A Kautz-Based Wireless Sensor and Actuator Network for Real-Time, Fault-Tolerant and Energy-Efficient Transmission

Haiying Shen, *Senior Member, IEEE,* and Ze Li, *Student Member, IEEE*

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 observed events. Therefore, real-time (i.e., fast) 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 embeds Kautz graphs into the physical topology of a WSAN for real-time communication and connects the Kautz 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 based only on node IDs upon routing failure, rather than relying on retransmission from the source. 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 preserves the consistency between the overlay and physical topology. We further improve routing in REFER by multi-path based routing and energy-efficient multicasting within and between Kautz graph cells, respectively. Extensive experimental results demonstrate the superior performance of REFER in comparison with existing WSAN systems in terms of real-time communication, energy-efficiency, fault-tolerance and scalability.

Index Terms—Wireless sensor and actuator networks (WSANs), routing, Kautz graph, real-time, energy-efficiency, fault-tolerance

1 Introduction

A wireless sensor network (WSN) 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 used for data gathering by sampling surroundings and reporting to predefined data sinks. As hardware technology advances, it is now evolving toward service-oriented wireless sensor and actuator networks (WSANs) [1]. A WSAN consists of sensor nodes capable of measuring stimuli in the environment and actuator nodes capable of affecting their local environment. Similar to WSNs, WSAN sensors usually are low-cost and low-power devices with a short transmission range that are used for the sensing a physical phenomenon. WSAN actuators are resource-rich devices characterized by higher processing and transmission capabilities and a longer battery life. 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 and environment monitoring in battlefields, factories, buildings and cities. For example, smoke detectors (i.e., sensors) deployed in a building report detected fire events to sprinklers (i.e., actuators); Sensors deployed in a battlefield report their detected malicious objects to actuators, which immediately takes action accordingly. Since sensors are densely deployed to ensure the coverage and topology connectivity usually, the scenario we considered in this paper is a highly dense and mobile WSAN which consists of densely populated and possibly mobile sensors.

Actuators need to quickly and reliably respond to nearby sensed events. Delay response may lead to disastrous consequences such as a large loss of life. Therefore, *real-time* (i.e., very fast) 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 [2].

Most of the routing protocols for mobile ad hoc networks (MANETs) and 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. Recently, mesh-based [3], [4] and tree-based [5], [6] systems have been proposed

• *The authors are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634. E-mail: {shenh, zel}@clemson.edu.*

Manuscript received 19 Aug. 2014; revised 19 Jan. 2015; accepted 22 Feb. 2015. Date of publication 26 Feb. 2015; date of current version 1 Dec. 2015. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2015.2407391

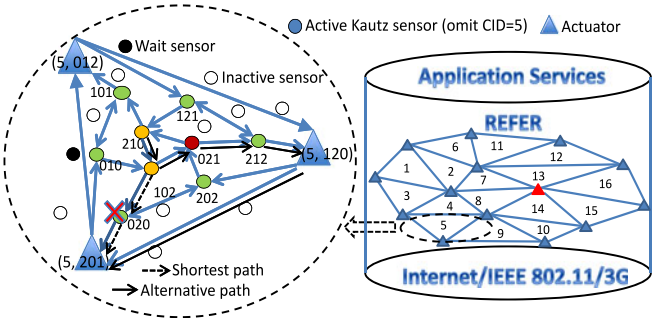


Fig. 1. The architecture of the REFER system.

for data transmission in WSANs. In the mesh-based methods, physically close sensors form a cluster and the cluster head reports their sensed events 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 must retransmit a message upon a routing failure. For example, as shown in Fig. 1, source node 210 wants to send a packet to node 201 and the path is $210 \rightarrow 102 \rightarrow 021 \rightarrow 212 \rightarrow 120 \rightarrow 201$. If node 120 fails, source node 210 must retransmit the packet again. Also, both methods employ either geographical routing [7] or topological routing [8], [9], which consume large amounts of energy by relying on position information generated by GPS or a virtual coordination method [10], [11], [12] or flooding to discover and update routing paths. Therefore, both systems cannot achieve real-time, energy-efficient, fault-tolerant transmissions simultaneously in a highly dense and mobile WSAN.

To meet the requirements of WSANs, we propose a Kautz-based REal-time, Fault-tolerant and EneRgy-efficient WSAN (REFER). Note the "real time" here means fast communication rather than the real time concept in the traditional real-time systems. The contributions of this work include:

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) [13] for high scalability and real-time communication, and an energy-efficient topology maintenance strategy.
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 without data transmission. For instance, in the aforementioned example, forwarder 212 can use an alternative path (e.g., $212 \rightarrow 121 \rightarrow 210$) to continually forward the packet to 201. Further, a multi-path routing algorithm and an energy-efficient multicasting algorithm are proposed for intra- and inter-Kautz cell communication.
4) *Extensive experiments to* demonstrate the superior performance of REFER in comparison with a tree-based, a mesh-based, and a Kautz-based WSAN.

REFER is advantageous over previous 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 maintain topology consistency. Previous works on Kautz graphs directly build a Kautz graph overlay on the application layer in peer-to-peer (P2P) networks [14], [15], [16] or MANETs [17]. 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; however, the previous method [18] has to depend on an energy-consuming routing generation algorithm.

Compared to the early version of this work [19], this version additionally presents the Kautz topology maintenance with a sleep/wake strategy, a multi-path routing protocol to increase the system fault tolerance when the node failure probability is high, and an energy-efficient multicasting algorithm for the transmission of sensed data between Kautz cells.

2 Related Work

WSNs can be regarded as a subcategory of MANETs with 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 [8], [9], [20] or geographic routing [7]. AODV [8] and DSR [20] are reactive routing protocols, in which a source node broadcasts a query to find a path to the destination. DSDV [9] is a proactive routing algorithm, in which each node maintains and periodically updates a routing table by message flooding. Keeping a complete routing table reduces route acquisition latency for data transmission. However, its correct operation depends on its periodical global dissemination of connectivity information, leading to low scalability. Further, 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 [10], [11], [12], both consume a considerable amount of energy, which are also not suitable for WSANs.

Many routing algorithms have been proposed for WSNs. Pöttner et al. [21] introduced a heuristic to calculate a schedule that can support the given application requirements in terms of data delivery latency and reliability and described methods and tools to collect the data necessary as input for schedule calculation. He et al. [22] investigated the on-demand scenario where data collection requests arrive at the mobile element progressively, and modelled the data collection process as a queuing system. Based on this model, the authors evaluated the performance of data collection through both theoretical analysis and extensive simulation. Ji et al. [23] introduced a more reasonable model, probabilistic network model, for the network capacity of data

collection. They proposed a cell-based path scheduling (CPS) algorithm for snapshot data collection, and proposed a zone-based pipeline scheduling (ZPS) algorithm for continuous data collection. Hsu et al. [24] proposed a joint design of asynchronous sleep-wake schedules and opportunistic routing to maximize the network lifetime. Ma et al. [25] identified the contiguous link scheduling problem in WSNs, in which each node is assigned consecutive time slots so that the node can wake up only once in a scheduling period to fulfil its data collection task, and presented efficient centralized and distributed algorithms. Since these methods are for general WSNs, they may be not suitable for WSANs [3] that have different features on the network structure and communication paradigms between sensors and actuators.

