International Journal of Modern Science and Technology, 2021;6(8):134-140.
ISSN: 2456-0235.
www.ijmst.co

Research Article

Energy and load balancing scheme based path tracing for WSN networks

U. Mohideen Abdul Kader*, S. Sumithra

Department of Electronics and Communication Engineering,
JJ College of Engineering and Technology, Trichy.
Tamilnadu, India.

*Corresponding author's e-mail: umar.kader@gmail.com

Abstract

Wireless sensor networks (WSN) groups specialized transducers that provide sensing services to Internet of Things (IoT) devices with limited energy and storage resources. Since replacement or recharging of batteries in sensor nodes is nearly impossible, power consumption becomes one among the crucial design issues in WSN. Most sensor networks employ dynamic routing protocols in order that the routing topology are often dynamically optimized with environmental changes. The routing behaviours are often quite complex with increasing network scale and environmental dynamics. Knowledge on the routing path of each packet is certainly a great help in understanding the complex routing behaviours, allowing effective performance diagnosis and efficient network management. We propose PAT, a universal SensorNet path tracing approach. PAT includes an intelligent path encoding scheme that allows efficient decoding at the base station side. To make PAT more scalable, we propose techniques to accurately estimate the degree information by exploiting timing information, allowing more compact path encoding. Moreover, we employ subpath concatenation to infer excessively long paths with a high recovery probability. We propose an analytical model to quantify the advantages of PAT with varying network scale, network density, routing dynamics and packet delivery performance. Simulation analysis shows the modified version performs better than the existing protocol by enhancing the throughput, end-to-end delay, and residual energy.

Keywords: Wireless sensor networks; Internet of things; Recharging of batteries; SensorNet.

Introduction

As an emerging technology that bridges cyber systems and the physical world, wireless sensor networks (WSNs) are envisioned to support numerous applications such as military surveillance, environmental monitoring, infrastructure protection, etc. In these networks, large numbers of tiny, low-power wireless sensing devices are self-organized, collecting the sensing data to a central sink in a multihop manner. Most sensor networks employ dynamic routing protocols in order that the routing topologies are often dynamically optimized with environmental changes [1].

The TinyOS protocol is an instance of dynamic routing protocol with which each node regularly estimates the expected number of transmissions (ETX) to the sink and dynamically selects the next-hop forwarder with a minimum ETX along the path. Due to the dynamic routing scheme, a node could forward different packets to different next hops, although these packets have the same destination, i.e., the sink node [2]. In recent years an efficient design of a Wireless Sensor Network has become a number one area of research. A Sensor is a device that responds and detects some type of input from both the physical or environmental conditions, such as pressure, heat, light, etc. The output of the sensor is usually an electrical signal that's transmitted for further processing [3].

The main contribution of this phase I project is a SensorNet path tracing approach that satisfies both the above requirements. We present a novel path tracing approach called PAT (Path Tracing) which includes several versions with increasing recovery capability.

Received: 28.05.2021; Received after Revision: 02.08.2021; Accepted: 05.08.2021; Published: 20.08.2021.

The basic version (bPAT) attaches to each data packet a field called pathvalue which is updated hop-by-hop and encodes path information towards the sink. When a packet is received at the sink, the attached can be efficiently decoded to recover the routing path [4].

The path value, however, can overflow (e.g., when the routing path is excessively long), impairing PAT's effectiveness. This problem is addressed from two perspectives. The dynamic version (tPAT) accurately estimates each node's in-degree (which is time-varying) by exploiting timing information, allowing more compact path encoding. The extended versions (xbPAT and xtPAT) try to infer a long path by concatenating two short subpaths. The key insight of PAT for inferring such a path (e.g., A to S) is to let an intermediate node on the long path (e.g., B) generates an area packet towards the sink. It is highly probable that the 2 local packets will follow an equivalent path towards the sink since the routing topology keeps stable during a brief period in most cases. PAT can thus obtain the long path by concatenating two short subpaths (i.e., A to B and B to S) [5].

