International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

Title: **DESIGN ANALYSIS OF PATH INTERFERENCE IN WIRELESS SENSOR NETWORKS**

Volume 08, Issue 05, Pages: 280–287.

Paper Authors

P. KISHORE, P. CHAKRADHAR, DR. V. SURYA NARAYANA
Ramachandra college of Engineering, Eluru, Affiliated to JNTU Kakinada.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

DESIGN ANALYSIS OF PATH INTERFERENCE IN WIRELESS SENSOR NETWORKS

[1]P. KISHORE, [2]P. CHAKRADHAR, [3]DR. V. SURYA NARAYANA
[1]Department of Computer Science and Engineering, Ramachandra college of Engineering, Eluru, Affiliated to JNTU Kakinada.
[2]Associate Professor, Department of Computer Science and Engineering, Ramachandra College of Engineering, Eluru.
[3]Head of the Department in Computer Science and Engineering, Rama Chandra College of Engineering, Eluru.
kishorepalle123@gmail.com, chakradhar1379@gmail.com ,s_vadhri@yahoo.co.in

ABSTRACT:

Recent wireless sensor networks (WSNs) are becoming increasingly complex with the growing network scale and the dynamic nature of wireless communications. Many measurement and diagnostic approaches depend on per-packet routing paths for accurate and fine-grained analysis of the complex network behaviors. In this paper, we propose iPath, a novel path inference approach to reconstructing the per-packet routing paths in dynamic and large-scale networks. The basic idea of iPath is to exploit high path similarity to iteratively infer long paths from short ones. iPath starts with an initial known set of paths and performs path inference iteratively. iPath includes a novel design of a lightweight hash function for verification of the inferred paths. In order to further improve the inference capability as well as the execution efficiency, iPath includes a fast bootstrapping algorithm to reconstruct the initial set of paths. We also implement iPath and evaluate its performance using traces from large-scale WSN deployments as well as extensive simulations. Results show that iPath achieves much higher reconstruction ratios under different network settings compared to other state-of-the-art approaches.

INTRODUCTION:

This survey provides an overview of wireless sensor network (WSN) connectivity, and discusses existing work that focuses on the connectivity issues in WSNs. In particular, we are interested in maintaining connected WSNs and their connectivity related characteristics including sensor node placement, as well as the construction of a small connected relay set in WSNs. We aim to review extensively the existing results related to these topics, and stimulate new research. Sensor networks have a long history, which can be traced back as far as the 1950's. It is recognized that the first obvious sensor network was the Sound Surveillance System (SOSUS) [1, 2]. The SOSUS was made up of an array of acoustic sensors that were interconnected by wired cables and were deployed by the US in deep ocean basins during the Cold War to detect and track Soviet submarines. In its early stages, the development of sensor networks was mainly driven by military use, in which sensor nodes were wired together to provide battlefield surveillance. Evolution of technologies has driven sensor networks away from their original appearance. With the emergence of integrated sensors embedded with wireless capability, most of current sensor networks consist of a collection of wirelessly interconnected sensors, each of which is embedded with sensing, computing and communication components. These sensors can observe and respond to phenomena in the physical environment [3]. Such sensor networks are referred to as wireless sensor networks (WSNs). These WSNs provide flexibility in deployment and maintenance, exploit the ability of wireless networks to be deployed in highly dynamic environments and hence enable sensor networks to be potentially used in a wide range of civilian and military applications, including security surveillance (e.g., to alert of terrorist threats), environmental monitoring, habitat monitoring, hazard and disaster monitoring and relief operations, health field applications, and home applications (e.g., smart environments) [3]. The wireless communication in WSNs can be either ad hoc (multi-hop) or single-hop wireless transmission [4]. Though the latter is popular in short-range applications, such as smart homes, the former, ad hoc technique, attracts more interests due to its high flexibility and ability to support long-range, large scale, and highly distributed applications. In this survey, we only focus on wireless sensor networks adopting multi-hop transmission. In a WSN, after collecting information from the environment, sensors need to transmit aggregated data to gateways or information collection nodes. It is important to ensure that every sensor can communicate with the gateways. Due to the multi-hop communication of WSNs, a sufficient condition for reliable information transmission is full connectivity of the network. A network is said to be fully connected if every pair of nodes can communicate with each other, either directly or via intermediate relay nodes. Due to the large number of sensors in a WSN, the total cost could be high for the whole network, though the cost of each individual sensor is low. Therefore, it is important to find the minimum number of nodes required for a WSN to achieve connectivity. Another related important problem for WSNs is finding a small connected relay set to assist in routing. Multi-hop WSNs need to perform efficient routing. Since mobile ad hoc networks (MANETs) and WSNs often have very limited, or even does not have, fixed infrastructure, the routing process in such networks is often complicated and inefficient; it can generate a large amount of overhead, and there are many possible paths, due to the broadcast nature of the wireless communications. Thus it is helpful to find a small connected set of sensor nodes to form a routing “backbone”, and restricted all other nodes to connecting to this backbone by a single hop. This node set can also help to resolve the broadcast storm problem [5], which is often caused by blind flooding. As