A number of routing protocols have been proposed specifically for WSANs. Melodia et al. [6] 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 using the geographical routing. Hu et al. [5] 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 using the topological routing. The tree structure is not fault-resilient to node mobility since a parent failure prevents its children from sending or receiving data in time. Ngai et al. [3] and Shah et al. [4] proposed a distributed protocol to form sensors into clusters. The cluster heads form a backbone mesh network to provide routes toward actuators. Rezgui and Eltoweissy [26] presented reliable adaptive service-driven efficient routing (μRACER), a routing protocol suite based on a novel service-oriented design for sensor-actuator networks where nodes expose their capabilities to applications as a service profile. Pan et al. [27] proposed a geometry-based path-planning algorithm (GPA) to plan a route that can navigate an aircraft to position a sensor network using the shortest time in sensor deployment. Vazifehdan et al. [28] proposed two energy-aware routing algorithms for WANET, called reliable minimum energy cost routing (RMECR) and reliable minimum energy routing (RMER). However, the above methods need to retransmit a message from the source to the destination upon a routing failure, generating a certain delay. Also, most of these methods are not energy-efficient 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 [14], [15], [16]. Zuo et al. [17] 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. Ravikumar et al. [29] and Li et al. [16] studied the shortest and longest path routing. In BAKE [15] and DFTR [18], 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. [30] 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.

3 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 scalability, transmission speed and energy-efficiency [31]. A well-designed overlay should be energy-efficient in topology maintenance, resilient to node mobility, and enables efficient and reliable routing. With this objective, we present the applicability of the Kautz graph topology to the WSAN overlay (Section 3.1), the Kautz graph embedding protocol (Section 3.2) and the efficient fault-tolerant routing protocol (Section 3.3).

3.1 Is Kautz Graph a 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 a reasonable overlay topology that achieves a tradeoff between degree and diameter for WSANs.

Defination 1 [32]. *In a Kautz graph $K(d,k)$ with degree d and diameter k, nodes are labeled as $(u_1 \dots u_k)$, where u_i belongs to an alphabet of $d+1$ letters $(A=(0,1,,d))$, and $u_i \neq u_{i+1}$ $(1 \le i \le k)$. 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 Fig. 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 [33].

Defination 2 [33]. *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 [33] shows that for a graph, $|E(G)| \ge N(G)\delta_{min}(G)$ 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 $N(G) = n = (d+1)d^{k-1}$ [32]. □

It has been proved that the Kautz graph has a smaller diameter than the de-Bruijn and hypercube topologies [29], which have been widely studied as promising overlay topologies [16]. Based on this finding, Definition 2 and Lemma 3.1, we can get Proposition 3.1 below.

Proposition 3.1. *A Kautz graph $K(d,k)$ can help solve the graph connection optimization problem by achieving a tradeoff between degree and diameter with its minimum degree and relatively shorter diameter.*

Therefore, Kautz graph is a reasonable topology for WSAN overlays to meet the energy-efficiency and real-time requirements. The second issue that needs to consider in designing a WSAN overlay 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 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 topology 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 [33], in which a path traverses through every vertex in the graph exactly once before returning to the starting vertex. In order to achieve topology 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. $d \geq \frac{n}{2}$ 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 [33]. With the assumption that nodes are independent and identically distributed (i.i.d.), 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

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

□

This 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. Also, this proposition indicates 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 and ensure the connectivity of Kautz cells. The awake/sleep scheme will be presented in Section 3.2.4.

3.2 Kautz Graph Embedding Protocol

Proposition 3.2 indicates that the physically close sensors need to be grouped into cells. Each cell is composed by actuators and sensors. REFER embeds a Kautz graph into each cell. It has been proved [14] that as the diameter k decreases, the number of nodes in Kautz graph $K(d,k)$ approaches the Moore bound [34]. That is, node density in $K(d,k)$ increases as k decreases. Therefore, a Kautz graph with a smaller diameter k should be a 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 cell and k, the value d can be determined. d is the smallest number that generates the value closest to but smaller than n based on this equation. Here, we choose the Kautz graph $K(2,3)$ 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 in applications (e.g., habitat monitoring [6], battlefield monitoring [1]).

Fig. 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. 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. The DHT facilitates the information transmission between cells. DHTs are well-known for their scalability and dynamism-resilience. In addition, without relying on energy-inefficient topological routing or geographic routing, the constructed DHT preserves the physical topology to enable the DHT routing algorithm to transmit data along physically close actuators to its destination, thus leading to fast and energy-efficient routing.

Each WSAN cell has a cell ID (CID) (e.g, 1-16 in Fig. 1). Each node in a cell with CID has ID = (CID, KID), where KID = $\{u_1 \dots u_i \dots u_k \mid u_i \in \mathcal{A},\ u_i \neq u_{i+1}\}$ (e.g, 201) is the Kautz ID in the Kautz graph, where alphabet $\mathcal{A}$ contains $d+1$ distinct symbols. For a pair of nodes $U = u_1u_2 \cdots u_k$ and $V = v_1v_2 \cdots v_k$, 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, and the shortest distance between Kautz nodes U and V in one cell $P(U,V) = k - L(U,V)$. 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. We present the details of each step below.

3.2.1 Actuator ID Assignment

The actuator ID assignment process detects triangles among the neighboring actuators and sequentially assigns IDs to the actuators. For this purpose, we first introduce a distributed method for a large-scale network and then introduce a centralized method for a small-scale network. Note that this process is only conducted once right after the initial system deployment. We use the triangulation algorithm in [35] to construct actuators to triangles.

In the distributed method, neighboring actuators exchange the information with each other. We first select one node as a starting node. The starting node finds its nearest neighbor and builds an edge to it. The two nodes connected by this edge (edged nodes) choose a nearby node to form a triangle (cell) so that the triangle's circumcircle is minimized. We call this node the nearest neighbor of the edged nodes or the edge. To find the nearest neighbor on one side of edge e_{ij} connecting nodes, s_i and s_j, these edged nodes communicate with each other to determine their shared neighbors. For each shared neighbor s_{k_m}, the perpendicular bisectors of e_{ik_m} and e_{jk_m} intersect with the perpendicular bisector of e_{ij} at a point. Then, the shared neighbor that produces the least distance between such a point and e_{ij} is the nearest neighbor (denoted by s_k) of e_{ij}. Next, s_i and s_j connect to s_k to form triangle $\triangle s_i s_j s_k$ and generate CID $= 1$ for this triangle. Then, the edged nodes conduct the same operation by finding their nearest neighbor to construct a new triangle with the minimum circumcircle and assign the triangle CID $= 2$. This process repeats until the triangulation completes. Thus, closer cells have closer CIDs. The used CIDs are propagated along the nodes as the triangulation proceeds to ensure that the same CID will not be used more than once.

In the centralized method, each actuator A has a value $H(A)$; the consistent hash value [36] of the actuator's 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 after a sufficient time period. To avoid redundant information exchanges, a node does not send out redundant information it has received. 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 using the same process explained previously.

In both distributed and centralized methods, after a triangle is formed, the KIDs for the actuators in the triangle are then determined. Neighboring actuators cannot have the same KID since they are in the same cell. For this purpose, we employ the sequential vertex-coloring algorithm [33], 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). For example, in Fig. 1, the IDs of the actuators in cell 5 are $(5, 201)$, $(5, 120)$ and $(5, 012)$. For simplicity, we can also use a traditional way in sensor networks to assign IDs to actuators; that is, the IDs of actuators are predefined before they are deployed.

