ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Wireless Sensor Networks Path Inference in: I Path

M.Soumya
M.Tech Student,
Gokul Institute of Technology & Sciences,
Piridi, Vizianagaram, Bobbili, Andhra Pradesh.

G Bhagya Lakshmi
Assistant Professor,
Gokul Institute of Technology & Sciences,
Piridi, Vizianagaram, Bobbili, Andhra Pradesh.

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.

EXISTING SYSTEM:

- 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.

DISADVANTAGES OF EXISTING SYSTEM:

- 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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

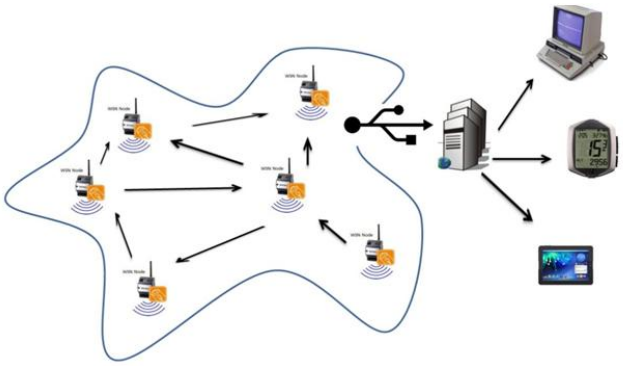
PROPOSED SYSTEM:

- 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.

ADVANTAGES OF PROPOSED SYSTEM:

- 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.

SYSTEM ARCHITECTURE:



MODULES:

- Network Model
- Iterative Boosting
- PSP-Hashing
- Performance Analysis

MODULES DESCSRIPTION:

Network Model

- In the first module, we design the Network Model Module. We assume a multi-hop WSN with a number of sensor nodes.
- Each node generates and forwards data packets to a single sink. In multi-sink scenarios, there exist multiple routing topologies.
- The path reconstruction can be accomplished separately based on the packets collected at each sink. In each packet , there are several data fields related to iPath.
- The first two hops of the routing path, origin and parent. Including the parent information in each packet is common best practice in many real applications for different purposes like network topology generation or passive neighbor discovery.
- The path length. It is included in the packet header in many protocols like CTP. With the path length, iPath is able to filter out many irrelevant packets during the iterative boosting.
- A hash value of packet 's routing path. It can make the sink be able to verify whether a short path and

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

a long path are similar. The hash value is calculated on the nodes along the routing path by the PSP-Hashing.

- The global packet generation time and a parent change counter. These two fields are not required in iPath. However, with this information, iPath can use a fast bootstrapping algorithm to speed up the reconstruction process as well as reconstruct more paths.

Iterative Boosting

- iPath reconstructs unknown long paths from known short paths iteratively. By comparing the recorded hash value and the calculated hash value, the sink can verify whether a long path and a short path share the same path after the short path's original node.
- When the sink finds a match, the long path can be reconstructed by combining its original node and the short path.
- There are two procedures, the Iterative-Boosting procedure and the Recover procedure. The Iterative-Boosting procedure includes the main logic of the algorithm that tries to reconstruct as many as possible packets iteratively.
- The input is an initial set of packets whose paths have been reconstructed and a set of other packets. During each iteration, is a set of newly reconstructed packet paths. The algorithm tries to use each packet in to reconstruct each packet's path. The procedure ends when no new paths can be reconstructed.
- The Recover procedure tries to reconstruct a long path with the help of a short path. Based on the high path similarity observation, the following cases describe how to reconstruct a long path.