WSNs may be deployed in inaccessible terrains, and may contain a tremendous number of sensor nodes, it is often difficult or impossible to replace or recharge their batteries. Thus, energy conservation is critical for WSNs, both for each sensor node and the entire network level operations. Various approaches have been proposed to reduce energy consumption for sensor networks. For example, for the network level operations such as routing, if only a small fraction of sensors are involved in the routing process, the rest of the sensors can be turned off to save energy. This scheme is supported by the hardware and software advances that leverage the capability of temporarily shutting down those sensors that are not involved in any network operations. For instance, Rockwell's WINS sensor nodes can achieve a factor of ten power reduction by shutting down the radio transceiver, compared to those idle nodes whose transceivers are on [6]. However, a prerequisite for this type of energy saving scheme is that the WSNs still perform all the required functions even with some nodes turned off. This raises an important research problem: what is the maximum number of sensors that can be turned off, while maintaining functionality of the WSN? This problem is equivalent to minimizing the total number of active nodes, subject to ordinary operations of the system. The selected sensors will function as backbone relay nodes to maintain communications within the entire sensor network. A further important problem, which is beyond the scope of this survey, is how to optimally shut off and turn on sensors over time to maximise network lifetime [7]. The remainder of this paper is organized as follows. Section 2 gives a brief introduction to the graph models applied to wireless network investigations. Section 3 provides an overview of the prior results for connectivity studies in wireless ad hoc networks and WSNs, including percolation theory. Section 4 describes models with more general radio coverage patterns, and some hybrid models. The implications of connectivity on the achievable capacity are discussed in Section 5 Section 6 considers the construction of a small connected relay set, such that the packet delivery can be achieved by forwarding packets using only sensors in the relay set. Section 7 covers the optimal placement of sensor nodes, which has a fundamental impact on the connectivity and other operational requirements of WSNs. Section 8 summarizes this survey

Protocol Design

RFS divides the problem of real-time flow scheduling into two parts (see Figure 1). First, we consider the problem of scheduling the transmissions of a single flow in isolation. RFS will construct plans according to which all instances of a flow are executed. A plan is the sequence of transmissions required to deliver data from the flow's source to its destination over multiple hops. The planner accounts for unreliable links and enforces the precedence constraints introduced by multi-hop forwarding during the construction of plans. Next, we consider the problem of scheduling multiple flows concurrently. RFS's dynamic scheduler executes multiple flows concurrently based on their temporal properties and the previously constructed plans. The scheduler dynamically determines the transmissions that will be executed in each slot such that no conflicting transmissions are scheduled in the same slot and prioritization among flows is provided. The division of the problem in two parts has several intrinsic advantages: (1) RFS isolates the concerns of handling precedence constraints and link unreliability (handled by the planner) from the concerns of handling interference and providing prioritization (handled by the scheduler). (2) RFS separates the process of constructing plans from their dynamic execution allowing us to develop a computationally efficient scheduler. (3) RFS executes flows dynamically based on their temporal properties rather than constructing an explicit transmission schedule. Therefore, flows may be added/removed without reconstructing an explicit schedule. RFS works as follows: (1) Any node may initiate the creation of a new flow that has it as a source. The node first checks whether an existing plan may be used to execute the new flow. As discussed in Section IV-A, it is often possible to reuse plans existing plans to execute new flows. When this is not possible, the planner initiates the construction of a plan for the new flow. (2) Next, admission control is performed on the source to determine whether the new flow may be added without any flows missing their deadlines. (3) At run-time, the scheduler dynamically executes flows based on their plans and temporal properties. The remainder of the section is organized as follows. We start by considering the problem of constructing plans that account for unreliable links (see Section IV-A). Next, we present the design and analysis of the centralized RFS