3.2.2 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. Algorithm 1 shows the KID assignment algorithm to select Kautz nodes to construct a Kautz graph. It has three steps:

1) the Kautz nodes connecting the actuators in a Kautz graph are selected (Lines 1-8),
2) the Kautz nodes connecting these selected Kautz nodes are selected (Lines 9-16), and
3) the remaining Kautz nodes in the Kautz graph are selected (Lines 17-18).

For KID $= kid$, we use kid_l to denote the KID after left rotating kid once. The successor actuator of actuator kid, denoted by $suc(kid)$, is the actuator that has KID $= kid_l$ in the same Kautz graph. For example, in Fig. 1, $(5, 120) \rightharpoonup (5, 201) \rightharpoonup (5, 012) \rightharpoonup (5, 120)$, where $\rightharpoonup$ here means successor actuator.

Algorithm 1. Pseudo-code for (CID, KID) assignment on sensors (i.e., Kautz graph construction)

```
1:  /*Executed by an actuator for selecting the Kautz nodes to
    connect to its successor actuators*/
2:  if no Kautz nodes have been selected to connect to the
    successor actuator with the same CID then
3:     Broadcast a path query to its neighbors with TTL = 2
4:  end if
5:  if received a path query then
6:     Choose the sensors on the path with the highest
       aggregated power as Kautz nodes
7:     Notify the selected Kautz nodes of their assigned
       (CID, KID); KID is sequentially assigned to them
8:  end if
9:  /*Executed by Kautz node sensor s_i for selecting Kautz nodes to
    connect to other selected Kautz nodes*/
10: if s_i is the successor of an actuator and
    H(s_i) = min(H(s ∈ K)) then
11:    Send a query to the Kautz node s_j that is the
       predecessor of the actuator and
       H(s_j) = max(H(s ∈ K)) with TTL = 2
12: end if
13: if received a query then
14:    Choose the sensors on the path that has not been built
       and has the highest aggregated power as Kautz nodes
15:    Notify the selected Kautz nodes of their assigned
       (CID, KID); KID is sequentially assigned to them
16: end if
17: //Select the remaining Kautz nodes in the Kautz graph
18: Build the connectivity to its successor Kautz node
```

In the first step, each actuator selects sensors with high energy to connect to its successor actuator. For example,

actuator $(5, 201)$ finds sensors to connect itself to its successor actuator $(5, 012)$. Actuator kid (e.g., $(5, 201)$) broadcasts a path query message towards actuator $suc(kid)$ (e.g., $(5, 012)$) in the same cell with TTL (Time to live) $= 2$, 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, actuator $suc(kid)$ (e.g., $(5, 012)$) receives a number of messages during a certain time period. It selects a path with the highest accumulated energy, and assigns (CID, KID) to the sensors in the path. Each sensor's CID equals to the actuator's CID. The KID is sequentially assigned to the sensors. If a sensor's predecessor in the path has KID $= (u_1u_2u_3)$, its KID is then $(u_2u_3u_k)$ where u_k makes $u_2u_3u_k$ close to kid_l. For example, actuator $(5, 012)$ assigns IDs $(5, 010)$ and $(5, 101)$ to the two sensors 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 $(5, 012) \rightarrow (5, 121) \rightarrow (5, 212) \rightarrow (5, 120)$. In Fig. 1, we mark these identified sensors connecting actuators in green color. To avoid the case that a sensor is selected for different paths connecting different pairs of actuators, the selected sensors must be approved by the two end actuators and one sensor can only be selected once. If an actuator receives two approval requests simultaneously, it will handle one request first to avoid the collision.

An actuator's successor or predecessor is the sensor that succeeding or proceeding itself in the path. For example, the successor of actuator $(5, 012)$ is sensor $(5, 121)$, and the predecessor of actuator $(5, 201)$ is sensor $(5, 020)$. In the second step, we select sensors (orange nodes in Fig. 1) to ensure that a message can traverse the path between each pair of Kautz nodes. Specifically, we first choose the successor of the actuator with the smallest KID$= u_1u_2u_3 = 012$ (i.e., $S_i = u_2u_3u_2 = 121$), and the predecessor of the actuator with the largest KID $= u_3u_1u_2$=201 (i.e., $S_j = u_1u_3u_1 = 020$). Then, 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. In the third step, using the same method, each selected Kautz node selects other Kautz nodes to build the connectivity to its successor Kautz node. For example, Kautz nodes 210 and 102 find the node with the highest battery power (the brown node in Fig. 1) to connect themselves, and assign it with KID $= u_1u_3u_2 = 021$. Finally, all the Kautz nodes are selected to construct a Kautz graph.

3.2.3 DHT-Based Upper Tier Structure

CAN [13] 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.

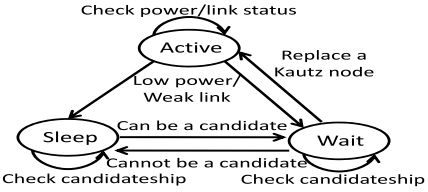


Fig. 2. Three states of sensors.

3.2.4 Energy-Efficient Topology Maintenance

In a large-scale WSAN where the number of low-price sensors deployed in a target area is in the order of hundreds or thousands, it is not necessary for all sensors to be active and involved in data transmission. The sensors are usually operated in duty-cycle with awake and sleeping periods to save energy [2]. As shown in Fig. 2, REFER sets three functional states for sensors to maintain the Kautz overlay: *active*, *wait* and *sleep*. Active nodes form a Kautz graph, and they periodically exchange "Hello" messages to check the connectivity with their neighboring active nodes. Like other wireless networks, the frequency for the "Hello" message exchanges is determined by the application needs. A higher frequency can ensure the connectivity of the network but generates a high overhead and vice versa. The sleep scheduling in REFER is based on sensor energy level and the connectivity to other Kautz nodes. Each node in the sleeping mode periodically wakes up to see if it can be a backup candidate for a Kautz node. The selected candidate nodes stay in the *wait* state. A candidate node should meet three requirements. First, it must be able to communicate with a Kautz node. Second, it has at least one neighboring Kautz node with battery power below a safety threshold or with a link weaker than a threshold. Third, it can build connections with the neighboring Kautz nodes of this Kautz node. Otherwise, it cannot replace this node for an intact Kautz graph. A node can check if it meets these requirements by contacting its neighbors. When a Kautz node notices that its links to its current neighbors are about to break or its battery power is below a safety threshold, it selects one of its current candidate nodes to replace itself by passing its (CID, KID) to the candidate and then goes to sleep. The candidate then notifies its predecessor and successor in the Kautz graph, and functions as a Kautz node and provides data forwarding service. A node may be disconnected with its neighbors or out of power before it is replaced. To handle this case, if a node notices that its connectivity with another node is broken, it uses the previously introduced method to re-establish its failed neighbor. This ensures the connectivity of the Kautz graph.