Proposed system

We propose an analytical model to quantify the benefits of PAT with varying network scale, network density, routing dynamics, and packet delivery ratio. We carefully evaluate PAT's performance using simulation scenario experiments, trace-driven study, and extensive simulations. Results show that PAT significantly outperforms existing approaches.

The contributions of this article are summarized as follows:

- We propose a novel path tracking approach which simultaneously satisfies both the universal and scalable requirements.
- We propose an analytical model to gain insights into different path tracking approaches with varying network scale, network density, routing dynamics, and packet delivery ratio.
- We evaluate our approach using traces of a large-scale urban sensing network as well as simulation experiments and large-scale simulations with different network configurations.
- Results show that our approach significantly outperforms existing approaches.

System design

The proposed sensor network where some nodes generate local packets to a central sink. We would like to trace the routing paths of those data packets with a small and bounded message overhead. PAT contains a series of versions with increasing recovery capability at the cost of slightly increasing message overhead. We will describe different versions in the following subsections. The Fig. 1 shows the path value is initialized to 0 at the original node. It is then updated using multiplication and addition according to the rule associated with each in-edge.

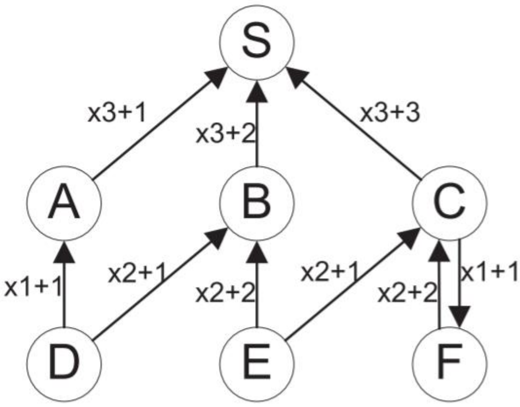

Fig. 1. Proposed system design-basic PAT routing

bPAT: the basic version F

Figure shows the basic idea of PAT. With PAT, data packets are instrumented with a field called pathvalue which is updated at each forwarder along the routing path towards the sink. Each forwarder updates the pathvalue according to its in-edges. If there are n in-edges and packets are coming from the i-th edge ($1 \le i \le n$), the pathvalue is updated to: path value × n + i. In Fig. 1, we annotate each edge with an update rule "×n + i".

For example, we consider routing paths from node D. The pathvalue is initialized as 0 at D. If the routing path is DAS, the pathvalue is updated as 1 at A, and is updated as 4 at the sink S. If the routing path is DBS, the path value is updated as 1 at B, and is updated as 5 at S.

The update rules can not only work for a cyclic paths as illustrated above, but also work for cyclic paths. Consider the routing paths from node F. If the routing path is FCS, the path value is updated as 2 at C, and is updated as 9 at S. If the routing path is FCFCS, the path value is updated subsequently as 2 (at C), 3 (at F), 8 (at C), and finally 27 at S. We see that the different path values can differentiate different paths in this example.

The proposed algorithm 1 shows the decoding algorithm. We will use an example to show the working details of the algorithm. Suppose at the sink node, we get a path value of 27. The path P is initialized to (S).

- k=S. We would like to determine which node forwards the packet to the current node k=S. The index is calculated as (27-1) % 3 + 1 = 3 which corresponds to node C. Therefore, k=C, P=(S,C), and path value= (27-3)/3=8.
- k=C. Again we would like to determine C’s previous hop. The index is calculated as (8−1) % 2 + 1 = 2 which corresponds to node F. Therefore, k=F, P=(S,C,F), and pathvalue=(8−2)/2=3.
- k=F. The index is (3−1) % 1 + 1 = 1 which corresponds to node C. Therefore, k=C, P=(S,C,F,C), and pathvalue=(3−1)/1=2.
- k=C. The index is (2−1) % 2 + 1 = 2 which corresponds to node F. Therefore, k=F, P=(S,C,F,C,F), and pathvalue=(2−2)/2=0.
- The while loop terminates since pathvalue is no longer larger than 0, and the decoded path is obtained by reversing P. The resulting path is hence FCFCS.

