1: A Kautz graph $K(d,k)$ is a d-connected k-diameter n-vertex graph with minimum connectivity d.

Proof: Euler's Degree-Sum Theorem [30] shows that for a graph, $|E(G)| \geq N(G)\delta_{min}(G)$ where $\delta_{min}(G)$ is the minimum degree d. If G is a d-connected graph with the minimum degree, it must satisfy

$$|E(G)| = N(G)\delta_{min}(G).$$

The Kautz graph meets this condition since its $E(G){=}\ (d+1)d^k$ and $N(G)=n=(d+1)d^{k-1}$ [29]. ∎

It has been proved that the Kautz graph has a smaller diameter than the de-Bruijn and hypercube topologies [31], which have been widely studied as promising overlay topologies [19]. Based on this finding, Definition 2 and Lemma 3.1, we can get Proposition 3.1 below.

Proposition 3.1: A Kautz graph $K(d,k)$ optimizes the graph connection and it achieves an tradeoff between degree and diameter with its minimum degree and relatively shorter diameter.

Therefore, Kautz graph is an reasonable topology for WSAN overlays to meet the energy-efficiency and real-time requirements. The second issue that needs to be considered in designing a WSAN overlay structure involves the consistency between the overlay topology and the underlying physical topology, which is critical to real-time communication and energy-efficiency. However, the limited transmission range of sensors poses a challenge to achieving the topology consistency since two neighbor nodes in an overlay may be out of the transmission range of each other, and cannot be neighbors in the physical topology. Next, we study the precondition for the Kautz graph embedding to achieve the consistency.

Two neighbor nodes in the embedded graph overlay must be within the transmission range of each other. Otherwise, the nodes cannot be neighbors in the underlying physical network. A Kautz graph has a Hamiltonian cycle [30], in which a path traverses through every vertex in the graph exactly once before returning to the starting vertex. In order to achieve the consistency in the Kautz graph embedding, the underlying physical topology must also have a Hamiltonian cycle. Proposition 3.2 shows the requirement of a physical network for forming wireless nodes into a Hamiltonian cycle.

Proposition 3.2: Assume that the nodes are uniformly distributed in a square area with space length b in a WSAN, in order to guarantee that the selected nodes for graph embedding can form a Hamiltonian cycle, the transmission range r of the selected nodes should satisfy $r \geq 0.8 * b$.

Proof: Dirac [30] shows $d \geq \frac{n}{2}$ and $n \geq 3$ are sufficient condition to guarantee that the nodes in a graph can constitute a Hamiltonian cycle, where d is the connectivity of a node, and n is the total number of nodes in the graph. With the i.i.d assumption, when a node moves to the corner of the square, it has the least coverage area $(\frac{\pi r^2}{4})$ in the square. Then, the number of nodes in the coverage area (i.e., the degree d of the node) is $\frac{\pi r^2}{4b^2}n$. Thus

$$\frac{\pi r^2}{4b^2}n \geq \frac{n}{2} \Longrightarrow b \leq \frac{\sqrt{2\pi}}{2}r \Longrightarrow r \geq 0.8 * b \qquad (1).$$

∎

The proposition indicates that for a collection of sensors that can form a Kautz graph, the coverage area of the sensors is upper bounded by $(2*r+b)^2 = (\frac{13}{4}r)^2$. As the transmission range of sensors r is limited, the coverage area of the collection of sensors is limited. Therefore, we need a number of Kautz graph cells with small diameter and degree to cover a large area. Meanwhile, From the proposition we can see that the density of the sensors in a Kautz cell is high. Therefore, a sensor awake/sleep scheme is needed, which can save the energy of the sensors as well as ensure the connectivity of Kautz cells. The awake/sleep scheme will be further discussed in Section III-B4.

B. Kautz Graph Embedding Protocol

Proposition 3.2 indicates that the physically closed sensors need to be grouped into cells. Each of cell is composed by actuators and sensors. REFER embeds a Kautz graph into each WSAN cell. It has been proved [17] that as the diameter k decreases, the number of nodes in Kautz graph $K(d,k)$ approaches the Moore bound [32]. That is, node density in $K(d,k)$ increases as k decreases. Therefore, a Kautz graph

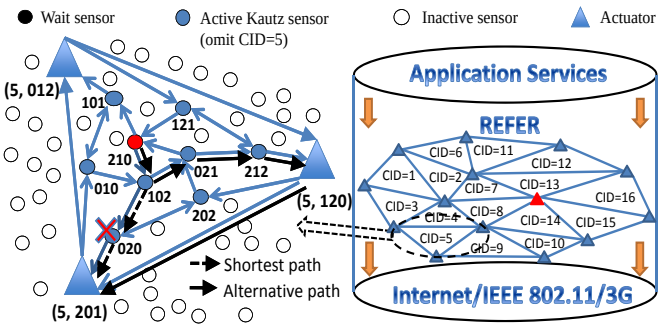

Fig. 1: The architecture of the REFER system.

with a smaller diameter k should be an good choice for an overlay to seamlessly cover a sensed region. The value of k should be set to the maximum of all minimum hop distance between any two nodes in the network. Based on the number of nodes $n = (d+1)d^{k-1}$ in a WSAN and k, the value d can be determined. Here, we choose the Kautz graph $K(d,3)$ as an example to explain the REFER overlay. We assume the WSAN meets the requirement of Proposition 3.2 and the sensors are densely deployed (e.g., habitat monitoring [2], battlefield monitoring [1].)

Although the communication between two nodes in a WSAN is bi-directional, we represent a WSAN as a directed graph $G(d,k)$ in order to clearly present the routing. Communication in the other direction can be conducted by simply reversing the direction. To facilitate the information transmission between cells, all actuators further constitute a DHT structure [16] for the action coordinations between actuators. DHTs are well-known for their scalability and dynamism-resilience. Figure 1 illustrates the architecture of REFER with an example of a Kautz graph in a cell. Actuators and several selected active sensors in each cell form a Kautz graph and the actuators further constitute a DHT structure. We use resource-rich actuators for the corner vertices of a Kautz graph because they can directly communicate with each other even though they are physically far apart.