3.3 Efficient Fault-Tolerant Routing Protocol

Communication between sensors consumes high energy [2]. A tradeoff exists between fault-tolerance/real-time and energy consumption in routing. A Kautz graph can help to achieve a reasonable tradeoff. A Kautz graph with degree d has d disjoint paths between any two nodes [29]. This topology feature supports fault-tolerant routing protocols [15], [18], in which if a node fails to forward a message along the

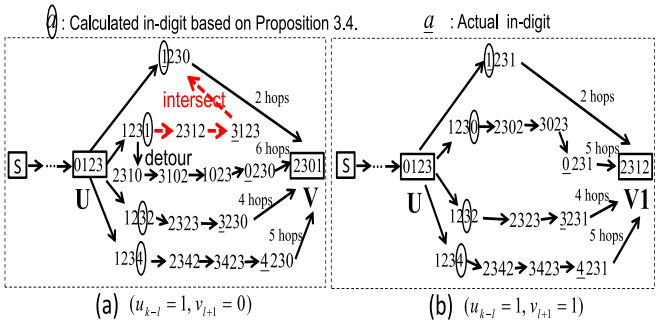


Fig. 3. Examples of routing paths in a Kautz graph.

shortest path, it can choose the successor in the second-shortest path, then third-shortest path, and so on. For example, Fig. 3a illustrates the 4 paths between node 0123 and node 2301 in Kautz graph $K(4,4)$. After node 0123 initiates or receives a message destined to node 2301, if node 1230 in the shortest path fails to forward the message, 0123 chooses the successor in the next shortest path, say node 1232. In Fig. 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 [18], 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.

3.3.1 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 appending 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 $\beta_i v_1 v_2 \ldots v_{k-1} (v_1 \neq \beta_i)$. 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 $[0,d]$.

Defination 3. *For a U-V pair (Fig. 4), 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.

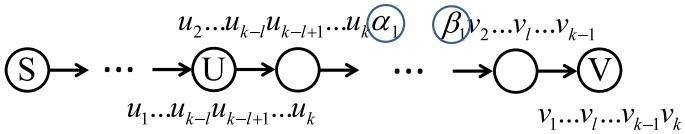


Fig. 4. An example of a path between a pair of nodes.

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 Fig. 3a, 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 $\beta v_1 v_2\ldots v_{k-1}$ 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 $\beta v_1 v_2\ldots v_{k-1}$ using the greedy shortest protocol. □

Example for Proposition 3.4. Considering the U-V pair in Fig. 3a, 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 Fig. 3a, 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: nodes 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 $u_{k-l} = 1$. Nodes 1230 and 1231 have the same in-digit 1, the paths traversing them using the greedy shortest protocol intersect at 1230. In Fig. 3b, 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: nodes 1230, 1231, 1232 and 1234. Since node 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.

Defination 4. *For a U-V pair with $u_{k-l} \neq v_{l+1}$, the U's successor $u_2u_3 \ldots u_ku_{k-l}$ 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 $u_2u_3 \ldots u_ku_{k-l}$ should forward data to node $u_3u_4 \ldots u_ku_{k-l}v_{l+1}$ in order to avoid intersection with the shortest path.*

Proof. In addition to the conflict node $u_2u_3 \ldots u_ku_{k-l}$, other successors of U are $u_2u_3u_{k-l}u_k\alpha_i(\alpha_i \prec [0, d] \ \& \ \alpha_i \neq v_{l+1})$ 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 Fig. 3a, 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 $U = u_1u_2 \ldots u_k$ forwards data to node $V = v_1v_2 \ldots v_k$, the successor, path length and corresponding condition of the d disjoint U-V paths are*

$$\begin{cases} (1)\ u_2 \ldots u_ku_{k-l}; & k+2, & \text{when } u_{k-l} \neq v_{l+1}; \\ (2)\ u_2 \ldots u_{k-l} \ldots u_kv_{l+1}; & k-l, & \text{the shortest path}; \\ (3)\ u_2 \ldots u_kv_1; & k, & \text{when } u_k \neq v_1; \\ (4)\ u_2 \ldots u_k\alpha_i; & k+1, & \text{otherwise}, \end{cases}$$

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 $k + 1$. □

3.3.2 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 [18]). Algorithm 2 shows the fault-tolerant routing algorithm in a Kautz graph 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 (as in 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 found. 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 [8], 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 Fig. 3a, 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 is 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 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 P2P routing protocol, in which a node forwards the data to its neighbor closest to the destination. For example, in Fig. 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 and energy-efficiency.

Algorithm 2. Pseudo-code for the fault-tolerant routing algorithm in a Kautz graph

```
/*Executed by every node in the network*/
while initiate or receive data to forward do
    Identify the multiple paths to the destination based on Theorem 3.8
    Order the successors of the paths in the ascending order of the path length
    Select the successor in the ordered list that is capable of forwarding the data
    Transmit the data to the selected successor
end while
```

Since the alternative paths are not as short as the shortest path length, taking alternative paths with more sensors would consume more sensor energy. In the Kautz graph, the maximum path length is $k+2$ and the minimum path length is $k-l$, and there are at least $d-3$ paths with transmission lengths equal to $k+1$. Using the longest path increases the length of the shortest path by $(k+2)-(k-l)=(2+l) \in [2,5]$ hops for the $K(d,3)$ Kautz graph. We see that the path increase and hence the additional energy consumption are not significant.

3.3.3 Multi-Path Based Routing Within a Kautz Cell

Although the Kautz-based routing algorithm provides high fault tolerance, if the failure probability of the mobile nodes is high, an alternative path will always be used, which greatly increases the transmission delay. With the assumption of the i.i.d. distribution, suppose the failure probability of each node is p, then the transmission success probability of a path between (U, V) with length c hops is $P(1)=(1-p)^c$. Given a constant c, as p increases, $P(1)$ decreases polynomially. Given a constant p, as c increases, $P(1)$ also decreases exponentially. In the Kautz-based routing, as p increases, c also increases because an alternative path with a longer path is used, then $P(1)$ decreases extremely fast, leading to a high packet transmission dropping rate. To enhance the routing fault tolerance of the Kautz routing protocol when the node failure probability is high, the source node can use all the shortest paths to all the actuators in its cell for the packet routing. If this method is still not sufficient to handle the problem, the source node can use a multi-path routing protocol. Recall that a Kautz graph with degree d has d disjoint paths between any two nodes. When a source node suffers a large packet dropping rate or long packet transmission delay, it chooses to use multiple top-shortest paths for packet routing. The other packet receivers still follow Algorithm 2 in packet forwarding. Let's use $\bar{c}$ to denote the average path length of the d disjoint paths of a routing. Then, given the average success probability of a single path routing $\bar{P}(1)=(1-p)^{\bar{c}}$, the success probability of the multi-path based routing protocol with b multiple paths is

$$P(b)=1-(1-\bar{P}(1))^b=1-(1-(1-p)^{\bar{c}})^b. \quad (2)$$

The equation shows that given a constant $\bar{P}(1)$, as b increases, $P(b)$ increases. When $b=2$, $P(2)=1-(1-(1-p)^{\bar{c}})^2=(2(1-p)^{\bar{c}}-(1-p)^{2\bar{c}})=(1-p)^{\bar{c}}(2-(1-p)^{\bar{c}})$. When $p=0.5$ and $\bar{c}=4$, the success probability of $P(2)$ is almost twice of $P(1)$. Though larger b enhances the fault tolerance more, the value of b cannot be too large because transmitting more packets will generate more interference and hence reduce the throughput and increase delay. It also leads to more energy consumption. Thus, it is important to determine b that achieves an optimal tradeoff between throughput, delay, energy consumption and transmission cost.

3.3.4 Energy-Efficient Multicasting between Kautz Cells

In a WSAN, when a sensor detects an event, it may need to notify a number of actuators to collaboratively handle the event. It is indicated that the energy consumed by forwarding sensed data between two nodes is twice the energy consumed for communication overhead (such as hello messages and control messages) between two nodes [37]. Thus, when there are many destinations, using CAN routing algorithm to send sensed data to multiple destinations may consume much energy.