The encoding and decoding scheme of PAT has some nice properties:

1. The update rules can be locally determined by a node.
2. A pathvalue can uniquely determine a routing path. In contrast to PathZip which uses exhaustive search in the entire space of possible routing paths,

PAT has an efficient decoding algorithm (constant overhead per hop), enabling fast packet path tracing.

A network usually employs broadcasting for routing initialization and maintenance. Each node k can find all its direct neighbors that k can hear. The i-th found neighbor is assigned with index i. Each node can locally obtain the above information and transfer the information to the central sink after the network initialization phase. There are chances that the message drops along the path towards the sink. bPAT uses the most significant bit in path value to record whether the path value overflows. bPAT cannot recover path of a data packet with an overflown pathvalue. In the initialization phase, path values are intentionally marked as overflown.

tPAT

In bPAT, each node k estimates n(k) as the number of all its direct neighbors that k can hear. It is not accurate since it considers all direct neighbors that k can hear, some of which never forward packets to k. For example, in Fig. 1, node B’s direct neighbors include all other nodes. However, only nodes D and E forward packets to B. If B takes into account in all neighbors, the number of in-edges of B will be 6 while the actual number of in-edges is 2. Overestimating the in-degree of a node will waste the encoding space, causing the path value to overflow more easily.

Instead of relying on control-plane broadcasting packets to find all incoming neighbors, tPAT exploits data-plane packets: each time a node receives a data packet from a new direct neighbor, it adds the new neighbor into its neighbor table. In-degree estimation based on data-plane packets can avoid the overestimation issue, resulting in more efficient use of the encoding space.

xbPAT and xtPAT

So far, we have only considered the case when the pathvalue never overflows. Both bPAT and tPAT cannot decode overflown pathvalue. Indeed, for long path with high in-degree forwarders, it is possible that the pathvalue overflows, causing decoding failures at the sink. We can extend the pathvalue field to accommodate complete information about the path. However, the pathvalue field might be very long for some packets. It is favourable to devise an approach with small and bounded overhead, and it can also recover very long path with a high probability.

The basic idea of our approach is to let an intermediate node on the path, e.g., B, generate a local packet. If path from A to B’s previous hop can be recovered and path B to S can be recovered, path A to S can also be recovered by concatenating the two paths if packets from A and B follow the same path to the sink.

Note that we do not require that path B to S is directly recovered via path value; it can also be recovered using another intermediate node. Hence the above recovering process can be performed in an iterative manner, resulting in excellent scalability.

The Inferring Path with Overflown Algorithm shows the algorithm for recovering a pathvalue which will overflow with subpath concatenation. We will use the example shown in this section to show its working details.

In Fig. 2, A generates a data packet. The path value in A's packet will be updated hop-by-hop towards the sink. An intermediate node on the path, B, detects that the path value will overflow if it is further updated at B. B keeps the path value unchanged, and at the same time, attaches two fields to A's packet: the anchor field records B's previous hop while the crc field will be the CRC value of the subpath starting from B to S. With the anchor field and the non-overflown path value field, PAT can recover the subpath from A to B's previous hop. B will also generate a local packet towards the sink. It is assumed that if the transmission times of the two packets (from A and from B) are close, B's packet will follow the same path as A's packet with a high probability. The path from B to S can be directly recovered since B's packet does not experience overflow in pathvalue. PAT verifies if the recorded crc in A's packet equals to the calculated CRC values of the known path from B to S. If it is the same, PAT considers the path from A to S is the concatenation of the path from A to anchor (i.e., B's previous hop) and the path from B to S.