Each WSAN cell has a cell ID (CID) (e.g, 1-15 in Figure 1). Each node in a cell with CID has ID=(CID, KID), where KID=$\{u_1 \dots u_i \dots u_k \mid u_i \in [0,d],\ u_i \neq u_{i+1}\}$ (e.g, 201) is the Kautz ID in the Kautz graph. For a pair of nodes $U = u_1u_2...u_k$ and $V = v_1v_2...v_k$, we use $l = L(U,V)$ to denote the length of the longest suffix of node U that appears as a prefix of node V. The distance between two cells is measured by the Euclidean distance between their CIDs, while the distance between Kautz nodes U and V in one cell equals $k - L(U,V)$. For example, the distance between 120 and 201 is $k - L(120,201) = 3 - 2 = 1$. An actuator stays in several adjacent cells and hence has different CIDs for different cells. In order to reduce the system complexity, we let each actuator have the same KID used for all Kautz graphs it resides in.

To achieve the consistency between overlay and physical topology, we rely on node communication to determine node ID since the real node communication distance reflects node physical distances. The process of embedding Kautz graph to a cell is actually the process of Kautz ID assignment. It involves two steps: actuator ID assignment and sensor ID assignment.

1) Actuator ID Assignment: The actuator ID assignment process detects triangles among the neighboring actuators and sequentially assign IDs to the actuators. Each actuator A has a value $H(A)$ which is the consistent hash value [33] of its IP address. Neighboring actuators exchange the information of their neighbors along with their $H(A)$, and finally each actuator learns the global topology of actuators. The actuator with the minimum $H(A)$ functions as a starting server to assign CIDs to others. It locally partitions the global topology to a series of triangles and assigns a distinct CID to each triangle (cell). Closer cells have closer CIDs. The starting server also calculates the KIDs for the actuators in each cell. Neighboring actuators cannot have the same KID since they are in the same cell. For this purpose, we employ the sequential vertex-coloring algorithm [30], in which a node is assigned with the smallest color number not used by its neighbors. As only three actuators are in Kautz graph $K(d,3)$, three colors (i.e., KID 012, 120, 201) are needed. Finally, each actuator is assigned with an ID=(CID,KID), e.g., $(5, 201)$. The starting server then notifies other actuators about their IDs using the depth-first search algorithm based on the topology of actuators.

2) Sensor ID Assignment: After the actuators in each cell receive their IDs, they select active sensors in the cell to be Kautz nodes to form a complete $K(d,3)$ graph. First, the Kautz nodes connecting actuators are identified. Then, the Kautz nodes connecting the Kautz sensors are identified. For KID=kid, we use kid_l to denote the result of left rotating kid once. The successor actuator of actuator kid is the actuator that has KID=kid_l in the same Kautz graph. First, each actuator selects sensors to connect to its successor actuator. Actuator kid broadcasts a path query message towards actuators with KID=kid_l in the same cell with TTL=2, which ensures the diameter $k = 3$ for $K(d,3)$. Each forwarding sensor includes the information of itself and its energy level into the routing message. Finally, the successor actuator receives a number of messages, and each message includes its path and the energy of path sensors. It selects a path with the highest accumulated energy, and assigns CIDs and KIDs to the sensors in the path. Each sensor's CID equals to the actuator's CID. If its previous node's KID in the path is $(u_1u_2u_3)$, its KID is then $(u_2u_3u_k)$ where u_k is the letter that makes $u_2u_3u_k$ close to the successor actuator's KID. For example, actuator $(5,012)$ assigns IDs $(5,010)$ and $(5,101)$ to the two nodes in the path from actuator $(5,201)$ to itself. Similarly, the other two paths are built: $(5,120) \rightarrow (5,202) \rightarrow (5,020) \rightarrow (5,201)$ and $(5,012) \rightarrow (5,121) \rightarrow (5,212) \rightarrow (5,120)$.

To select the rest of the Kautz nodes connecting Kautz sensors, we rely on sensor communication. In order to ensure that the message traverses the longest path between a pair of Kautz sensors, we choose the successor of the actuator with the smallest KID= $u_1u_2u_3$ (i.e., S_i=$u_2u_3u_2$=121), and the predecessor of the actuator with the largest KID= $u_3u_1u_2$ (i.e., S_j=$u_1u_3u_1$=020). S_i broadcasts a path query message with TTL=2 towards S_j. S_j selects the path with the highest accumulated energy and assigns KID=$u_3u_2u_1$ (i.e., 210) and KID=$u_2u_1u_3$ (i.e., 102) to the sensors in the path. Next, the

common neighbor of the two newly selected nodes with the highest battery power is assigned with KID=$u_1u_3u_2$=021 as the last Kautz node in the cell.

3) DHT-based Upper Tier Structure: CAN [16] is a mesh-based structured P2P network, in which nodes in a virtual multi-dimensional coordinate space are dynamically partitioned and every node owns a distinct zone. Each node maintains a neighbor set including those nodes that hold coordinate zones adjoining its own zone. Using its neighbor set, a node routes a message by simply forwarding it to the neighbor with coordinates closest to the destination coordinates. REFER builds actuators into a CAN by directly using CID as CAN ID. Basically, each actuator exchanges its CID with its neighbors and establishes its neighbor set. When an actuator receives a message destined to a cell, it forwards the message to its neighboring actuator with the CID closest to the cell's CID.

4) Topology Maintenance: Considering the high density of the WSAN and the fact that not all sensors are Kautz nodes, we use node replacement strategy for Kautz topology maintenance and use an awake/sleep scheme [6] to further save energy. Specifically, REFER sets three functional states for sensors: *active*, *wait* and *sleep*. Nodes in the active state form a Kautz graph. Each node in the sleep state periodically wakes up and probes its Kautz node neighbors to see if it can be a candidate for them. The candidate of Kautz node S must be able to build connections with the neighboring Kautz nodes of S. The candidate nodes stay in the *wait* state. When a Kautz node notices that its links to its current neighbors are about to break up according to sensed signal strength or its own battery power is below a threshold, it selects one of its current candidate nodes to replace itself by sending notification messages.