PSP-Hashing

- The PSPHashing (i.e., path similarity preserving) plays a key role to make the sink be able to verify whether a short path is similar with another long path. There are three requirements of the hash function.
- The hash function should be lightweight and efficient enough since it needs to be run on resource-constrained sensor nodes.
- The hash function should be order-sensitive. That is, hash(A, B) and hash(B, A) should not be the same.
- The collision probability should be sufficiently low to increase the reconstruction accuracy.
- Traditional hash functions like SHA-1 are order-sensitive. However, they are not desirable due to their high computational and memory overhead. We propose PSP-Hashing, a lightweight path similarity preserving hash function to hash the routing path of each packet.
- PSP-Hashing takes a sequence of node ids as input and outputs a hash value. Each node along the routing path calculates a hash value by three pieces of data. One is the hash value in the packet that is the hash result of the subpath before the current node. The other two are the current node id and the previous node id. The previous node id in the routing path can be easily obtained from the packet header

Performance Analysis

- The fast bootstrapping algorithm reconstructs the routing path of a packet hop by hop. When the sink reconstructs the path of a packet to a forwarder, it can reconstruct the next-hop only when the packet is in one of stable periods. Therefore, the probability of a successful reconstruction of the fast bootstrapping algorithm is the product of the ratios of stable periods on all forwarding nodes.
- We can calculate the probability of a successful reconstruction by multiplying the probabilities there exists at least one shorter helper path at several hops.
- In iPath, the computational overhead at the node side is negligible since there are only several arithmetic operations. MNT, Pathfinder, and Pathzip do not require high computational

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

overhead at the node side either. At the PC side, the time complexity of iPath is polynomial.

SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

- System : Pentium Dual Core.
- Hard Disk : 120 GB.
- Monitor : 15'' LED
- Input Devices : Keyboard, Mouse
- Ram : 1GB.

SOFTWARE REQUIREMENTS:

- Operating system : Windows 7.
- Coding Language : JAVA
- Tool : ECLIPSE

INPUT DESIGN AND OUTPUT DESIGN

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2.It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

IMPLEMENTATION

MODULES:

- Network Model
- Iterative Boosting
- PSP-Hashing
- Performance Analysis

MODULES DESCSRIPTION:

Network Model

- In the first module, we design the Network Model Module. We assume a multi-hop WSN with a number of sensor nodes.
- Each node generates and forwards data packets to a single sink. In multi-sink scenarios, there exist multiple routing topologies.
- The path reconstruction can be accomplished separately based on the packets collected at each sink. In each packet , there are several data fields related to iPath.
- The first two hops of the routing path, origin and parent. Including the parent information in each packet is common best practice in many real applications for different purposes like network topology generation or passive neighbor discovery.
- The path length. It is included in the packet header in many protocols like CTP. With the path length, iPath is able to filter out many irrelevant packets during the iterative boosting.
- A hash value of packet 's routing path. It can make the sink be able to verify whether a short path and a long path are similar. The hash value is calculated on the nodes along the routing path by the PSP-Hashing.
- The global packet generation time and a parent change counter. These two fields are not required in iPath. However, with this information, iPath can use a fast bootstrapping algorithm to speed up the reconstruction process as well as reconstruct more paths.

Iterative Boosting

- iPath reconstructs unknown long paths from known short paths iteratively. By comparing the recorded hash value and the calculated hash value, the sink can verify whether a long path and a short path share the same path after the short path's original node.
- When the sink finds a match, the long path can be reconstructed by combining its original node and the short path.
- There are two procedures, the Iterative-Boosting procedure and the Recover procedure. The Iterative-Boosting procedure includes the main logic of the algorithm that tries to reconstruct as many as possible packets iteratively.
- The input is an initial set of packets whose paths have been reconstructed and a set of other packets. During each iteration, is a set of newly reconstructed packet paths. The algorithm tries to use each packet in to reconstruct each packet's path. The procedure ends when no new paths can be reconstructed.
- The Recover procedure tries to reconstruct a long path with the help of a short path. Based on the high path similarity observation, the following cases describe how to reconstruct a long path.

PSP-Hashing

- The PSPHashing (i.e., path similarity preserving) plays a key role to make the sink be able to verify whether a short path is similar with another long path. There are three requirements of the hash function.
- The hash function should be lightweight and efficient enough since it needs to be run on resource-constrained sensor nodes.