Fig. 3 shows a more complicated case in which the path from B to S cannot be directly recovered. In this case, the local packet from B will trigger another helper node, C, to generate a local packet. The path from B to S can be recovered in a process similar to the one we have described above. If the path from B to S is recovered, the path from A to S can be recovered if the recorded crc in A' packet equals to the calculated CRC value of the known path from B to S.

The proposed section summaries the PAT's basic version, bPAT, requires the smallest message overhead and the recovery capability is relatively low. The dynamic version, tPAT, increases the message overhead. The recovery capability is also enhanced since the encoding space is more efficiently used. The full-fledged version, xtPAT, requires the largest message overhead and recovery capability is high.

Fig. 2. Illustrates the case in which B to S can be directly decoded

Fig. 3. Illustrates the case in which the recovery of B to S depends on another intermediate node, C.

Experimental implementation

In this module, a wireless sensor network is created. All the nodes are configured and randomly deployed in the network area. Since our network is a wireless sensor network, nodes are assigned with initial energy, transmitting energy and receiving energy. A routing protocol is implemented in the network. Sender and receiver nodes are randomly selected and the communication is initiated.

Initialize the wireless network nodes with multihop network by randomly deploying 30 nodes in an area of 1500 X 1500. Nodes are assigned with initial energy, transmitting energy and receiving energy. A routing protocol is implemented in the network. Sender and receiver nodes are randomly selected, and the communication is initiated.

Network deployment

All the nodes are configured to exchange the location and initial energy information among all the nodes. All the nodes are configured to exchange the location and initial energy information among all the nodes. A routing protocol is implemented in the network. Base Station is configured with highest communication range. Data Transmission is established between nodes using UDP agent and CBR traffic.

Implementation of proposed scheme

In this module, to enable all the nodes to get the global energy model, we propose a proposed algorithm, that under this general dynamic battery model, there exists an optimal policy consisting of time-invariant routing probabilities in a fixed topology network and these can be obtained by solving a set of problems.

Performance analysis

In this module, the performance of the proposed network coding method is analyzed. Based on the analyzed results X-graphs are plotted. Throughput, delay, energy consumption are the basic parameters considered here and X-graphs are plotted for these parameters.Finally, the results obtained from this module is compared with previous results and comparison X-graphs are plotted. Form the comparison result, final RESULT is concluded.

Performance metrics

Throughput

It is the ratio of the total number of bits transmitted (Btx) to the time required for this transmission, i.e. the difference of data transmission end time and start time (tstart). This metric depicts how the congestion control mechanism at the source node is affected by the packet losses caused by JF-nodes. A decrease in throughput is an outcome of any JF attack.

$$\text{Throughput} = (B_{tx})/(t_{end} - t_{start})\ \text{bps}$$

Packet Delivery Ratio

This is defined as the ratio of the number of packets received at the destination and the number of packets sent by the source. Here, $pktd_i$ is the number of packets received by the destination node in the ith application, and $pkts_i$is the number of packets sent by the source node in the ith application.

Average End-to-End Delay

It is average transmission delay of packets transmitted from source to destination. D is computed as the ratio of the sum of individual delay of each received data packet to the total number of data packets received. This metric is used to evaluate impact of a JF-attack on delay-sensitive applications of TCP-based MANETs. By intentionally discarding, delaying or reordering packets, a JF-node can increase the value of this metric; increase being caused by re-transmissions of such packets due to timeout at TCP source.

D= number of received packed/total time

Results and discussion

We would like to analyze the performance of different approaches with different affecting factors, including network scale (hop count), network density (node degree), routing dynamics, and packet delivery ratio. We propose a theoretical model to quantify the performance of various approaches with varying network parameters accurately. With this analytical model, we can analyze the performance of different approaches before deployment and make the appropriate decision for a specific network. Table 1 shows the simulation parameter settings.