When the actuator of a source sensor (i.e., source actuator) multicasts sensed data to multiple destination nodes, in order to reduce the energy consumption in data forwarding, it can use multicasting and reduce the number of data forwarding operations between nodes. Thus, we propose a multicasting algorithm, in which each actuator builds a tree and multicasts data to the destinations using the minimum number of forwarding operations.

As we mentioned in Section 3.2, neighboring actuators exchange the information of their neighbors to build and maintain the Kautz-based topology. Finally, each actuator can learn the global topology of actuators and the hop distance between actuators. Based on the global topology, each node treats itself as root node and builds a minimum spanning tree (as shown in Fig. 5) using the Prim's algorithm [38]. In the tree, the weight of an edge between two actuators equals their hop distance. The process of building such a tree is the process for a node to iteratively find the closest nodes to itself and connects to the nodes. The minimum spanning tree is characterized by the feature that the sum of the weights from all other nodes to the root node is the minimum. This means that a node sending sensed data

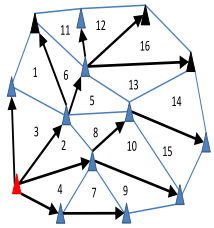


Fig. 5. A minimum spanning tree.

to all other nodes along the tree minimizes the energy consumption. Algorithm 3 shows the pseudo-code of building the minimum spanning tree and the multicasting algorithm. Before an actuator forwards sensed data to several destination actuators, based on its minimum spanning tree, the actuator figures out the routes to forward the message that leads to the minimum forwarding operations. It first finds the forwarding path for each destination and then combines the common paths of different destinations. The source actuator piggybacks the path information on the data, so the data receivers forward data based the path information. That is, the data is forwarded along one path and is replicated when it must be forwarded along different paths to the destination nodes in the tree.

Algorithm 3. Pseudo-code for minimum spanning tree construction and the multicasting algorithm

```
/*Executed by every actuator v_i in the network*/
Learn the global topology of actuators (V, E) by information exchange
Initialize its minimum spanning tree (V_min, E_min) (V_min = {∅}, E_min = {∅})
for each vertex u in graph (V, E) do
    u.key = ∞ and u.minV = null
end for
v_i.key = 0
Enqueue all v ∈ V to minimum-heap Q sorted by the key
while Q is not empty do
    //find (v ∈ Q, u ∉ Q) with minimal weight(v, u) and add v and E(u, v) to (V_min, E_min)
    Dequeue vertex v with the minimal key from Q
    Add v and E(v, v.minV) to (V_min, E_min)
    for each adjacent vertex u of v do
        u.key = weight(v, u)
        if (u ∈ Q) and (weight(v, u)<u.key) then
            u.key = weight(v, u), u.minV = v
        end if
    end for
end while
/* Executed by actuator v_i in sending sensed data*/
//Calculate paths to the destination nodes
Find the forwarding path for each destination
Combine the common forwarding path of different destinations
Send the data to the next hops in the paths along with the calculated paths
/* Executed by actuator v_i in forwarding sensed data*/
Forward received data to the next hops specified in the received data
```

Fig. 6 shows an example of the comparison of the P2P routing versus the multicasting algorithm. We see that to

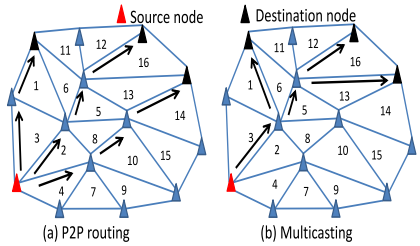


Fig. 6. Forwarding sensed data between Kautz cells.

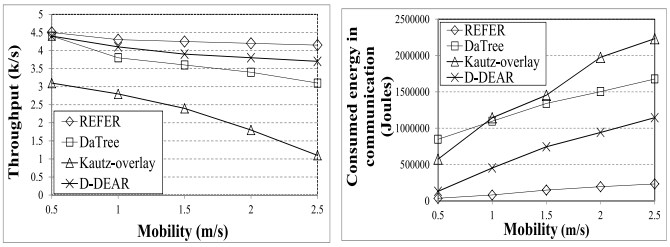
forward data to three different destination nodes, the P2P routing needs 8 forwarding operations while the multicasting only needs five forwarding operations, which greatly reduces the energy consumption for transmission overhead in the system. If there is a change in the global topology, affected nodes update their minimum spanning trees accordingly. Since the actuators are comparably stable in our study model, the trees would not be updated frequently. For urgent events that need to quickly notify the destination actuators, we can use flooding instead of multicasting.

4 Performance Evaluation

We used NS-2 [39] to evaluate the performance of REFER in comparison with the DaTree [6] tree-based system, the D-DEAR [4] mesh-based system, and the Kautz-based overlay [17] (denoted by Kautz-overlay) for WSANs. To make the systems comparable, we use the topological routing in [40] 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 broadcasts 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.

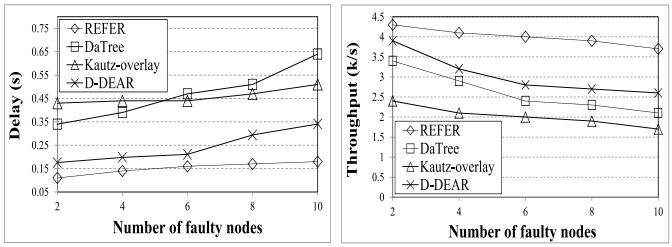
Unless otherwise specified, five actuators were uniformly distributed in a 500×500 m area and 200 sensors were i.i.d. distributed around the actuators, which form 4 $K(2,3)$ Kautz cells. Such simulation scenario is similar to that in [41]. The transmission ranges of sensors and actuators were set to 100 and 250 m, respectively. Every 10 seconds, we randomly chose five source nodes to transmit data to their nearby actuators.

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, unless otherwise specified. The warmup time and simulation time were set to 100 and 1000 s, respectively. Since most packets can arrive at the destinations within 1s, we only counted those arriving at the destination within 0.6 s into the throughput in order



(a) Throughput vs. mobility (b) Energy consumed vs. mobility

Fig. 7. Performance of mobility resilience.



(a) Delay vs. # of faulty nodes (b) Throughput vs. # of faulty nodes

Fig. 8. Performance of fault-tolerant routing.

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 [42], respectively. All experimental results report the 95 percent confidence intervals. We use the following metrics for the performance evaluation: (1) *Throughput*. The size of received data by all actuators per second. 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.

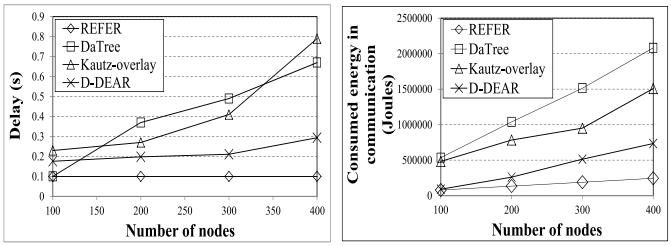
4.1 Mobility Resilience

In this experiment, a node's speed was randomly selected from $[0, 5]$ m/s. Fig. 7a 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. 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 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.

Fig. 7b 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. By avoiding message retransmission and maintaining 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 needs to maintain multiple consecutive multi-hop paths, it consumes much more energy than DaTree in a highly mobile environment. It is interesting to see that when mobility is 0.5 m/s, Kautz-overlay consumes less energy than DaTree. Since fewer path updates and message retransmissions occur in a low mobile environment, Kautz-overlay consumes less energy than DaTree.

