International Journal of Progressive Research in Science and Engineering
Volume-1, Issue-5, August-2020
www.ijprse.com

Design of A Self-Healing Mechanism for Wireless Sensor Networks

Sannidhi P[1], Sathwik R. Gutti[1], R Shamanth M[1], Sai Charan R[1], Manjunath kotari S[2]

[1]*Student, Department of Computer Science Engineering, Alva's Inst. of Engg. and Tech., Moodbidri, India.*

[2]*Assosiate Professor, Department of Computer Science Engineering, Alva's Inst. of Engg. and Tech., Moodbidri, India.*

Corresponding Author: sathwikrgutti@gmail.com

Abstract: - A sensor network consists of multiple detection stations called sensor nodes, each of which is small, lightweight and portable. Self-healing in Sensor Networks are increasingly becoming important. Especially in wireless sensor network, interference is anything which modifies, or disrupts a signal as it travels along a channel between a source and a receiver .Successful communication occurs in a wireless sensor network only in the absence of interference which is usually achieved by assigning non- interfering channels to the pairwise links (edges) that are necessary for good connectivity To overcome this, self-healing mechanism is used when the interference occurs, while communicating with each other. In this regard designing a self-healing routing mechanism for sensor networks, which restore connectivity after a node failure. This can be achieved by using the MATLAB tool for creating a base-level accessible, open-source, real- time ad-hoc routing scheme simulations, here we are targeting the ad-hoc on-demand distance vector (AODV) routing protocol.

Key Words: — ***Wireless sensor networks, security, self-healing.***

I. INTRODUCTION

The per-packet routing path serves as the meta-information for understanding detailed Wireless Sensor Networks (WSNs) behaviours in many network maintenance and diagnosis situations, e.g., routing dynamics [33], detections on wormholes [9] or packet loss holes [32], end-to-end packet transmission delay [29] or even perhaps per-packet transmission delay [13], network diagnosis [18] [26], etc. Reconstructing per-packet routing path information, however, has been known non-trivial. WSNs are self-organized and usually deployed in dynamic environments. The underlying network topology constantly changes and no fixed routing path can be expected for each node [30]. A straightforward solution to reveal the packet\ path is to record the complete path during packet forwarding, e.g., storing the ID sequence of all relay nodes, in each packet. The introduced overhead linearly grows with the routing path length, far scalable. The key insight of our design is as follows. The length of a routing path is usually much smaller than the network size. As a concrete example, the maximum path length reported in City See [22] is only 20 hops in comparison with its network size of 1200 nodes. Therefore, we can construct a path representation space, the number of whose dimension's equals to the total number of nodes in the network. In such a representation space, an arbitrary routing path can be represented by a path vector, where each element corresponds to a node in the network. The path vector sets the hop numbers for nodes on the path and zeros for those not involved in the path. As the path length is much smaller than the network size, such path vectors are thus sparse, i.e., the majority of elements are zeros. The path reconstruction becomes a problem of unveiling all existing path vectors hidden in the representation space. If all nonzero elements of a path vector can be encoded (with few bytes) into the packets forwarded along the path, we can recover the path vector (and thus the represented routing path) based on a small amount of packets using compressive sensing technique [5] [12].

In this paper, we propose a Compressive Sensing based Path Reconstruction method, CSPR, which formalizes the sparse path representation and leverages compressive sensing to recover per packet routing path. CSPR lets intermediate nodes briefly annotate the transmitted packets and classifies packets travelling along different paths into different groups. For a particular path, the forwarded packets encode independent observations and CSPR performs compressive sensing to recover the path when a certain amount of pack etc. (and the annotations) are collected at the sink. The path reconstruction by CSPR requires no inter-packet correlations and utilizes only a small number of received packets. CSPR is thus invulnerable to topology dynamics and lossy links. On the protocol level, CSPR introduces only small and fixed overhead in annotating each packet, which could be optimized accordingly for practical WSNs (e.g., 8 bytes per packet for a network with 245 nodes). In addition to the basic design, we further propose a set of optimization techniques to gradually shrink the representation space and heuristically scan possible paths for all unrecovered path vectors through the network topology learnt from already reconstructed routing paths. The numbers of packets needed for remaining path reconstructions are lowered and processing is thus accelerated. To examine the performance of CSPR, we first evaluate our method using a AODV testing method. The

experiment results validate the feasibility and applicability of CSPR in practice. We further conduct extensive and large-scale trace-driven simulations to examine the efficiency and scalability of CSPR. Compared to the state-of-the-art methods, CSPR achieves higher path recovery accuracy (i.e., 100% and 96% for experiments and simulations, respectively) with comparable overhead (8 extra bytes per packet).