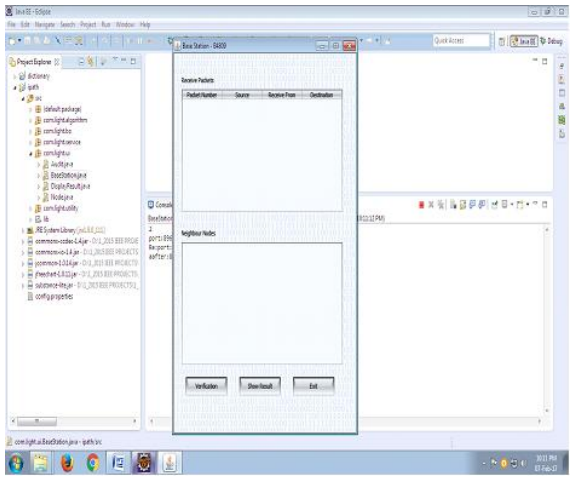
ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

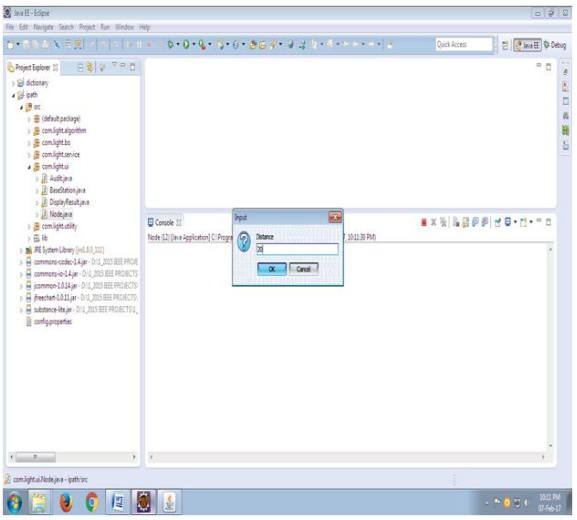
- The hash function should be order-sensitive. That is, hash(A, B) and hash(B, A) should not be the same.
- The collision probability should be sufficiently low to increase the reconstruction accuracy.
- Traditional hash functions like SHA-1 are order-sensitive. However, they are not desirable due to their high computational and memory overhead. We propose PSP-Hashing, a lightweight path similarity preserving hash function to hash the routing path of each packet.
- PSP-Hashing takes a sequence of node ids as input and outputs a hash value. Each node along the routing path calculates a hash value by three pieces of data. One is the hash value in the packet that is the hash result of the subpath before the current node. The other two are the current node id and the previous node id. The previous node id in the routing path can be easily obtained from the packet header

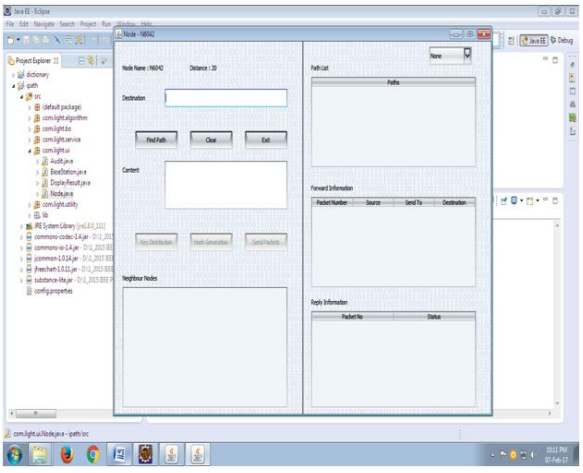
Performance Analysis

- The fast bootstrapping algorithm reconstructs the routing path of a packet hop by hop. When the sink reconstructs the path of a packet to a forwarder, it can reconstruct the next-hop only when the packet is in one of stable periods. Therefore, the probability of a successful reconstruction of the fast bootstrapping algorithm is the product of the ratios of stable periods on all forwarding nodes.
- We can calculate the probability of a successful reconstruction by multiplying the probabilities there exists at least one shorter helper path at several hops.
- In iPath, the computational overhead at the node side is negligible since there are only several arithmetic operations. MNT, Pathfinder, and Pathzip do not require high computational overhead at the node side either. At the PC side, the time complexity of iPath is polynomial.