C. Efficient Fault-Tolerant Routing Protocol

Communication between sensors consumes high energy [6]. A tradeoff exists between fault-tolerance/real-time and energy consumption in routing. A Kautz graph can help to achieve an reasonable tradeoff. A Kautz graph with degree d has d disjoint paths between any two nodes [31]. For example, Figure 2(a) illustrates the 4 paths between node 0123 and node 2301 in Kautz graph $K(4,4)$. This topology feature of Kautz graphs supports fault-tolerant routing protocols [18, 21], in which if a node fails to forward a message along the shortest path, it chooses the successor in the second-shortest path, then third-shortest path, and so on. For example, in Figure 2(a), after node 0123 initiates or receives a message destined to node 2301, if node 1230 in the shortest path (dotted links) fails to forward the message, 0123 chooses the successor in the next shortest path, say node 1232. In Figure 1, node 102 locally chooses an alternative path from itself to node 201 with 021 as the next hop (links with solid arrows) when node 020 fails. However, in the previous Kautz-based routing protocols, a node needs to use a routing generation protocol to find the d disjoint paths to a destination node and their lengths [21], which consumes enormous amount of computing resources. To overcome this shortcoming, REFER aims to develop a routing protocol that can find successors quickly based only on node IDs with low energy consumption.

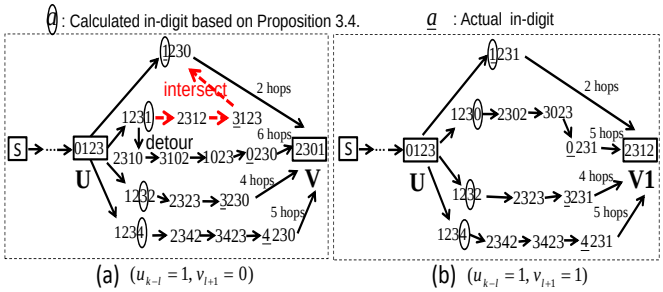

Fig. 2: Examples of routing paths in a Kautz graph.

1) Analysis of Kautz Graph Properties for Routing: In the U-V path, the next hop's label is generated by left shifting U one digit and appends a new digit v_i at the right side of U. Thus, in the routing along the shortest path, node U_0 greedily forwards data to the next hop U_1 whose suffix shares the maximum identical digits with the destination V, i.e., $\max(L(U_1, V))$. We call this routing protocol *greedy shortest protocol*. An example of the shortest routing path is: $12345 \rightarrow 23450 \rightarrow 34501$. For any pair of nodes U-V, there exists only a single shortest path, and its length is $k-l$.

In the d U-V disjoint paths, U's *successors* are the next hops of U in the paths denoted by $u_2u_3u_4\ldots u_k\alpha_i$ $(u_k{\neq}\alpha_i)$. Similarly, V's *predecessors* are the previous hops of V in the paths denoted by $\beta_i v_1 v_2 ... v_{k-1}$ $(v_1{\neq}\beta_i)$. In this paper, we use $\alpha_i{\prec}[0,d]$ to mean that α_i are all different values in the range of $[0,d]$, and use $\alpha_i{\in}[0,d]$ to mean that α_i is one value in $[0,d]$.

Definition 3. For a U-V pair (Figure 3), the last digit of the successor of U in a path is called the *out-digit* of the path $(\alpha_i{\in}[0,d])$, and the first digit of the predecessor in a path of V is called the *in-digit* of the path $(\beta_i{\in}[0,d])$.

In a U-V routing, if U's successor fails to forward data, simply choosing another successor may lead to an intersection between this U-V path and another U-V path, leading to traffic congestion in the intersection node. To avoid the congestion, a key question is how to proactively find the intersection nodes and avoid these nodes in routing. Seeking the answer is also the process of exploring the d-disjoint U-V paths. We show the process of our exploration in below. We finally reach Theorem 3.8, which allows a node to directly discover d-disjoint paths and their lengths by simply comparing its own KID and the destination KID.

Proposition 3.3: In a U-V path, if U's successor $u_2u_3u_4\ldots u_k\alpha_i$ $(\alpha_i{\in}[0,d]$ & $\alpha_i{\neq}u_k)$ uses the greedy shortest protocol, the generated *in-digit* are:

$$\beta = \begin{cases} u_{k-l} & \text{if } \alpha_i = v_{l+1} \text{ (shortest path)} \\ \begin{cases} u_k & \text{if } \alpha_i = v_1 \\ \alpha_i & \text{if } \alpha_i{\neq}v_1 \end{cases} & \text{if } \alpha_i{\neq}v_{l+1} \text{ (non-shortest path)} \end{cases}$$

Proof: For a U-V path with $u_2u_3u_4\ldots u_k\alpha_i$ $(\alpha_i{\in}[0,d]$ & $\alpha_i{\neq}u_k)$ as U's successors, if $\alpha_i = v_{l+1}$, the U-V path is the shortest path and $u_{k-l+1} = v_1$. Thus, the in-digit of this path is u_{k-l}. In the non-shortest U-V paths, if $\alpha_i = v_1$, the in-digit is u_k. If $\alpha_i{\neq}v_1$, the in-digit of the path equals α_i, since the next hop is $u_3u_4\ldots u_k\alpha_i v_1$ ■

Example for Proposition 3.3: In Figure 2(a), because U

and V share digits 23, $l = L(U, V) = 2$. For the shortest path traversing successor 1230, the in-digit of the path is $u_{k-l} = 1$. For the successor with $\alpha_i = v_1$, i.e., 1232, the in-digit is $u_k = 3$. For other successors, 1231's in-digit is $\alpha_i = 1$ and 1234's in-digit is $\alpha_i = 4$.

Proposition 3.4: For a U-V pair, paths traversing nodes having the same in-digit β will intersect at $\beta v_1 v_2 \dots v_{k-1}$ using the greedy shortest protocol.

Proof: For a U-V pair, data in a node with in-digit β will be forwarded to the predecessor of V $\beta v_1 v_2 \dots v_{k-1}$ using the greedy shortest protocol. ■

Example for Proposition 3.4: Considering the U-V pair in Figure 2(a), we can see that the successor 1230 of the shortest path shares the same in-digit (i.e., 1) as successor 1231 if both of these successors forward data with the greedy shortest protocol. The two paths will intersect at 1230 as shown by the dotted line.