II. THEORETICAL CONSIDERATION

In a WSN, all sensor nodes generate and relay packets to the sink along some routing paths [28]. At the sink, a path reconstruction method is desired to recover the routing path each packet traveled. One packet path is an ID sequence from the source of the packet to the sink, including IDs of all intermediate nodes relaying this packet and their hop numbers as well. There have been many efforts made to address the path reconstruction problem (as reviewed in Section 5). Two state-of-the-art methods, MNT [16] and Pathfinder [14], have been recently proposed. MNT [16] reconstructs per-packet path by exploiting inter packet correlation, i.e., a relayed packet and its adjacent packets locally generated at any node i are usually forwarded to the same next hop. Such local packets serve as anchors of the relayed packet at node i. As the first-hop receiver is recorded in packets, the path of a packet can be obtained by concatenating the first-hop receivers of all its anchors. Improving on MNT, Pathfinder [14] tolerates certain inconsistence in inter- packet correlation via explicitly recording inconsistence in packets. The reconstruction failure occurs once the inconsistence exceeds the tolerance capacity. To accurately locate anchors, Pathfinder further imposes the packet generation rate of each node to be identical and fixed. Both MNT and Pathfinder require stable network topology such that inter-packet correlation can be captured. The practical WSNs, however, behave dynamically and the wireless links are far from stable [17] Both network dynamic and packet loss have strong impacts on the anchor identifications, and thus deteriorate the performances of MNT and Pathfinder.

Ad-hoc On-demand Distance Vector(AODV) was developed almost 15 years ago, it still remains highly studied in the current state of the art. AODV is a reactive routing protocol that aims to reduces overhead by requiring few network-wide broadcasts. There is no centrally kept routing information and each node discovers routes only as necessary. Only in the event that a node does not know a path to its target destination, does it initiate a network- wide broadcast. This is known as flooding and involves a series of route request messages that propagate through the network, landing once on each connected node

until that node either is the destination, or knows a path to the destination. As flooding is the single most expensive action within AODV. A key feature of AODV is its elegant error handling. In this case, that a node cannot complete a link stored in its route table, a route error message must be sent. Whenever this happens during a data send transmission, the transmission must be halted and a route error message propagated all the way back to the source node. While this does eliminate the need for a network wide broadcast initially, this could still be a significant cause for additional overhead. After the propagation back to the source, flooding must occur again to determine new route.

A. Key point detection

AODV is an on-demand routing algorithm in that it determines a route to a destination only when a node wants to send a packet to that destination. It is a relative of the Bellman-Ford distant vector algorithm, but is adapted to work in a mobile environment. Routes are maintained as long as they are needed by the source. AODV is capable unicast and multicast routing. AODV enables dynamic, self-starting, multi-hop routing between mobile nodes wishing to establish and maintain an ad-hoc network. It allows for the construction of routes to specific destinations and does not require that nodes keep these routes when they are not in active communication. The AODV protocol is only use when two endpoints do not have a valid active route to each other.

In AODV, every node maintains a table, containing information about which neighbor to send the packets in order to reach the destination. Sequence numbers, which is one of the key features of AODV, ensures the freshness of routes. And in AODV,

networks are silent until connections are established. Network nodes that need connections broadcast a request for connection. The remaining AODV nodes forward the message and record the node that requested a connection. Thus, they create a series of temporary routes back to the requesting node. AODV determines a route to a destination only when a node wants to send a packet to that destination. Routes are maintained as long as they are needed by the source.