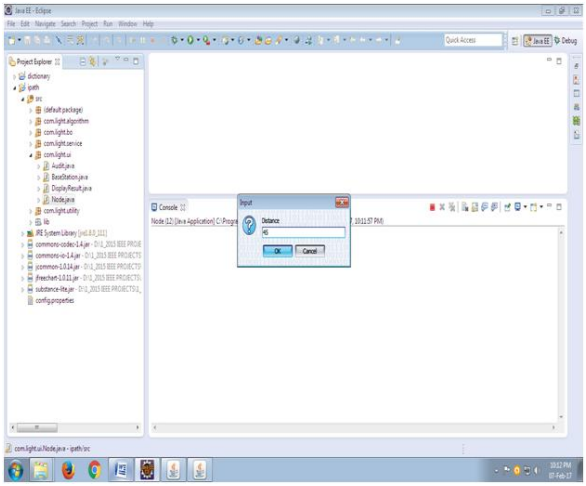
SCREEN SHOTS

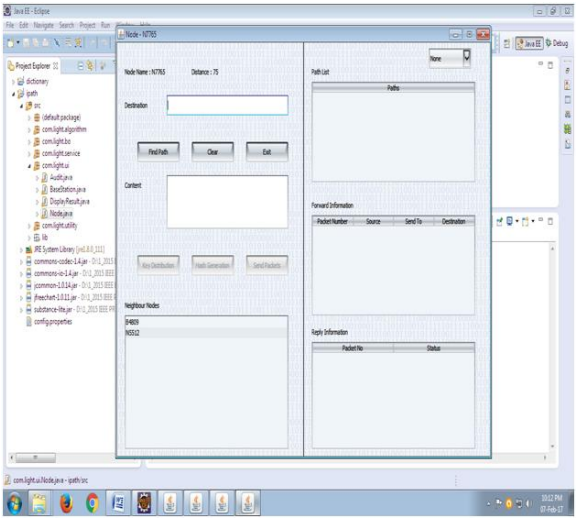


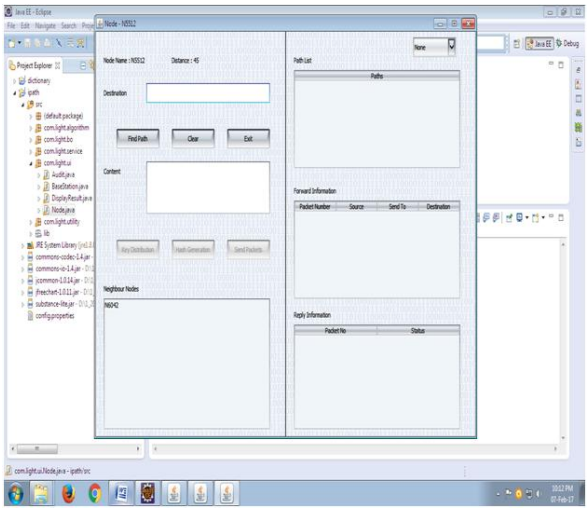


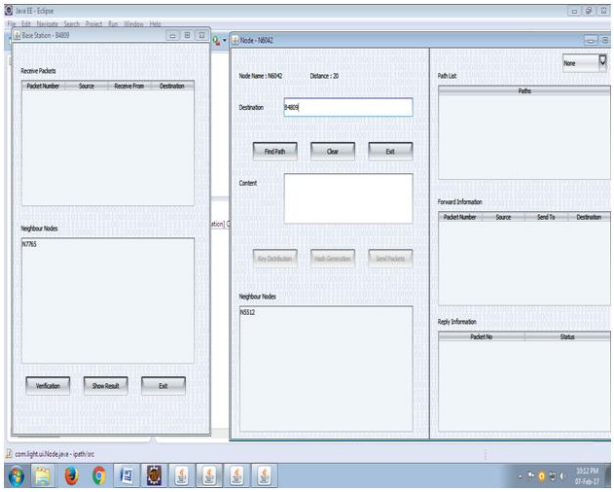


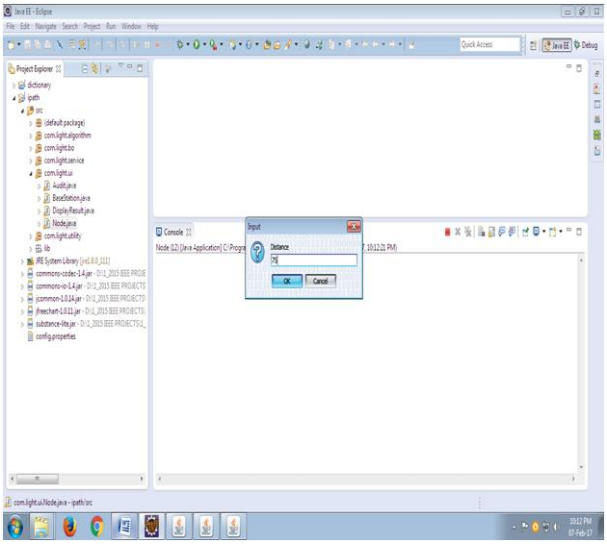
ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