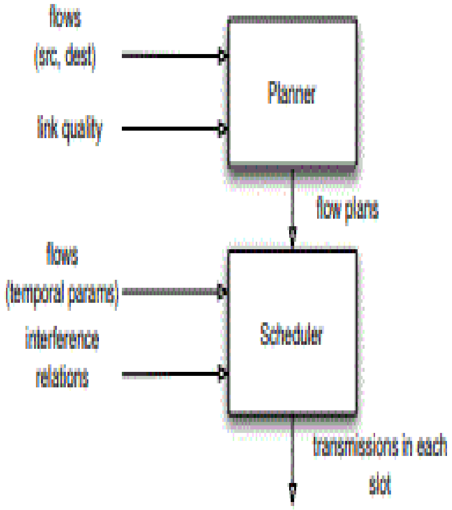


Fig. 1. RFS has two key components: a planner and a scheduler

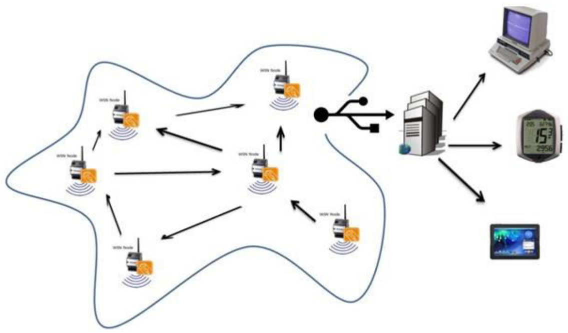
Plans The plan of flow i is an ordered sequence of steps that contains the transmissions necessary to forward a packet from the source to the destination of flow i. A plan is a sequence of steps such that: (1) a single transmission is assigned in each step and (2) the order of transmissions respects the constraints of hop-by-hop forwarding. All instances of a flow are executed according the same plan. We use the following notations: Πi denotes the plan of flow i, Πi [s] refers to the transmission assigned to step s of Πi , and Li is the plan's length. An example of a plan is shown in Fig. 3. In the case when links are perfect, a plan is the routing path between the flow's source and destination. However, since all instances of a flow are executed according to the same plan, plans must be stable over time, otherwise plans would have to be reconstructed frequently. Unreliable links are usually handled through Automatic Repeat reQuest (ARQ). The ARQ mechanism automatically retransmits a packet that is unacknowledged up to a maximum number of retransmissions. Existing TDMA protocols do not coordinate their activity with the link layer. As a result, retransmitted packets are usually queued up for an additional TDMA frame until the sender is scheduled to transmit. This introduces significant delays when packets are retransmitted multiple times. An alternative is to increase the slot size to allow for retransmissions. However, since nodes are synchronized on slots boundaries, a TDMA protocol is forced to treat all links uniformly. Overestimating the number of retransmissions lowers throughput while underestimating it results in packet drops over low quality links. RFS accounts for link unreliability by allowing a node to be assigned to multiple steps. In contrast to ARQ, we allow a maximum number of transmissions (MNT) to be specified per link. A number of link estimators evaluate the quality of a link using Expected Transmission Count (ETX) [23]. It is tempting to use ETX as an estimate of MNT. However, the ETX provided by the link layer estimates the average MNT. To ensure that plans remain stable over time, we are interested in estimating the worst-case MNT. The worst-case MNT may be estimated using Jacobson's algorithm [24]: Jacobson's algorithm computes both the average and standard deviation of ETX and then combines the two components

With the routing path of each packet, many measurement and diagnostic approaches are able to conduct effective management and protocol optimizations for deployed WSNs consisting of a large number of unattended sensor nodes. For example, PAD depends on the routing path information to build a Bayesian network for inferring the root causes of abnormal phenomena.

- Path information is also important for a network manager to effectively manage a sensor network. For example, given the per-packet path information, a network manager can easily find out the nodes with a lot of packets forwarded by them, i.e., network hop spots. Then, the manager can take actions to deal with that problem, such as deploying more nodes to that area and modifying the routing layer protocols.
- Furthermore, per-packet path information is essential to monitor the fine-grained per-link metrics. For example, most existing delay
- and loss measurement approaches assume that the routing topology is given as *a priori*.
- The time-varying routing topology can be effectively obtained by per-packet routing path, significantly improving the

values of existing WSN delay and loss tomography approaches.

SYSTEM ARCHITECTURE:



.

The growing network scale and the dynamic nature of wireless channel make WSNs become increasingly complex and hard to manage.

- The problem of existing approach is that its message overhead can be large for packets with long routing paths.
- Considering the limited communication resources of WSNs, this approach is usually not desirable in practice.

PROPOSED SYSTEM:

In this paper, we propose iPath, a novel path inference approach to reconstruct routing paths at the sink side. Based on a real-world complex urban sensing network with all node generating local packets, we find a key observation: It is highly probable that a packet from node and *one of* the packets from 's parent will follow the same path starting from 's parent toward the sink. We refer to this observation as *high path similarity*.

- The basic idea of iPath is to exploit high path similarity to iteratively infer long paths from short ones. iPath starts with a known set of paths (e.g., the one-hop paths are already known) and performs path inference iteratively. During each iteration, it tries to infer paths one hop longer until no paths can be inferred.
- In order to ensure correct inference, iPath needs to verify whether a short path can be used for inferring a long path. For this purpose, iPath includes a novel design of a lightweight hash function. Each data packet attaches a hash value that is updated hop by hop. This *recorded hash value* is compared against the *calculated hash value* of an inferred path. If these two values match, the path is correctly inferred with a very high probability.
- In order to further improve the inference capability as well as its execution efficiency, iPath includes a fast bootstrapping algorithm to reconstruct a known set of paths.