Proposition 3.5: The non-shortest paths with different out-digits will not intersect with each other.

Proof: Suppose two non-shortest paths with different out-digits have an intersection. Using the greedy shortest protocol, the paths will have the same in-digit. It conflicts with Proposition 3.3, which shows that the non-shortest paths with different out-digits have different in-digits. ■

Proposition 3.6: For a U-V pair, when U's successors use the greedy shortest protocol, only when $u_{k-l} \neq v_{l+1}$, the shortest path intersects with the non-shortest path which has $\alpha_i = u_{k-l}$.

Proof: According to Proposition 3.3, we know that the out-digit and in-digit of the shortest path are $\alpha_i = v_{l+1}$ and $\beta_i = u_{k-l}$, respectively. The in-digits of other non-shortest paths are $\beta_i = (\alpha_i \prec [0, d]$ & $\alpha_i \neq v_{l+1})$. If $u_{k-l} = v_{l+1}$, then $\alpha_i \neq u_{k-l}$. Then, none of the in-digits of non-shortest paths equal to the in-digit of the shortest path. Therefore, the in-digits of all d disjoint U-V paths (shortest and non-shortest paths) are different, i.e., $\beta_i \prec [0, d]$. If $u_{k-l} \neq v_{l+1}$, one non-shortest path's in-digit is u_{k-l}, which is also the in-digit of the shortest path. Thus, the in-digit of two of d disjoint paths is u_{k-l}. Based on Proposition 3.4 and Proposition 3.5, the proof is completed. ■

Example for Proposition 3.6: In Figure 2(a), the U-V pair satisfies $u_{k-l}(u_2{=}1) \neq v_{l+1}(v_3{=}0)$. There are four successors for the total $d = 4$ disjoint paths of the U-V pair: node 1230, 1231, 1232 and 1234. Node 1230 is in the shortest path (when $\alpha = 0$) and its in-digit is $u_{k-l} = 1$. The in-digits of the remaining paths (when $\alpha \neq 0$) are 1, 3, and 4. Thus, the in-digit of two of the d disjoint paths is $u_{k-l} = 1$. 1230 and 1231 have the same in-digit 1, the paths traversing them using the greedy shortest protocol intersect at 1230. In Figure 2(b), the U-V1 pair does not satisfy $u_{k-l}(u_2 = 1) \neq v_{l+1}(v_3 = 1)$. It has four successors for U in the total $d = 4$ disjoint paths: node 1230, 1231, 1232 and 1234. Since 1231 is in the shortest path, its in-digit is $u_{k-l} = 1$. The in-digits of the non-shortest paths are 0, 3 and 4, i.e., $\beta_i{=}(\alpha_i \prec [0, 4]$ & $\alpha_i \neq v_{l+1} = 1)$. Thus, the in-digits of total d disjoint paths are different, i.e., $\beta_i \prec [0, 4]$. As shown in the figure, the paths do not intersect.

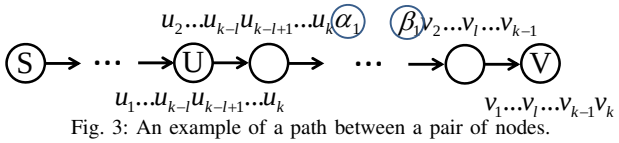

Fig. 3: An example of a path between a pair of nodes.

Definition 4. For a U-V pair with $u_{k-l} \neq v_{l+1}$, the U's successor $u_2 u_3 \dots u_k u_{k-l}$ with $\alpha_i = u_{k-l}$ is called a *conflict node* that leads to an intersection with the shortest path.

Proposition 3.7: For a U-V pair, the conflict node $u_2 u_3 \dots u_k u_{k-l}$ should forward data to node $u_3 u_4 \dots u_k u_{k-l} v_{l+1}$ in order to avoid intersection with the shortest path.

Proof: In addition to the conflict node $u_2 u_3 \dots u_k u_{k-l}$, other successors of U are $u_2 u_3 u_{k-l} u_k \alpha_i$ $(\alpha_i \prec [0, d]$ & $\alpha_i \neq v_{l+1})$ and their in-digits are $\beta_i = (\alpha_i \prec [0, d]$ & $\alpha_i \neq v_{l+1})$. Thus, having v_{l+1} as the in-digit of the path for the conflict node results in different in-digits for different d paths, i.e., $\beta_i \prec [0, d]$. That is, no paths exist with the same in-digit. Based on Proposition 3.5, the proof is completed. ■

Example for Proposition 3.7: In Figure 2(a), the in-digits of non-shortest paths are 1, 3, 4, i.e., $\beta_i{=}(\alpha_i \prec [0, 4]$ & $\alpha_i \neq v_{l+1} = 0)$, respectively. The conflict node is 1231 $(u_{k-l} = u_2 = 1)$. To avoid intersection with the shortest path, node 1231 uses $v_{l+1} = 0$ as its in-digit by forwarding data to 2310. Thusly, the d-disjoint paths with $\beta_i \prec [0, 4]$ for the U-V pair are built.

Theorem 3.8: When node U=$u_1 u_2 \dots u_k$ forwards data to node V=$v_1 v_2 \dots v_k$, the successor, path length and corresponding condition of the d disjoint U-V paths are:

$$\begin{cases} (1)\ u_2 \dots u_k u_{k-l}; & k+2, & \text{when } u_{k-l} \neq v_{l+1}; \\ (2)\ u_2 \dots u_{k-l} \dots u_k v_{l+1}; & k-l, & \text{the shortest path}; \\ (3)\ u_2 \dots u_k v_1; & k, & \text{when } u_k \neq v_1; \\ (4)\ u_2 \dots u_k \alpha_i; & k+1, & \text{otherwise}, \end{cases}$$

where $\alpha_i \neq (v_1, v_{l+1}, u_{k-l})$.