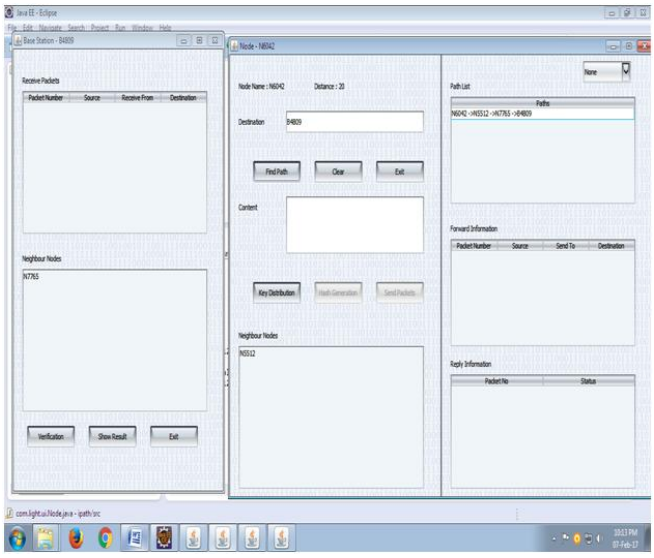




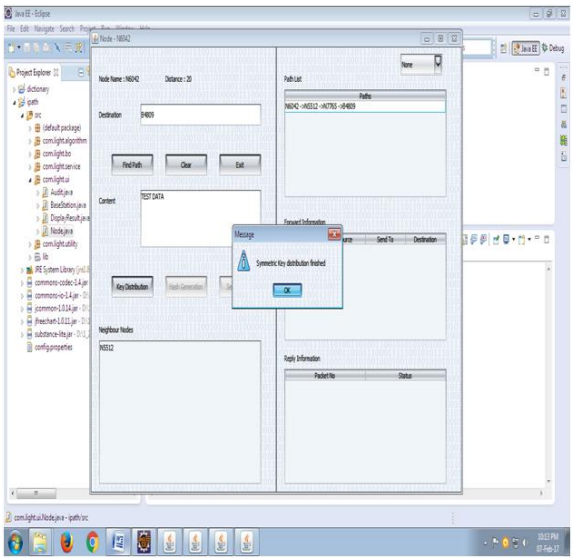


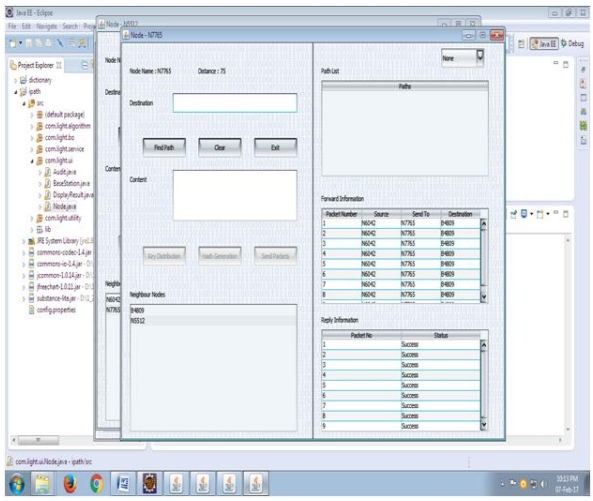


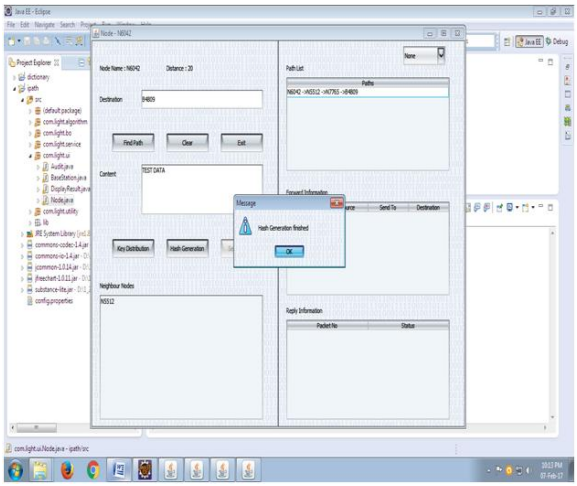