4.2 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 $10\,s$ and recovered the previous set of faulty nodes. The number of faulty nodes was set to $2\,x$, where x is randomly chosen from $[1, 5]$. Fig. 8a 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 enables 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



(a) Delay vs. # of total nodes (b) Energy consumed vs. # of nodes

Fig. 9. Performance in real-time and scalable transmission.

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.

Fig. 8b shows the throughput of systems versus the number of faulty nodes. It shows that the throughput of all systems decreases as the number of faulty nodes grows. 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 that of DaTree and D-DEAR due to their fault-tolerant routing as explained in Fig. 8a. In DaTree and D-DEAR, upon a routing failure due to faulty nodes, the delay from path re-establishment reduces the throughput during the simulation time. Faulty nodes only affect the paths between the cluster heads and actuators in D-DEAR, but affect the paths between all sensors and the actuators in DaTree. Therefore, D-DEAR generates higher throughput than DaTree. Because Kautz-overlay produces a much longer transmission path for one message transmission than all other systems, it produces the least throughput during the limited simulation time.

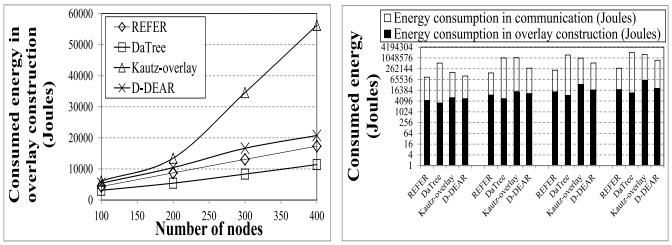
4.3 Real-Time Transmission

Fig. 9a 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 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 nearly 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, 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.

4.4 Scalability and Energy-Efficiency

Fig. 9b demonstrates the energy consumed in communication for each system versus network size. Here, as the network size increases, 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. This increases the probability that a path is broken and hence triggers more path updates. Thus, the consumed energy of these systems increases quickly. We also observe that 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 moderately mobile environment, message transmission dominates the influence on the energy consumed due to fewer 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 Fig. 7b.

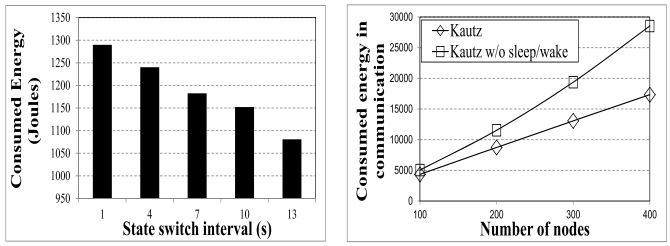
Fig. 10a shows the energy consumed in topology construction for each system versus the network size. We can see that Kautz-overlay consumes the most energy for overlay construction. The reason is that every node in Kautz-overlay needs to use broadcasting to build a multi-hop path to each of its overlay neighbor. As the overlay in both D-DEAR and REFER are formed by physically close nodes,



(a) Energy consumed in overlay construction

(b) Total energy consumed

Fig. 10. Performance in energy-efficiency.



(a) Topology maintenance energy consumption vs. state switch interval

(b) Topology maintenance energy consumption with and without the sleep/wake strategy

Fig. 11. Performance of topology maintenance.

they consume less energy. In D-DEAR, since every node locally contacts neighbors within two hops to select its cluster head, it consumes less energy than Kautz-overlay. In REFER, actuators need to exchange information and broadcast messages to all nodes in the cells for actuator ID assignment. Also, communications between actuators and sensors are needed for selecting Kautz nodes. 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. Fig. 10b combines the energy consumed for communication and topology construction. We notice that topology construction consumes negligible energy compared to that of communication (0.1 percent). Thus, the result confirms that REFER is energy-efficient in terms of total energy consumption.

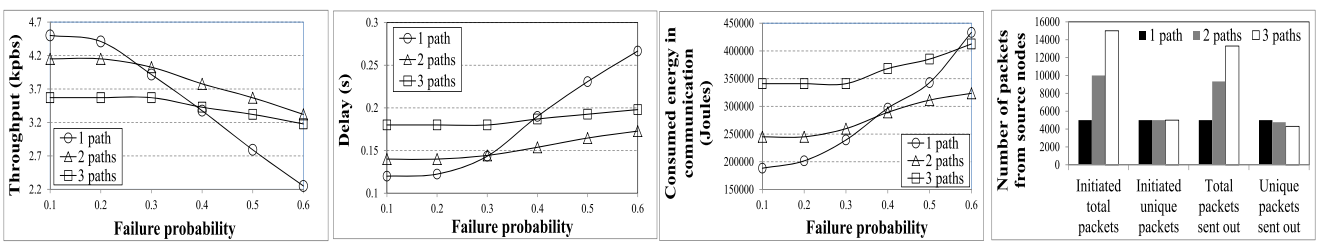
4.5 Evaluation of Topology Maintenance

In this section, we further evaluate the effectiveness of topology management in Kautz cells. We define the *state switch interval* as the time period a node stays in one state before it changes to another state. A node can change its state from "active" to "wait", "wait" to "sleep" or "sleep" to "active". Fig. 11a shows the consumed energy in topology maintenance versus the state switch interval. We see that as the state switch interval increases, the topology management overhead decreases, leading to an energy consumption decrease in topology management. When the state switch interval is small, the nodes periodically become active with a high frequency and exchange "Hello" messages with neighboring nodes, which consumes high energy. When the state switch interval becomes larger, the nodes are in the sleep states for a longer time period to save energy, leading to less energy consumption.

Fig. 11b further shows the consumed energy in topology maintenance in Kautz cells with and without the awake/sleep strategy, respectively. As the number of nodes in the system increases, more energy is consumed in node communication for topology maintenance. The reason is that more nodes in the system generate more "Hello" messages exchanged between neighbors in the system, which consumes high energy. The figure also shows that with the awake/sleep strategy, the consumed energy decreases. This is because with the asleep/wake strategy, some nodes can stay in the sleep state without "Hello" message exchanges to save their energy.

4.6 Evaluation of Multi-Path Based Routing

In this section, we evaluate the performance of the multi-path based routing protocol of REFER. If all alternative paths fail to deliver a packet, the packet is dropped. Here, we define the *throughput* as the size of successfully delivered unique packets during the simulation time. A packet with multiple successfully delivered copies was only counted once. We varied the failure probability of Kautz nodes from 0.1 to 0.4. Fig. 12a shows the throughput of the unique packets versus failure probability of the Kautz nodes. We see that as the failure probability increases, the throughput decreases. When the nodes fail with a high probability, even though a packet can use an alternative path for routing, the packet still has a high probability to be dropped. The figure also shows that when the failure probability is small, single-path routing produces higher throughput than multi-path routing. This is because nodes in multi-path routing suffer high transmission interference with each other when nodes in different paths forward the packets at the same time. However, we see that multi-path routing has much lower throughput decreasing rate. Also, when the packet failure probability is larger than 0.3 and 0.4,



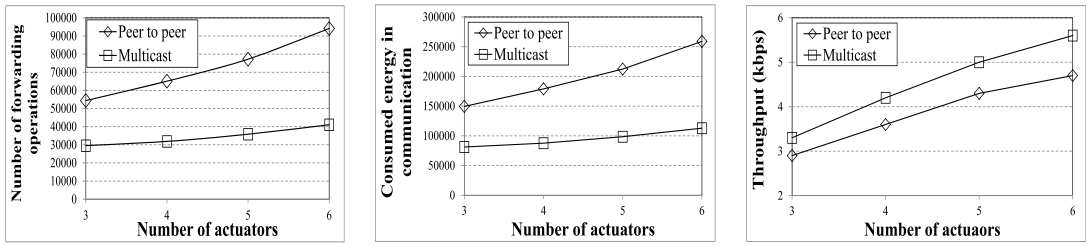
(a) Throughput vs. failure probability

(b) Delay vs. failure probability

(c) Energy consumption vs. failure probability

(d) Energy consumption vs. failure probability

Fig. 12. Performance of the multi-path based routing.