Proof: (1) According to Proposition 3.6, when $u_{k-l} \neq v_{l+1}$, there is one non-shortest path that intersects with the shortest path. According to Proposition 3.7, this path enters a path with in-digit v_{l+1}. The shortest length of this path to V is k. Therefore, the entire path length is $k + 2$; (2) For a U-V pair, the maximum length of the shortest path is $k - l$; (3) When $u_k \neq v_1$, there exists a successor of U with an out-digit of v_1. The path length of the path starting from this successor is $k - 1$. Therefore, the path length of the U-V pair through this successor is k; (4) For all other cases, each successor starts with out-digit v_1. The path length from the successor to the destination is k. Then, the lengths of the U-V paths are $k + 1$. ■

2) REFER Routing Protocol: The REFER routing protocol consists of intra-cell communication and inter-cell communication. The intra-cell communication is developed based on Theorem 3.8. The theorem incorporating Proposition 3.7 enables a node to quickly and efficiently determine the different successors of the d-disjoint paths from itself to the destination node and corresponding path lengths simply based on node IDs without relying on an energy-consuming method (e.g.,

tree [21]). Thus, in the intra-cell communication, when node U initiates or receives a message destined to node V, it initially chooses its successor in the shortest path to V (the greedy shortest protocol). If the successor is congested/failed or the link to the successor is broken down, based on Theorem 3.8, without the need to notify the source node, U locally chooses the second shortest path, third shortest path, and so on until a successor capable of forwarding data is identified. If a number of paths with the same path length exist, U randomly chooses a successor among these paths. To forward the message to the successor, the node chooses a path with the lowest delay [12], which could be either a multi-hop path or direct path. After a node receives U's message, it repeats the same process in choosing its successor for message routing. For example, in Figure 2(a), node 0123 wants to send a message to 2301. It first uses the greedy shortest protocol to forward the message to node 1230. If node 1230 is congested/failed or the link between 0123 and 1230 has broken, node 0123 chooses the successor in the second shortest path. It compares its own KID with 2301's KID based on Theorem 3.8. Since $u_k \neq v_1$ and $u_{k-2} \neq v_3$, the successors and path lengths of the remaining 3 disjoint paths are (node 1231, k+2=6), (node 1232, k=4), and (node 1234, k+1=5). Then, node 0123 chooses the successor 1232 in the second shortest path (i.e., 4). If node 1232 has failed to forward the message, the successor 1234 in the third shortest path (i.e., 5) is chosen. After node 2342 receives the message, it repeats the same process by executing the routing protocol. Note that the routing is based on the Kautz node indices on the Kautz graph, a routing hole in which a node does not have a neighbor to forward a message will not occur as long as the Kautz graph topology is maintained.

In REFER, when a node sends out a data with destination (cid,kid), it forwards the data to its actuator by intra-cell transmission. The data is then forwarded to its destination cell identified by cid via inter-cell transmission, and subsequently forwarded to the destination node identified by kid via intra-cell transmission. In the inter-cell transmission, data is routed based on the CAN DHT routing protocol, in which a node forwards the data to its neighbor closest to the destination. For example, in Figure 1, the actuator with CID=14 wants to send a message to node $(5, 201)$, the actuator forwards the message to its neighbor actuator with CID=7, which is the closest to 5 in its neighbor set. Then, the message receiver forwards the message to its neighbor actuator with CID=4, which further forwards the data to its neighbor actuator in cell 5. Lastly, intra-cell transmission is used to forward the data to node $(5, 201)$. Because the REFER overlay preserves the consistency between overlay topology and underlying physical topology, nodes with virtually close IDs are also physically close. Thus, the data is transmitted between physically close nodes, enhancing the real-time performance.

IV. Performance Evaluation

We used ns-2 [34] to evaluate the performance of REFER in comparison with the DaTree [2] tree-based system, the D-DEAR [8] mesh-based system, and the Kautz-based overlay [20] (denoted by Kautz-overlay) for WSANs. To make the systems comparable, we use the topological routing in [35] for node communication. In DaTree, one actuator (tree root) and its physically close sensors form a tree. Each sensor belongs to only one tree and forwards its detected events to its tree root. If a sensor's link to its parent breaks in routing, the sensor needs to broadcast a message to the root in order to update its parent. In D-DEAR, physically close sensors are clustered together and a sensor with more energy is selected as the cluster head, which maintains a multi-hop path to a close actuator. Messages are sent from sensors to their cluster head, and then further forwarded to the close actuator. The cluster heads also use broadcast to update the paths to the actuator upon a routing failure. We used REFER's routing protocol in Kautz-overlay to have a fair comparison. In Kautz-overlay, when a node fails to forward a message to another node, it uses broadcasting to re-establish a path to the node.

Unless otherwise specified, 5 actuators were uniformly distributed in a 500m $\times$ 500m area and 200 sensors were i.i.d distributed around the actuators, which form 4 Kautz cells with Kautz graph as K(2,3). Such simulation scenario is similar to that in [36]. The transmission ranges of sensors and actuators were set to 100m and 250m, respectively. Every 10 seconds, we randomly chose 5 source nodes, which transmit data to their nearby actuators at the rate of 1Mbps.

Sensors communicate with each other using the IEEE 802.11 protocol. In the simulation, each sensor randomly selects a destination point and moves to that point with a speed randomly selected from [0,3]m/s. The warmup time and simulation time were set to 100s and 1000s, respectively. Since most packets can arrive at the destinations within 1s, we only counted those arriving at the destination within 0.6s into the throughput in order to show the real-time transmission performance. We call these packages QoS-guaranteed data. The amounts of energy consumed in the transmission and receiving modes were set to 2 and 0.75 Joules/packet [37], respectively. All experimental results report 95% confidence intervals. We use the following metrics for the performance evaluation: (1) *Throughput*. The size of received data by all actuators per second. We scale the measured throughput in 0.6s to 1s. A high throughput implies higher fault-tolerance and real-time performance. (2) *Delay*. The average latency for the transmission of QoS-guaranteed data. Shorter delay indicates higher real-time performance. (3) *Energy consumed in topology construction/communication*. The total consumed energy of all sensors in topology construction and in node communication for data forwarding and topology maintenance, respectively. Less consumed energy indicates higher energy-efficiency of a system.