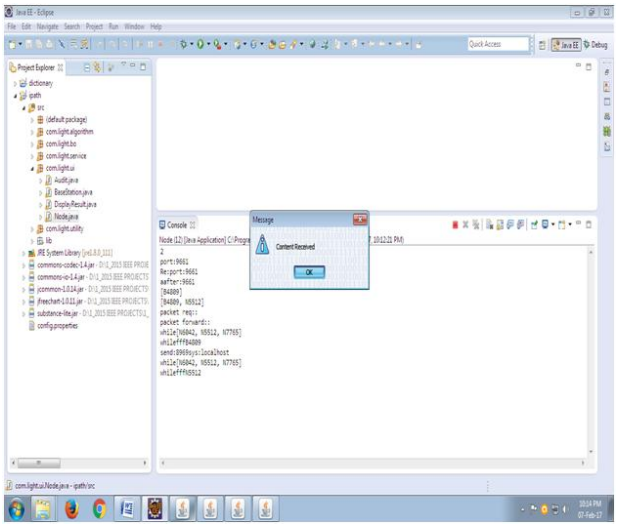


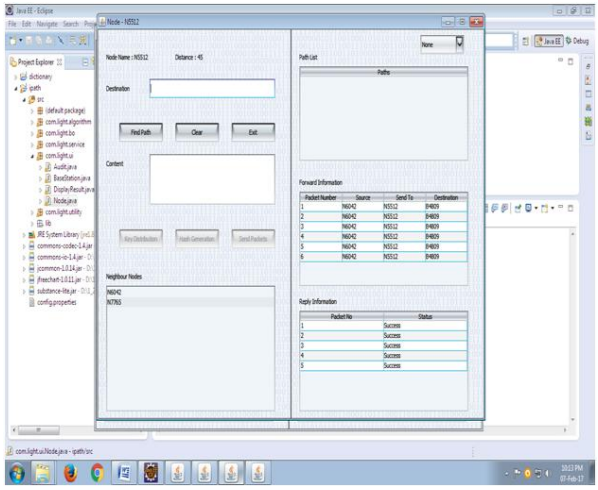
ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