Table 1. Simulation parameters

Parameters	Values
Simulation Tool	NS2
No. of Nodes	40
Area	1400 X 1400
Routing Protocol	AODV (PAT)
Clusters	4
Traffic	CBR
Transport Layer	UDP
Channel Type	Wireless Channel
MAC Type	IEEE 802.11
Antenna Type	Omni Directional Antenna
Queue Type	DropTail-PriQueue
Queue Length	1000
Initial Energy	100J
Transmitter Energy	0.6W
Receiver Energy	0.3W
Simulation START/STOP Time	0.0/5.0 s

We investigate the recovery probability of different approaches using our analytical model. We are mainly interested in how four different factors affecting the performance, including network scale (hop count), network density (degree), routing dynamics, and link PRR. We consider the recovery of a path from node k. The packet hop count is h. All forwarders along the path have degree of D and in-degree of Din = D/2.

The simulation shows the performance of different approaches with varying network parameters in networks with different scales. The real network experiments evaluate the message overhead and the performance of different approaches in real-deployed networks which show their performance in real network conditions. Fig. 4 shows the WSN network deployment used in this study.

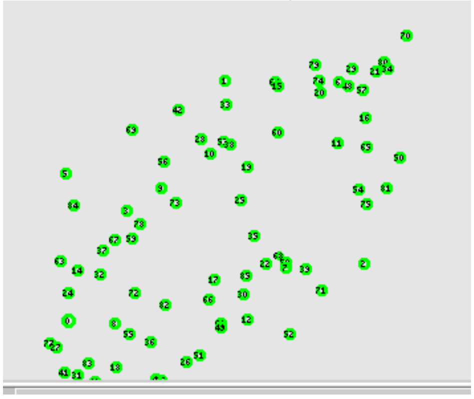

Fig. 4 WSN Network Deployment

Fig. 5-7 shows the throughput of different approaches varied with hop count. The performance of existing system degrades with increasing node count because the path probability is the product of the percentages of detected stable period at forwarders along the path.

This experiment considers the link failure issue for a single data flow, and the influence of other background traffic is not considered here. As shown in fig. 5, the proposed system switches the path quickly when a mobility event occurs at 6s.

Fig. 5. Time vs. No. of Packets

To further investigate performance of the proposed prediction model, we consider multiple data flows with different data rates and mobility. The Fig. 6 shows the utilization of energy in the proposed topology. The performance of proposed scheme with traditional distributed network control.

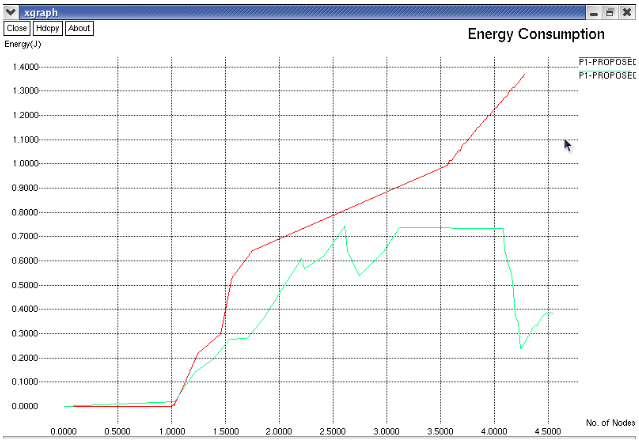

Fig. 6. No. of Nodes vs. Energy

To further investigate performance of the proposed two layer link failure prediction model, we consider multiple data flows with different data rates and mobility. In this project, we show the performance of proposed scheme with traditional distributed network control. The throughput and delay are observed for different data flows with different traffic rates (from 2K bps to 10K bps for each flow).