(a) Number of forwarding operations (b) Energy consumed (c) Throughput

Fig. 13. Performance of the energy-efficient multicasting algorithm for inter-Kautz cell communication.

two-path and three-path routing produce higher throughput than single path routing, respectively. This result confirms the effectiveness of multi-path routing in increasing the probability of successfully delivering a copy of a packet to its destination according to Equation (2). Because of the highest interference in three-path routing, it produces the longest delay and hence the lowest throughput in most cases. These experimental results imply that two-path routing is the best routing algorithm in terms of throughput when the failure probability is larger than 0.3.

We then define the *delay* of a packet as the delay of the first successfully delivered copy among all copies of the packet. We do not consider the unsuccessfully delivered packets in calculating delay. Fig. 12b shows the delay per unique packet versus failure probability of the Kautz nodes. We see that as the failure probability of the nodes increases, the delay of the unique packet increases. This is because routing along an alternative path for a dropped packet leads to longer transmission delay. Similar to Fig. 12a, because of the transmission interference between nodes and resultant alternative path usage in multi-path routing, multi-path routing has longer delay than single-path routing when failure probability is small, but it has a much smaller delay increase rate as failure probability increases. This is due to the reason that in multi-path routing, a copy of a packet has higher probability to be delivered to the destination according to Equation (2). This is also why when failure probability is larger than 0.3, two-path routing has lower delay than single-path routing. Although three-path routing is the most fault-tolerant, the higher interference in transmission and more nodes competing for channel resources lead to higher transmission delay. These experimental results imply that two-path routing is the best routing algorithm in terms of delay when the failure probability is larger than 0.3.

Fig. 12c illustrates the energy consumption during packet transmission versus failure probability of Kautz nodes. We see that as the failure probability increases, the energy consumption of packet transmission increases. A higher Kautz node failure probability increases the probability of transmission along alternative paths, leading to more energy consumed for transmission. In multi-path routing, the source node sends redundant copies of packets, which consumes higher energy with more copies. However, as multi-path routing can save energy in packet retransmission of the dropped packets in alternative paths, two-path routing leads to less energy consumption than single-path routing when the failure probability of the nodes is larger than 0.4. Three-path routing consumes the highest energy in most cases due to its much more packets in routing and much more retransmissions caused by interference.

A source node buffers its initiated packets until its channel is idle. Fig. 12d shows the number of all packets and unique packets initiated from and successfully sent out from source nodes during the simulation time. Because two- and three-path routing make two and three copies of an initiated packet, respectively, the number of all packets initiated from source nodes follows 1-path $<$ 2-path $<$ 3-path though their number of unique packets is the same. Because a source uses multiple forwarders in the multi-path routing algorithm, the number of total packets successfully sent out from source nodes follows 1-path $<$ 2-path $<$ 3-path. Since transmitting more packets leads to longer packet buffering time, the number of unique packets successfully sent out from source nodes follows 1-path $>$ 2-path $>$ 3-path. This result implies that if the number of multiple paths is too high, the initiated unique packets will be delayed at source nodes.

In conclusion, for the $K(2,3)$ Kautz graph, from Figs. 12a, 12b and 12c, we can find that two-path routing can achieve an optimal tradeoff between throughput, delay and energy consumption of transmission in our experimental environment.

4.7 Evaluation of the Energy-Efficient Multicasting

In this section, we evaluate the performance of the energy-efficient multicasting algorithm for inter-Kautz cell communication. In every 10 seconds, we randomly chose 1 source sensor node to transmit data to m randomly selected actuators from 6 actuators, and m was varied from 3 to 6 with 1 increase in each step.

Fig. 13a shows the number of forwarding operations between two nodes occurred in transmitting the data from the source actuator to the destination actuators using P2P routing and multicasting, respectively. We see that as the number of destination actuators increases, the number of forwarding operations increases. We also see that multicasting produces fewer forwarding operations than P2P routing. This is because in P2P routing, the source node generates one message for each destination actuator. Each message is independently forwarded to the destination actuator according to the P2P routing algorithm. In multicasting, fewer forwarding operations are generated in the system based on the minimum spanning tree. Also, only one message is forwarded along the common path towards different destinations. Fig. 13b further shows the comparison results of the energy consumed in the P2P routing

algorithm and the multicasting algorithm. We see that multicasting leads to less energy consumed than P2P routing because multicasting leads to fewer forwarding operations as shown in Fig. 13a.

Fig. 13c shows the comparison results of the total throughput of actuators in P2P routing and multicasting, respectively. The throughput is calculated as the average packet size per second received by all destination actuators. The figure shows that the throughput increases as the number of actuators increases because more actuators lead to more messages received by them. We also see that multicasting produces higher transmission throughput than the P2P routing. In the P2P routing, multiple messages are independently forwarded to the actuator destinations. The interference between the transmission of many messages reduces the throughput. Due to the same reason in Fig. 13a, multicasting produces fewer message forwarding operations, which produces less interference hence higher throughput.

5 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 it 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. To tackle the problem of high node failure rate, we further investigate the multi-path routing in a Kautz cell. We also studied an energy-efficient multicasting algorithm to further reduce energy consumption in communication between Kautz cells.

Extensive experimental results show the high performance of REFER compared with other WSAN systems and previous Kautz-based overlay, and the effectiveness of the multi-path routing and energy-efficient multicasting algorithms. In the future, we will investigate the performance of REFER in a sparse WSAN and using Kautz graph $K(d, k)$ with various d and k values.

Acknowledgments

This research was supported in part by U.S. NSF grants NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, Microsoft Research Faculty Fellowship 8300751, and the United States Department of Defense 238866. Haiying Shen is the corresponding author.

References

[1] L. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: Research challenges," *Ad Hoc Netw.*, vol. 2, no. 4, pp. 351–367, 2004.

[2] J. A. Sanchez, P. M. Ruiz, and I. stojmenovic, "Energy-efficient geographic multicast routing for sensor and actuator networks," *Comput. Commun., Elsevier*, vol. 30, no. 13, pp. 2519–2531, 2007.

[3] E. Ngai, M. R. Lyu, and J. Liu, "A real-time communication framework for wireless sensor-actuator networks," presented at the IEEE Aerospace Conf., Big Sky, MT, USA, 2006.

[4] G. A. Shah, M. Bozyigit, O. B. Akan, and B. Baykal, "Real-time coordination and routing in wireless sensor and actor networks," in *Proc. 6th Int. Conf. Next Generation Teletraffic Wired/Wireless Adv. Netw.*, 2006, pp. 365–383.