Conclustion:

We observe high path similarity in a real-world sensor network. It's an iterative boosting algorithm for efficient path inference. It's a lightweight hash function for efficient verification within iPath. The proposed system further propose a fast bootstrapping algorithm to improve the inference capability as well as its execution efficiency. iPath achieves higher reconstruction ratio under different network settings compared to states of the art.

REFERENCE:

[1] T. F. Abdelzaher, S. Prabh, and R. Kiran. On Real-Time Capacity Limits of Multihop Wireless Sensor Networks. In Proceedings of the 25th IEEE International Real-Time Systems Symposium(RTSS), 2004.

[2] atemu - sensor network emulator/simulator/debugger. http://www.hynet.umd.edu/research/atemu/.

[3] Q. Cao, T. Yan, J. Stankovic, and T. Abdelzaher. Analysis of Target Detection Performance for Wireless Sensor Networks. In Proceedings of the International Conference on Distributed Computing in Sensor Networks (DCOSS 2005), pages 84–89. TeX Users Group, June 2005.

[4] J. Elson and A. Parker. Tinker: A Tool for Designing Data-Centric Sensor Networks. In Proceedings of the Fifth Information Processing in Sensor Networks, Track on Sensor Platform, Tools and Design Methods for Networked Embedded Systems (IPSN SPOTS), April 2006. [5] P. H. Feiler, D. P. Gluch, and J. J. Hudak. The Architecture Analysis and Design Language (AADL): An Introduction. Technical Report CMU/SEI-2006-TN-011, Software Engineering Institute, Carnegie Mellon University, February 2006.

[6] N. Fournel, A. Fraboulet, G. Chelius, E. Fleury, B. Allard, and O. Brevet. Worldsens: from lab to sensor network application development and deployment. In Proceedings of the 6th international conference on Information processing in sensor networks, pages 551– 552. ACM Press, 2007.

[7] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. EmStar: a Software Environment for Developing and Deploying Wireless Sensor Networks. In USENIX General Track, 2004.

[8] J. Hatcliff, W. Deng, M. Dwyer, G. Jung, and V. Prasad. Cadena: An Integrated Development, Analysis, and Verification Environment for Component-based Systems. In Proceedings of the International Conference on Software Engineering (ICSE), May 2003.

[9] T. He, P. Vicaire, T. Yan, Q. Cao, G. Zhou, L. Gu, L. Luo, R. Stoleru, J. A. Stankovic, and T. F. Abdelzaher. Achieving Long-Term Surveillance in VigilNet. In Proceedings of the IEEE INFOCOM, April 2006.

[10] B. Henderson-Sellers et al. UML - the Good, the Bad or the Ugly? Perspectives from a panel of experts. Software and System Modeling, 4(1):4–13, February 2005.

[11] Z. Li. Communication and Schedulability Analysis in Wireless Sensor Network. Master's project report, University of Virginia, 2004.

[12] L. Luo, T. He, G. Zhou, L. Gu, T. A. Abdelzaher, and J. A. Stankovic. Achieving Repeatability of Asynchronous Events in Wireless Sensor Networks with EnviroLog. In Proceedings of the IEEE INFOCOM, April 2006. [13] V. Prasad. ANDES: an ANalysis-based DEsign tool for wireless Sensor networks. Master's thesis, University of Virginia, August 2007.

[14] R. Ramadan, K. Abdelghany, and H. El-Rewini. SensDep: A Design Tool for the Deployment of Heterogeneous Sensing Systems. In Proceedings of the Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS), pages 44–53, April 2006.

[15] Rhapsody. http://www.ilogix.com/sublevel.aspx?id=53.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

[16] J. A. Stankovic. VEST: A Toolset for Constructing and Analyzing Component Based Embedded Systems. Lecture Notes in Computer Science, 2001.

[17] STOOD. http://www.ellidiss.com/stood.shtml.

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

[19] TimeWiz Model and Analyze System Performance. http://www.bitpipe.com/detail/res/1103108790 56.h-tml.

[20] S. Vestal. MetaH Support for Real-Time Multi-Processor Avionics. In Proceedings of the Joint Workshop on Parallel and Distributed RealTime Systems (WPDRTS/ OORTS), 1997.

. Kishore, Department of Computer Science and Engineering, Ramachandra college of Engineering, Eluru, Affiliated to JNTU Kakinada.

P. Chakradhar, Associate Professor, Department of Computer Science and Engineering, Ramachandra College of Engineering, Eluru.

Dr. V. Surya Narayana Head of the Department in Computer Science and Engineering, Ramachandra College of Engineering, Eluru.