Sequence numbers ensure the freshness of routes and guarantee the loop-free routing. A new AODV-based routing protocol they call Dynamic Connectivity Factor routing Protocol (DCFP). This protocol also monitors local nodes and attempts to use additional local parameters to resolve route errors, thereby creating less overhead. One distinguishing feature of AODV is its use of a destination sequence number for each route entry. The destination

IJPRSE
Progressive Research
International Journal of Progressive Research in Science and Engineering
Volume-1, Issue-5, August-2020
www.ijprse.com

sequence number is created by the destination to be included along with any route information it sends to requesting nodes. Using destination sequence numbers ensures loop freedom and is simple to program. Given the choice between two routes to a destination, a requesting node is required to select the one with the greatest sequence number.

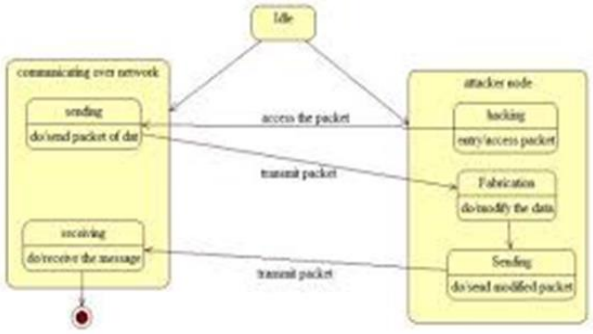


Figure 2. State machine diagram for Deceptive jamming attack

Fig.1. Call for send packet

B. Key point description

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, FOR- TRAN and Python. MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.

MATLAB is widely used in all areas of applied mathematics, in education and research at universities, and in the industry. MATLAB stands for MATRIX Laboratory and the software is built up around vectors and matrices. This makes the software particularly useful for linear algebra but MATLAB is also a great tool for solving algebraic and differential equations and for numerical integration. MATLAB has powerful graphic tools and can produce nice pictures in both 2D and 3D. It is also a programming language, and is one of the easiest programming languages for writing mathematical programs. MATLAB also has some tool boxes useful for signal processing, image processing, optimization, etc.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems. The MATLAB application is built around the MATLAB scripting language. Common usage of the MATLAB application involves using the Command Window as an interactive mathematical shell or executing text files containing MATLAB code. Like other scripting languages MATLAB also provides facilities for variables, vectors, matrices.

III. Proposed Algorithm

The main objective of the Self-healing mechanism in wireless sensor network is to detect the error and automatically en-routes the another shortest path except the earlier path. Ad-hoc On-demand Distance Vector(AODV) Routing Protocol is used to find the shortest path based on the congested or dead node then it en-route the path and calculate the shortest path from source to destination.

A *Source* will be the route node for all the nodes and it creates path to each and every node.

A *Destination* the destination node is considered as the final node to find the shortest distance between the starting node to the ending node.

The *Path-Connection* The nodes are connected between the adjacent nodes and the source node is considered as the root node and creates path to all other nodes then finds the shortest path to the destination node.

A. Key point detection

*Call to sendPacket***:** The sendPacket function is called with the parameter of source node and destination. By checking the arguments bring global into scope. Initialize persistent variable and finally get the sequence number. Then, initialize the table and add the start node for the function. If the node has a route entry for the destination, and try to send normally. Ignore and routes marked by request error. Then make a time to lightup paths iteratively. Finally remove the route with expired lifetime.

*Initialize table of Hops***:** A table data structure stores all the messages sent throughout the network as part of this interaction. It contains node, nextNode, messageType, depth, where depth is the iteration that it will be highlighted on when being displayed on the user interface.

*Source has route entry for destination***:** If the source node has a valid route to the destination, a call to send () is attempted. If the call to send fails (i.e. encounters a route error), then

International Journal of Progressive Research in Science and Engineering
Volume-1, Issue-5, August-2020
www.ijprse.com

flood () is called instead. Both subroutines write their changes to the table of hops.

Call to send (): The call to send () is relatively straightforward. Here sends a packet normally assuming the route table has an entry for it. Then, checks for existing route entry packets. Look in current nodes route table for the next hop node otherwise exit if no node was found. If expecting to have a valid path by request error, update lifetime field for the route. Then update the table otherwise exit when it reached the destination and set success to

true.