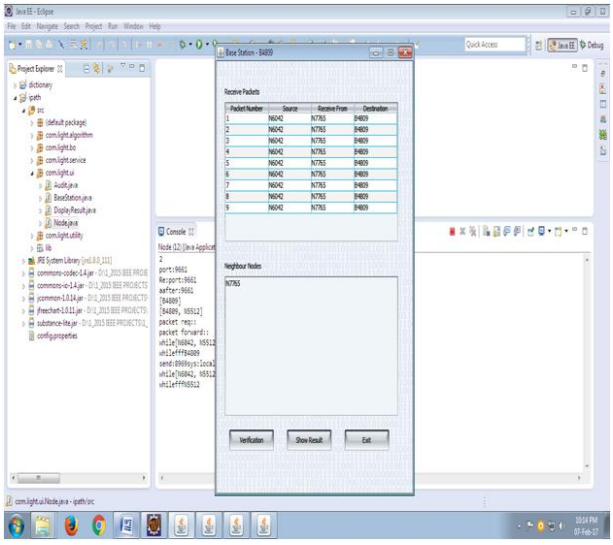




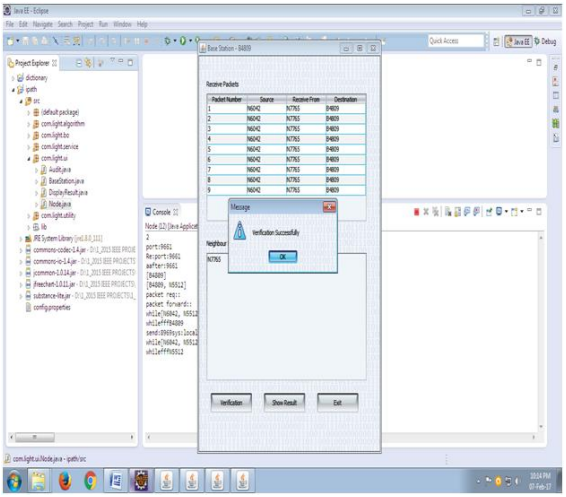


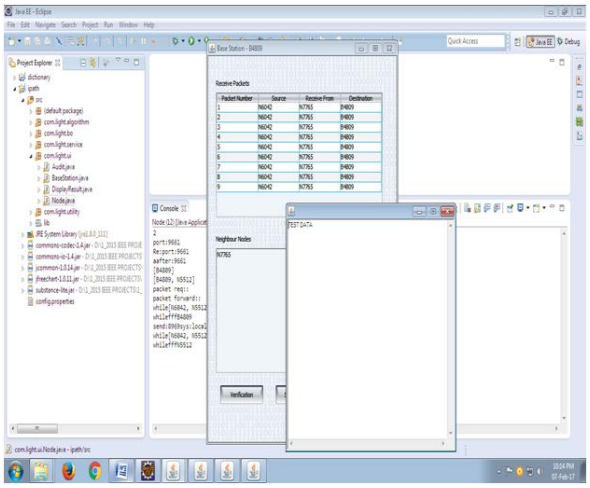






ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal





CONCLUSION:

In this paper, we propose iPath, a novel path inference approach to reconstructing the routing path for each received packet. iPath exploits the path similarity and uses the iterative boosting algorithm to reconstruct the routing path effectively. Furthermore, the fast bootstrapping algorithm provides an initial set of paths for the iterative algorithm. We formally analyze the reconstruction performance of iPath as well as two related approaches. The analysis results show that iPath achieves higher reconstruction ratio when the network setting varies. We also implement iPath and evaluate its performance by a trace-driven study and extensive simulations. Compared to states of the art, iPath achieves much higher reconstruction ratio under different network settings.

REFERENCES

[1] M. Ceriottiet al., "Monitoring heritage buildings with wireless sensor networks: The Torre Aquila deployment," in Proc. IPSN, 2009, pp. 277–288.

[2] L. Mo et al., "Canopy closure estimates with GreenOrbs: Sustainable sensing in the forest," in Proc. SenSys, 2009, pp. 99–112.

[3] X. Mao et al., "CitySee: Urban CO2 monitoring with sensors," in Proc. IEEE INFOCOM, 2012, pp. 1611–1619.

[4] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in Proc. SenSys, 2009, pp. 1–14.

[5] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high throughput path metric for multi-hop wireless routing," in Proc. MobiCom, 2003, pp. 134–146.

[6] Z. Li, M. Li, J. Wang, and Z. Cao, "Ubiquitous data collection for mobile users in wireless sensor networks," in Proc. IEEE INFOCOM, 2011, pp. 2246–2254.