A. Mobility Resilience

In this experiment, a node's speed was randomly selected from $[0, 5]$. Figure 4 shows the throughput of each system versus the average node mobility speed $x/2$. It demonstrates that higher node mobility leads to a slight throughput decrease in REFER, moderate throughput decrease in DaTree and D-DEAR, and a sharp throughput decrease in Kautz-overlay.

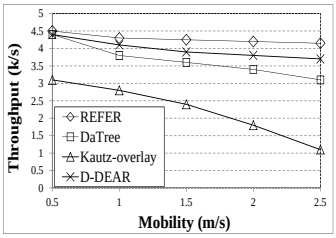

Fig. 4: Throughput vs. mobility.

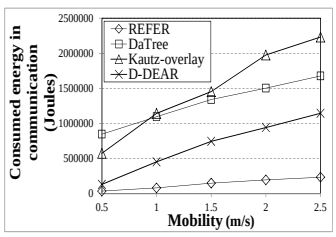

Fig. 5: Energy consumed vs. mobility.

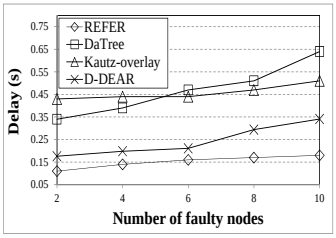

Fig. 6: Delay vs. faulty nodes.

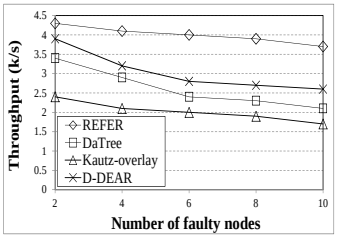

Fig. 7: Throughput vs. faulty nodes.

REFER directly embeds Kautz graphs into the physical topology in order to keep the topology consistency. Therefore, messages are quickly forwarded along physically close nodes. Higher mobility leads to more message forwarding failures. With REFER's routing protocol, a node can quickly use an alternative path from itself to the destination upon a forwarding failure. Thus, REFER can forward more messages in a limited time, leading to a high throughput. REFER's slight decrease in throughput is caused by the slightly longer lengths of the alternative paths. In D-DEAR, only cluster heads need to maintain long multi-hop paths to actuators, and all other sensors can directly reach their cluster heads. When a multi-hop path breaks, a cluster head uses broadcasting to find a new path to an actuator. The delay for the multi-hop path recovery and message retransmission results in the decrease in throughput. In DaTree, if a sensor fails to forward a message to its parent, it uses broadcasting destined to the root for a link reestablishment with a new parent, leading to a long delay and low throughput. Since DaTree has more nodes being affected by the mobility, its overall throughput is much smaller than D-DEAR in a highly mobile environment.

Figure 5 shows the energy consumed in communication for each system versus node mobility. The figure illustrates that the consumed energy of all systems increases as node mobility increases. This is because higher mobility triggers more path updates. The figure also shows that REFER consumes significantly less energy than others, and Kautz-overlay and DaTree consume much more energy than D-DEAR. Without the message retransmission and topology consistency, REFER consumes low energy in message transmission. In REFER's topology maintenance, nodes only need to periodically probe their nearby neighbors and replace them if they cannot continue to be the Kautz nodes. Therefore, REFER's consumed energy exhibits a slight increase when node mobility increases. In D-DEAR, upon a forwarding failure, a cluster head needs to use broadcasting to rebuild the routing path to its actuator, so its consumed energy increases rapidly as the node mobility increases. In DaTree, upon a forwarding failure, a node needs to use broadcasting to find a new parent and retransmit the message. Since DaTree needs all nodes to update links rather than partial nodes as in D-DEAR, DaTree consumes more energy than D-DEAR, especially in a highly mobile environment. In Kautz-overlay, the multi-hop paths between the neighboring overlay nodes are more likely to break up with high mobility, consuming more energy in path updates. Because Kautz-overlay need maintain multiple consecutive multi-hop paths, Kautz-overlay consumes much more energy than DaTree in a highly mobile environment. It is interesting to see that when mobility is 0.5m/s, Kautz-overlay consumes less energy than DaTree. As much less path updates and message retransmission occur in a low mobile environment, Kautz-overlay consumes less energy than DaTree.

B. Fault-Tolerant Routing

We define *faulty nodes* as broken-down nodes that cannot function normally. We randomly chose a set of faulty nodes in the system every $10s$ and recovered the previous set of faulty nodes. The number of faulty nodes was set to $2x$, where x is randomly chosen from $[1, 5]$. Figure 6 plots the average transmission delay versus the number of faulty nodes. We notice that as the number of faulty nodes increases, the delays of DaTree and D-DEAR grow faster than REFER and Kautz-overlay. This is because of the fault-tolerant routing in REFER and Kautz-overlay which enable a node to use an alternative path upon a forwarding failure. Their slight delay growth is caused by the lengthened routing path of an alternative path. However, Kautz-overlay's multi-hop transmission between two neighboring Kautz nodes leads to long transmission delay. In contrast, REFER keeps the topology consistency and enables neighboring Kautz nodes in a Kautz routing path to directly communicate with each other, resulting in the least delay.

In DaTree, every node needs to use broadcasting to send a message to its actuator to rebuild a link to a new parent upon a forwarding failure. More faulty nodes generate more link re-establishments, leading to higher transmission delay. In D-DEAR, as only cluster heads rather than all nodes need to update the transmission paths to the actuators, it generates less transmission delay than DaTree. It is intriguing to see that DaTree has lower delay than Kautz-overlay when the number of faulty nodes is less than 6, but it has higher delay thereafter. DaTree usually has one multi-hop path while Kautz-overlay has a number of consecutive multi-hop paths in one routing. When there are only a few faulty nodes, most messages can be transmitted successfully. Consequently, DaTree generates lower delay due to its shorter path length. More faulty nodes trigger more forwarding failures, for which DaTree needs message retransmission while Kautz-overlay does not. Therefore, DaTree produces higher delay than Kautz-overlay.