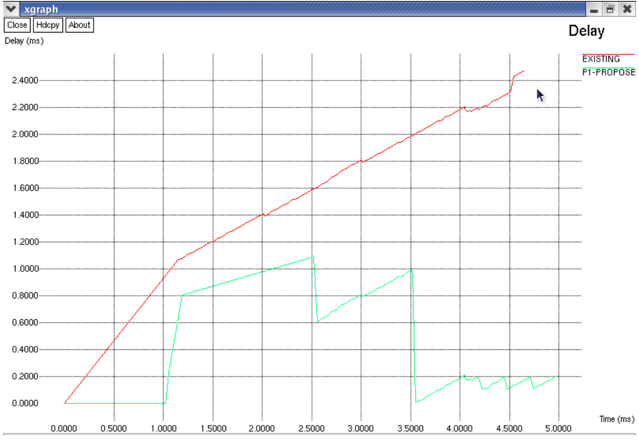

Fig. 7. Time vs. Delay

Conclusion

Since energy and lifetime are two major constraints in designing any routing protocol for WSN, much research has been done to achieve the goal. In this work, we proposed the PAT, a universal sensornet path tracing approach. PAT includes an intelligent path encoding scheme that allows efficient decoding at the BS side. To make PAT more scalable, we propose techniques to accurately estimate the degree information by exploiting timing information, allowing more compact path encoding. Moreover, we employ subpath concatenation to infer excessively long paths with a high recovery probability. We propose an analytical model to quantify the benefits of PAT with varying network scale, network density, routing dynamics and packet delivery performance. Simulation result shows improved network performance for metrics such as residual energy, delay, throughput and lifetime. In future, the work will extended to minimize the path tracking complication and improve the network performance via new protocol design also we implement the model hardware for the output scenario related to the universal path tracking based on WSN model.

Conflict of interest

I declare no conflict of interest of this work.

References

[1] Zhang H, Shen H. Balancing Energy Consumption To Maximize Network Lifetime In Data-Gathering Sensor Networks. IEEE Trans. Parallel and Distributed Systems 2009;20(10):1526-39.

[2] AlShawi IS, Yan L, Pan W, Luo B. Lifetime Enhancement In Wireless Sensor Networks Using Fuzzy Approach And A-Star Algorithm. IEEE Sensors J 2012;12:3010-3018.

[3] Cassandras C, Wang T, Pourazarm S. Optimal Routing And Energy Allocation For Lifetime Maximization Of Wireless Sensor Networks With Nonideal Batteries. IEEE Trans. Control of Network Systems 2014;1(1):86-98.

[4] Habibi, J, Aghdam, AG, Ghrayeb A. A Framework For Evaluating The Best Achievable Performance By Distributed Lifetime-Efficient Routing Schemes In Wireless Sensor Networks. IEEE Transactions on Wireless Communications 2015;14:3231-46.

[5] Cheng P, Qi Y, Xin K, Chen J, Xie L. Energy-Efficient Data Forwarding For State Estimation In Multi-Hop Wireless Sensor Networks. IEEE Trans. Autom. Control 2016;61:1322-27.

[6] Kurt S, Yildiz HU, Yigit M, Tavli B, Gungor VC. Packet Size Optimization In Wireless Sensor Networks For Smart Grid Applications, IEEE Trans Ind Electron. 2017;64:2392-1.

[7] Liu T, Gu T, Jin N, Zhu Y. A Mixed Transmission Strategy To Achieve Energy Balancing In Wireless Sensor Networks,” IEEE Trans. Wireless Commun. 2017;16:2111-22.

[8] Nayak P, Vathasavai B. Energy Efficient Clustering Algorithm For Multi-Hop Wireless Sensor Network Using Type-2 Fuzzy Logic. IEEE Sensors J 2017;17:4492-99.

[9] Pandey OJ, Mahajan A, Hegde RM. Joint Localization and Data Gathering Over Small World WsnWith Optimal Data Mule Allocation. IEEE Transactions on Vehicular Technology 2018;67:6518-32.

[10] Qianao D, Rongbo Z, Hao L, Maode M. An Overview of Machine Learning-Based Energy-Efficient Routing Algorithms in Wireless Sensor Networks. Electronics 2021;10(13):1539.

****