[7] X. Lu, D. Dong, Y. Liu, X. Liao, and L. Shanshan, "PathZip: Packet path tracing in wireless sensor networks," in Proc. IEEE MASS, 2012, pp. 380–388.

[8] M. Keller, J. Beutel, and L. Thiele, "How was your journey? Uncovering routing dynamics in deployed sensor networks with multi-hop network tomography," in Proc. SenSys, 2012, pp. 15–28.

[9] Y. Yang, Y. Xu, X. Li, and C. Chen, "A loss inference algorithm for wireless sensor networks to improve data reliability of digital ecosystems.," IEEE Trans. Ind. Electron., vol. 58, no. 6, pp. 2126–2137, Jun. 2011.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[10] Y. Liu, K. Liu, and M. Li, "Passive diagnosis for wireless sensor networks," IEEE/ACM Trans. Netw., vol. 18, no. 4, pp. 1132–1144, Aug. 2010.

[11] W. Dong, Y. Liu, Y. He, T. Zhu, and C. Chen, "Measurement and analysis on the packet delivery performance in a large-scale sensor network," IEEE/ACM Trans. Netw., 2013, to be published.

[12] J. Wang, W. Dong, Z. Cao, and Y. Liu, "On the delay performance analysis in a large-scale wireless sensor network," in Proc. IEEE RTSS, 2012, pp. 305–314.

[13] Y. Liang and R. Liu, "Routing topology inference for wireless sensor networks," Comput.Commun.Rev., vol. 43, no. 2, pp. 21–28, 2013.

[14] Y. Gaoet al., "Domo: Passive per-packet delay tomography in wireless ad-hoc networks," in Proc. IEEE ICDCS, 2014, pp. 419–428.

[15] M. Lee, S. Goldberg, R. R. Kompella, and G. Varghese, "Fine-grained latency and loss measurements in the presence of reordering," in Proc. ACM SIGMETRICS, 2011, pp. 329–340.

[16] Y. Shavitt and U. Weinsberg, "Quantifying the importance of vantage points distribution in internet topology measurements," in Proc. IEEE INFOCOM, 2009, pp. 792–800.

[17] M. Latapy, C. Magnien, and F. Oudraogo, "A radar for the internet," in Proc. IEEE ICDMW, 2008, pp. 901–908.

[18] I. Cunha, R. Teixeira, D. Veitch, and C. Diot, "Predicting and tracking internet path changes," in Proc. SIGCOMM, 2011, pp. 122–133.

[19] A. D. Jaggard, S. Kopparty, V. Ramachandran, and R. N. Wright, "The design space of probing algorithms for network-performance measurement," in Proc. SIGMETRICS, 2013, pp. 105–116.

[20] L. Ma, T. He, K. K. Leung, A. Swami, and D. Towsley, "Identifiability of link metrics based on end-to-end path measurements," in Proc. IMC, 2013, pp. 391–404.

[21] Y. Gaoet al., "Pathfinder: Robust path reconstruction in large scale sensor networks with lossy links," in Proc. IEEE ICNP, 2013, pp. 1–10.

[22] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in Proc. SenSys, 2003, pp. 14–27.

[23] Y. Gaoet al., "iPath: Path inference in wireless sensor networks," Tech. Rep., 2014 [Online]. Available: http://www.emnets.org/pub/gaoyi/tech-ipath.pdf

[24] A. Liu and P. Ning, "TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks," in Proc. IPSN, 2008, pp. 245–256.

[25] V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "TWIST: A scalable and reconfigurable testbed for wireless indoor experiments with sensor networks," in Proc. REALMAN, 2006, pp. 63–70.

[26] R. Lim, C. Walser, F. Ferrari, M. Zimmerling, and J.Beutel, "Distributed and synchronized measurements with FlockLab," in Proc. SenSys, 2012, pp. 373–374.

[27] Z. Li, M. Li, and Y. Liu, "Towards energy-fairness in asynchronous duty-cycling sensor networks," Trans. Sensor Netw., vol. 10, no. 3, pp. 38:1–38:26, 2014.

[28] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in Proc. SenSys, 2003, pp. 126–137.