Call to Flood(): Flood may be called repeatedly, if necessary. Once the computation is complete, the table of hops is parsed and the routes are iteratively highlighted on the user interface to simulate data flow. Performs network flooding for route discovery and walk down table rows and add connected nodes breadth-first then get connected nodes. Remove duplicates when finished at the depth. Then, find the distance between Node and from for all occurrences of duplicated node. Remove occurrences but the one with the minimum distance. If the node happens to have a valid entry on the route table, go ahead and send normally. Add each of this nodes connected nodes unless its already on the table before depth.

Congestions in the node: The WSN is a built of" nodes"- from a few to several hundred are even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting.

Failure of nodes in the Sensor networks:

Self-healing discover, diagnose, and react to network disruptions. Self-healing components can detect system malfunctions or failures and start corrective actions based on defined policies to recover the network for a node. The automatic recovery from damages improves the service availability.

IV. Experiment And Discussion

The results obtained are as discussed below

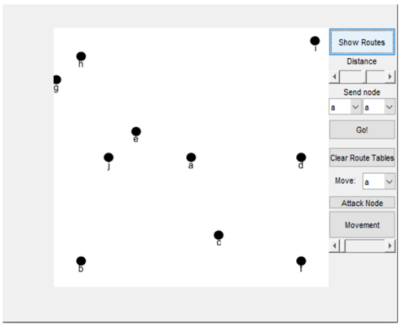


Fig.2. Creation of nodes

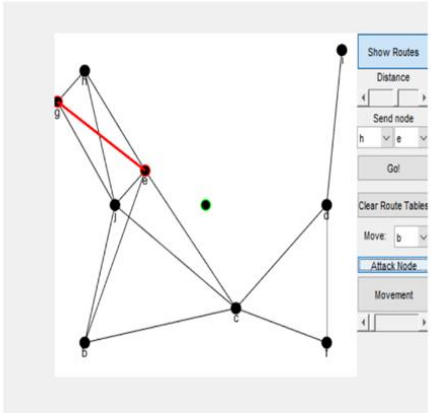


Fig.3. Error path detection

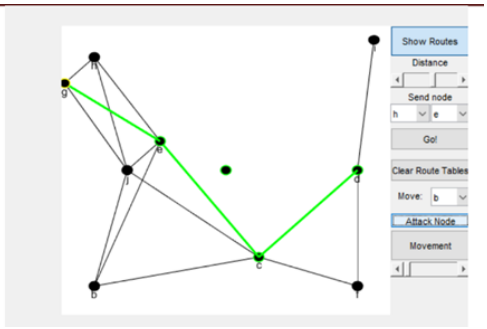


Fig.4. Self-healed shortest path

After receiving the acknowledgement from the destination node, then the self- healed node is discarded while establishing a path. Thus, the shortest path is created from source to destination.

International Journal of Progressive Research in Science and Engineering
Volume-1, Issue-5, August-2020
www.ijprse.com

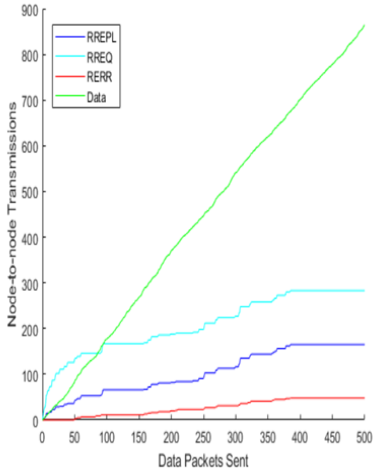
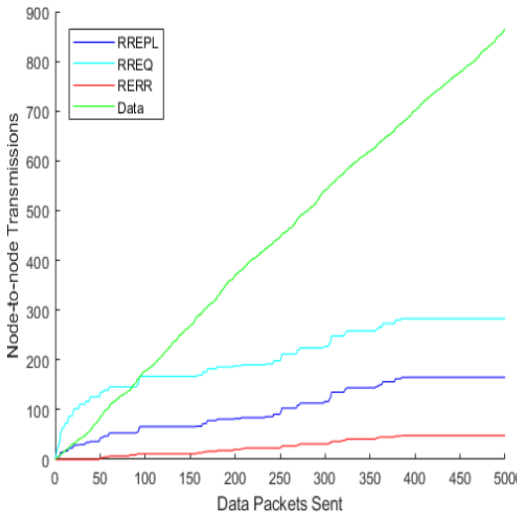