Figure 7 shows the throughput of systems versus the number of faulty nodes. The figure shows that the throughput of all systems decreases as faulty nodes grow. This is because more faulty nodes trigger more message drops and delayed transmission. We can also see that the throughput of REFER and Kautz-overlay decreases slower than DaTree and D-DEAR due to their fault-tolerant routing as explained in Figure 6. In DaTree and D-DEAR, upon a routing failure due to faulty

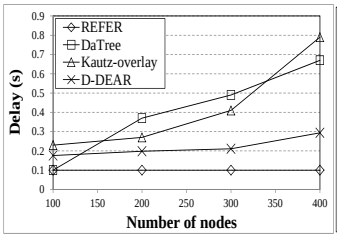

Fig. 8: Delay vs. total nodes.

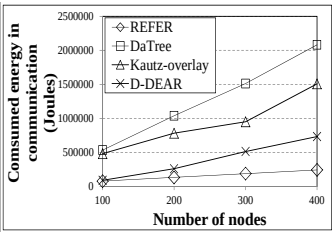

Fig. 9: Energy consumed vs. total nodes.

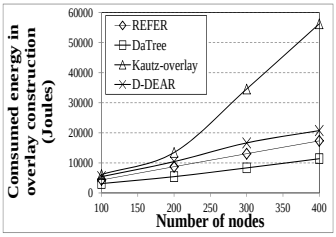

Fig. 10: Energy consumed in overlay construction.

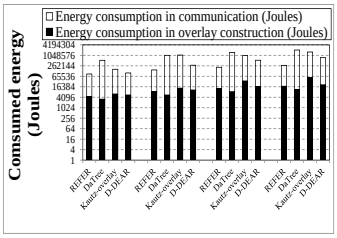

Fig. 11: Total energy consumed.

nodes, the delay from the path re-establishment reduces the throughput during the simulation time. The faulty nodes only affect the paths between the cluster heads and actuators in D-DEAR, but affect the paths between all sensors and actuators in DaTree. Therefore, D-DEAR generates higher throughput than DaTree. Because Kautz-overlay produces a much longer transmission path length for one message transmission than all other systems, it produces the least throughput during the limited simulation time.

C. Real-Time Transmission

Figure 8 shows the delay of each system when the network size was varied from 100 to 400. As the number of nodes in the system increases, the delay remains nearly constant in REFER, moderately increases in D-DEAR, while sharply increases in DaTree and Kautz-overlay. Also, DaTree and Kautz-overlay generate much higher delay than D-DEAR and REFER when the number of nodes is larger than 100. In REFER, messages are always forwarded between the physically close nodes. Also, since the number of nodes in a basic cell is fixed, the distances of message transmission do not change as the network size increases. Further, REFER's routing protocol can reliably forward messages without retransmission. Consequently, its transmission delay remains almost constant. In D-DEAR, only the transmission path lengths between cluster heads and actuators increase as network size grows, thus its overall transmission delay slightly increases. The reason for the sharp increase in DaTree and Kautz-overlay is because the path lengths between all sensors and their actuators increase as network size increases, since all sensors in the system function as relay nodes for message forwarding. It is intriguing to see that when the number of nodes is 100 in the system, DaTree generates approximately the same delay as REFER, which is less than that of D-DEAR. This is because when the network scale is small, many nodes are close to the actuators. In DaTree, many sensors can directly send messages to actuators. In D-DEAR, even if a sensor is close to an actuator, it still needs to send its messages to its cluster head, which further forwards the message to the actuator, resulting in longer delay.

D. Scalability and Energy-efficiency

Figure 9 demonstrates the energy consumed in communication for each system versus network size. Here, as the networks increase in size, the consumed energy of REFER shows a marginal increase while that of DaTree, Kautz-overlay and D-DEAR exhibits a rapid increase. The result verifies the high energy-efficiency and scalability of REFER. Recall that REFER chooses a multi-hop path rather than a direct path for routing between neighboring Kautz nodes if the multi-hop path leads to lower delay. Therefore, as the network size increases, REFER has higher probability of having a slightly longer multi-hop with lower delay, resulting in a slight increase in the consumed energy. In D-DEAR, DaTree and Kautz-overlay, the path length increases as the network size increases. Also, longer path length increases the probability of path breakups, which triggers more path updates. Thus, the consumed energy of these systems increases quickly. We also observe DaTree consumes more energy than D-DEAR and Kautz-overlay. The consumed energy in communication is for message transmission and topology updates. The routing paths between all sensors and actuators increase in DaTree, while only those between the cluster heads and actuators increase in D-DEAR. Thus, DaTree needs more energy than D-DEAR. In a moderate mobile environment, message transmission dominates the influence on the energy consumed due to less topology updates. Kautz-overlay does not need message retransmission upon routing failure due to its fault-tolerant routing protocol, while DaTree needs message retransmission. Consequently, DaTree consumes more energy than Kautz-overlay, which is consistent with the result in a low mobile environment in Figure 5.

Figure 10 shows the energy consumed in topology construction for each system versus the network size. We can see that Kautz-overlay consumes the most energy for the overlay construction. The reason is that every node in Kautz-overlay needs to use broadcasting to build a multi-hop path to each of its overlay neighbor. As the overlay in both D-DEAR and REFER are formed by physically close nodes, they consume less energy. In D-DEAR, since every node locally contacts neighbors within 2 hops to select its cluster head, its energy consumption is much less than Kautz-overlay. In REFER, actuators need to exchange information and broadcast messages to all nodes in the cells for actuator ID assignment. Also, several communications between actuators and sensors are needed for selecting Kautz nodes and KID assignment. Therefore, it consumes more energy in topology construction than D-DEAR. In DaTree, each actuator broadcasts one message to the sensors in the system. After receiving the message, a sensor sets the message forwarder as its parent. Therefore, it consumes the least energy in overlay construction. Figure 11 combines the energy consumed for communication and topology construction. We notice that topology construction consumes negligible energy compared to that of communication (0.1%). Thus, the result confirms that REFER is energy-efficient in terms of total energy consumption.