[5] W. Hu, N. Bulusu, and S. Jha, "A communication paradigm for hybrid sensor/actuator networks," in *Proc. 15th IEEE Int. Symp. Personal, Indoor Mobile Radio Commun.*, 2004, vol. 12, pp. 47–59.

[6] T. Melodia, D. Pompili, V. C. Gungor, and I. F. Akyidiz, "A distributed coordination framework for wireless sensor and actuator networks," in *Proc. 8th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2005, pp. 99–110.

[7] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *Wireless Netw.*, vol. 7, no. 6, pp. 609–616, 2001.

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

[9] E. P. Charles and P. Bhagwat, "Highly dynamic destination sequenced distance vector routing (DSDV) for mobile computers," *ACM Sigcomm Comput. Commun. Rev.*, vol. 24, pp. 234–244, 1994.

[10] M. Caesar, M. Castro, E. B. Nightingale, G. O. Shea, and A. Rowstron, "Virtual ring routing: Network routing inspired by DHTs," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2006, vol. 36, pp. 351–362.

[11] A. Awad, R. German, and F. Dressler, "P2P-based routing and data management using the virtual cord protocol," in *Proc. 13th Annu. ACM Int. Conf. Mobile Comput. Netw.*, 2006, pp. 443–444.

[12] H. Shen, Z. Li, and K. Chen, "A scalable and mobility-resilient data search system for large-scale mobile wireless networks," *IEEE Parallel Distrib. Syst.*, vol. 25, no. 5, pp. 1124–1134, Jul. 2013.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2001, pp. 161–172.

[14] 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. IEEE Conf. Comput. Commun.*, 2007, pp. 821–829.

[15] D. Guo, Y. Liu, and X. Ki, "Bake: A balanced kautz tree structure for peer-to-peer networks," presented at the 27th IEEE Conf. Comput. Commun., Phoenix, AZ, USA, 2008.

[16] D. Li, X. Lu, and J. Wu, "Fissione: A scalable constant degree and low congestion DHT scheme based on Kautz," in *Proc. IEEE Conf. Comput. Commun.*, 2005, pp. 1677–1688.

[17] K. Zuo, D. Hu, H. Wang, Q. Wu, and L. Su, "An efficient clustering scheme in mobile peer-to-peer networks," in *Proc. Int. Conf. Inf. Netw.*, 2008, pp. 1–5.

[18] W. K. Chiang and R. J. Chen, "Distributed fault-tolerant routing in kautz networks," *J. Parallel Distrib. Comput.*, vol. 20, pp. 99–106, 1994.

[19] Z. Li and H. Shen, "A kautz-based real-time and energy-efficient wireless sensor and actuator network," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2012, pp. 1–10.

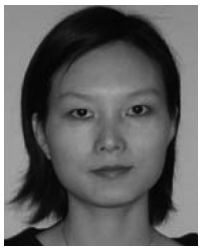
[20] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, vol. 353. New York, NY, USA: Springer, pp. 153–181.

[21] Wolf-Bastian Pöttner, H. Seidel, J. Brown, U. Roedig, and L. Wolf, "Constructing schedules for time-critical data delivery in wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 10, no. 3, pp. 44:1–44:31, May 2014.

[22] L. He, Z. Yang, J. Pan, L. Cai, J. Xu, and Y. Gu, "Evaluating service disciplines for on-demand mobile data collection in sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 4, pp. 797–810, Apr. 2014.

[23] S. Ji, R. Beyah, and Z. Cai, "Snapshot and continuous data collection in probabilistic wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 3, pp. 626–637, Mar. 2014.

[24] C.-C. Hsu, M.-S. Kuo, S.-C. Wang, and C.-F. Chou, "Joint design of asynchronous sleep-wake scheduling and opportunistic routing in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 63, no. 7, pp. 1840–1846, Jul. 2014.
[25] J. Ma, W. Lou, and X.-Y. Li, "Contiguous link scheduling for data aggregation in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1691–1701, Jul. 2014.
[26] A. Rezgui and M. Eltoweissy, "RACER: A reliable adaptive service-driven efficient routing protocol suite for sensor-actuator networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 5, pp. 607–622, May 2009.
[27] Q. Pan, J. Pan, X. Jin, and K. Shen, "Path planning for aviation actuators in deploying regular sensor networks," in *Proc. 9th Int. Conf. Mobile Ad-hoc Sensor Netw.*, 2013, pp. 161–166.
[28] J. Vazifehdan, R. V. Prasad, and I. Niemegeers, "Energy-efficient reliable routing considering residual energy in wireless Ad Hoc Networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 2, pp. 434–447, Feb. 2014.
[29] C. P. Ravikumar, T. Rai, and V. Verma, "Kautz graphs as attractive logical topologies in multihop lightwave networks," *Elsevier Sci.*, vol. 20, no. 14, pp. 1259–1270, 1997.
[30] M. Imase, T. Soneoka, and K. Okada, "Fault-tolerant processor interconnection networks," *Syst. Comput.*, vol. 17, no. 8, pp. 21–30, 1986.
[31] B. Han, "Zone-based virtual backbone formation in wireless ad hoc networks," *Ad Hoc Netw.*, vol. 7, no. 1, pp. 183–200, 2009.
[32] W. K. Chiang and R. J. Chen, "Distributed fault-tolerant routing in Kautz networks," in *Proc. 3rd Workshop Future Trends Distrib. Comput. Syst.*, 1992, pp. 297–303.
[33] J. L. Gross and J. Yellen, *Graph Theory and Its Applications*. Boca Raton, FL, USA: CRC Press, 2006.
[34] M. Miller, "Moore graphs and beyond: A survey of the degree/diameter problem," *Electron. J. Combinatorics*, vol. DS14, 61–63, 2005.
[35] M. Berg, O. Cheong, M. Kreveld, and M. Overmars, *Computational Geometry: Algorithm and Applications*. New York, NY, USA: Springer, 2008.
[36] 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. 29th Annu. ACM Symp. Theory Comput.*, 1997, pp. 654–663.
[37] R. Szewczyk, J. Polastre, A. Mainwaring, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Proc. 2nd ACM Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 214–226.
[38] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.
[39] The network simulator - ns-2 [Online]. Available: http://www.isi.edu/nsnam/ns/, 2015.
[40] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw*, vol. 11, no. 1, pp. 2–16, Feb. 2003.
[41] J. Wu, S. Yang, and M. Cardei, "On maintaining sensor-actor connectivity in wireless sensor and actor networks," presented at the IEEE Conf. Comput. Commun., Phoenix, AZ, USA, 2008.
[42] Linkquest, uwm1000 [Online]. Available: http://www.link-quest.com/, 2015.



Haiying Shen received the BS degree in computer science and engineering from Tongji University, China, in 2000, and the MS and PhD degrees in computer engineering from Wayne State University in 2004 and 2006, respectively. She is currently an associate professor in the Department of Electrical and Computer Engineering, Clemson University. Her research interests include distributed computer systems and computer networks, with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and grid and cloud computing. She was the program cochair for a number of international conferences and member of the program committees of many leading conferences. She is a Microsoft Faculty fellow of 2010 and a senior member of the IEEE and ACM.

Ze Li received the BS degree in electronics and information engineering from the Huazhong University of Science and Technology, China, in 2007. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering, Clemson University. His research interests include distributed networks, with an emphasis on peer-to-peer and content delivery networks, wireless multi-hop cellular networks, game theory, and data mining. He is a student member of the IEEE.