Fig.5. Network overhead with multiple source and destination

This represents an initial flooding where there are 2 source sends the data packets to different destination hops. Therefore, it is inconsistent from single source to multiple. The subsequent routing table is shown. The sequence number have increased several times. Each time the nodes notice a change in their local network topology the increase their sequence numbers. All nodes connected or disconnected from A have been reminted to 2.



The frequent movement in figure 13 causes the route request rate to finally surpass the data rate and would likely be enough to overwhelm the controllers on most realistic nodes. In such a high mobility case, a specialized routing algorithm would likely need to be utilized as AODVs performance is lacking here.

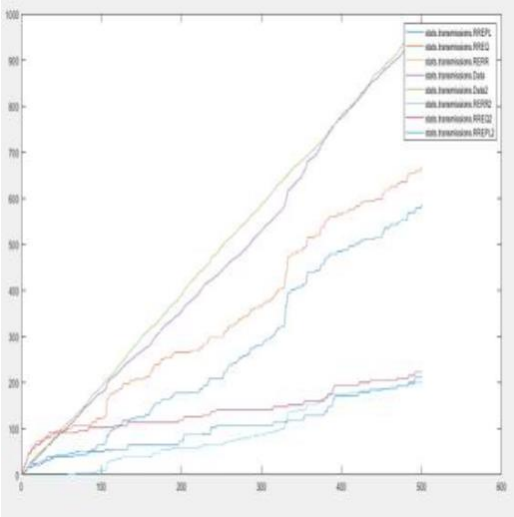


Fig.5. Network overhead

This represents an initial flooding where there are 2 source sends the data packets to different destination hops. Therefore, it is inconsistent from single source to multiple.

V. Conclusion And Future Scope

The main aim of this project implementation is to overcome the congestion which occurs in between the nodes. For this, the project creates a wireless sensor node to find the shortest path from the source to destination. The source node will start to calculate the shortest path by visiting each and every node. If any of the nodes fails due to some reason congestion occurs. In this situation, the project self-heals the network by recalculating and selecting new shortest path. This can be achieved by using one of the routing protocol called Ad-hoc On-demand Distance Vector(AODV) routing protocol. Then the MATLAB-based ad-hoc on-demand distance vector (AODV) simulation presented here provides a meaningful method of demonstrating basic routing concepts quickly and easily. The MATLAB environment provides for easy inspection and expansion into additional analysis with the provided functionality

In this regard designing a self-healing routing mechanism for sensor networks, which restore connectivity after a node failure. This can be achieved by using the MATLAB tool for creating a base-level accessible, open-source, real- time ad-hoc routing scheme simulations, here we are targeting the ad-hoc on-demand distance vector (AODV) routing protocol.

REFERENCES

[1]. Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal, Wireless sensor network survey, Department of Computer Science, University of California, Davis, CA 95616, United States.

[2]. Atsuko Miyaji and Kazumasa Omote, Self-healing wireless sensor networks, School of Information Science, JAIST, Nomi, Japan.

[3]. StuartMiller, Rea ltime AODV Simulation in MATLAB, Department of Electrical and Computer Engineering Missouri University of Science Technology.

[4]. Markus Lanthaler, Self-Healing Wireless Sensor Networks, Department of Computer Science, University of Helsinki.

[5]. Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir Das, Ad-Hoc On Demand Distance Vector (AODV) Routing, Proceedings of IEEE WMCSA'99, New Orleans, LA, February 1999.

[6]. Sumathi.S and Chandrasekaran M, Self-Healing Wireless Sensor Network, Anna University, Coimbatore India.