V. Conclusion

Real-time, energy-efficiency and fault-tolerance are critical requirements for WSAN applications. Current routing protocols proposed for WSANs fall short in meeting these requirements. In this paper, we theoretically studied the properties of the Kautz graph, which shows that the Kautz graph is an optimal topology for WSANs to meet the requirements. Thus, we propose REFER, which incorporates a Kautz graph embedding protocol and an efficient fault-tolerant routing protocol. REFER's embedded Kautz topology is consistent with the physical topology, facilitating real-time communication. Further, REFER leverages DHT for the communication between Kautz-based cells for high scalability. Our theoretical analysis on the Kautz paths serve as the cornerstone for REFER's routing protocol. It is advantageous over previous Kautz-based routing algorithms by enabling a node to directly determine different routing paths and path lengths simply based on node IDs without relying on an energy-consuming method. Extensive experimental results show the high performance of REFER compared with other WSAN systems and previous Kautz-based overlay. In the future, we will evaluate REFER in the GENI real-world testbed using trace data. We will also investigate the performance of REFER in a sparse WSAN and the Kautz graph $K(d, k)$ with various d and k values.

Acknowledgements

This research was supported in part by U.S. NSF grants OCI-1064230, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, CNS-0938189, CSR-2008826, CSR-2008827, Microsoft Research Faculty Fellowship 8300751, and Oak Ridge Award 2008833.

References

[1] L. F. Akyildiz and I. H. Kasimoglu. Wireless sensor and actor networks: research challenges. *Ad hoc networks*, 2004.

[2] T. Melodia, D. Pompili, V. C. Gungor, and I. F. Akyidiz. A distributed coordination framework for wireless sensor and actuator networks. In *Proc. of MobiCom*, 2005.

[3] J. A. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou. Real-time communication and coordination in embedded sensor networks. In *Proc. of the IEEE*, 2003.

[4] J. Wu, S. Yang, and M. Cardei. On maintaining sensor-actor connectivity in wireless sensor and actor networks. In *Proc. of Infocom*, 2008.

[5] B. Krishnamachari. An Introduction to Wireless Sensor Networks. In *Proc. of ICISIP*, 2005.

[6] J. A. Sanchez, P. M. Ruiz, and I. stojmenovic. Energy-eficient geographic multicast routing for sensor and actuator networks. *Computer Communications, Elsevier*, 2007.

[7] E. Ngai, M. R. Lyu, and J. Liu. A real-time communication framework for wireless sensor-actuator networks. In *Proc. of AC*, 2006.

[8] G. A. Shah, M. Bozyigit, O. B. Akan, and B. Baykal. Real-time coordination and routing in wireless sensor and actor networks. In *Proc. of NEW2AN*, 2006.

[9] W. Hu, N. Bulusu, and S. Jha. A communication paradigm for hybrid sensor/actuator networks. In *Proc. of PRIMC*, 2004.

[10] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wieless networks. *Wireless Networks*, 2001.

[11] Z. Li and H. Shen. A mobility and congestion resilient data management system for mobile distributed networks. In *Proc. of MASS*, 2009.

[12] C. Perkins, E. Belding-Royer, and S. Das. RFC 3561: Ad hoc on demand distance vector (AODV) routing, 2003.

[13] E. P. Charles and P. Bhagwat. Highly dynamic destination sequenced distance vector routing (DSDV) for mobile computers. In *Proc. of Sigcomm*, 1994.

[14] M. Caesar, M. Castro, E. B. Nightingale, G. O. Shea, and A. Rowstron. Virtual ring routing: network routing inspired by DHTs. In *Proc. of Sigcomm*, 2006.

[15] A. Awad, R. German, and F. Dressler. P2P-based routing and data management using the virtual cord protocol. In *Proc. of MobiCom*, 2006.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of Sigcomm*, 2001.

[17] D. Guo, J. Wu, H. Chen, and X. Luo. Moore: An extendable peer-to-peer network based on incomplete kautz digraph with constant degree. In *Proc. of Infocom*, 2007.

[18] D. Guo, Y. Liu, and X. Ki. Bake: A balanced kautz tree structure for peer-to-peer networks. In *Proc. of Infocom*, 2008.

[19] D. Li, X. Lu, and J. Wu. Fissione: A scalable constant degree and low congestion DHT scheme based on Kautz. In *Proc. of Infocom*, 2005.

[20] K. Zuo, D. Hu, H. Wang, Q. Wu, and L. Su. An efficient clustering scheme in mobile peer-to-peer networks. In *Proc.of ICOIN*, 2008.

[21] W. K. Chiang and R. J. Chen. Distributed fault-tolerant routing in kautz networks. *JPDC*, 1994.

[22] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. SPEED: a real-time routing protocol for sensor networks. In *Proc. of ICDCS*, 2003.

[23] E. Felemban, C. G. Lee, E. Ekici, R. Boder, and S. Vural. Probabilistic qos guarantee in reliability and timeliness domains in wireless sensor networks. In *Proc. of Infocom*, 2005.

[24] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He. RAP: a real-time communication architecture for large-scale wireless sensor networks. In *Proc. of RTAS*, 2002.

[25] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proc. of ACM MobiCom*, 2002.

[26] C. P. Ravikumar, T. Rai, and V. Verma. Kautz graphs as attactive logical topologies in multihop lightwave networks. *Elsevier Science*, 1997.

[27] M. Imase, T. Soneoka, and K. Okada. Fault-tolerant processor interconnection networks. *Systems and Computers*, 1986.

[28] B. Han. Zone-based virtual backbone formation in wireless ad hoc networks. *Ad Hoc Network*, 2009.

[29] W. K. Chiang and R. J. Chen. Distributed fault-tolerant routing in Kautz netowrks. In *Proc. of FTDCS*, 1992.

[30] J. L. Gross and J. Yellen. Graph theory and its applications. *Chapman and Hall CRC*, 2006.

[31] C. P. Ravikumar, T. Rai, and V. Verma. Kautz graphs as attractive logical topologies in multihop lightwave networks. *Comp. Comm.*, 1997.

[32] M. Miller. Moore graphs and beyond: a survey of the degree/diameter problem. *The Electronic Journal of Combinatorics*, 2005.

[33] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In *Proc. of STOC*, 1997.

[34] The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[35] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *TON*, 2003.

[36] J. Wu, S. Yang, and M. Cardei. On maintaining sensor-actor connectivity in wireless sensor and actor networks. In *Proc. of Infocom*, 2008.

[37] Linkquest, uwm1000. http://www.link-quest.com/.