Deployment Alternatives for Performance Debugging in Wireless Sensor Networks

Tony O'Donovan and Cormac J. Sreenan
Mobile and Internet Systems Laboratory,
Department of Computer Science,
University College Cork
Cork, Ireland
Email: t.odonovan@cs.ucc.ie

***Abstract*—A common approach for performance monitoring and diagnosis in wireless sensor networks (WSNs) is to send meta-data to a sink node to process. WSN radio constraints limit the amount of this meta-data that can be sent. Logging it to the node's onboard storage can also aid in long-term performance debugging. Using the stored data it is possible for nodes to do statistical analysis for the detection of performance anomalies in the network, rather than at the sink. In this paper we compare the cost and accuracy of performing anomaly detection in the network and at the sink.**

I. Introduction

Most of the research to date in sensor networks has been in areas such as environmental monitoring and agriculture, where performance assurances are not considered essential. The goal of the FP7 funded GINSENG project [1], on the other hand, is to develop a WSN to meet application-specif c performance targets in real industry settings where performance is critical. Such deployments require tools that can help in the detection and analysis of faults and anomolies in the network when they occur.

In networks such as this, both bandwidth and energy are at a premium. The common performance monitoring approach of frequently sending meta-data to the sink can be costly. In this paper we use the node's onboard storage and computational capabilities to log and analyse recent traff c patterns that can indicate problems or faults in the network. Carrying out this analysis on the node's themselves rather than at the sink allows us to reduce the number of performance related messages that are sent to the sink. We show that this can provide considerable savings, particularly for multi-hop networks.

The GINSENG project includes a test bed in an operating oil ref nery at Sines, Portugal where part of our evaluation is performed. Due to the possibility of explosion the nodes need to be enclosed in ATEX boxes [2]. The presence of the oil tanks and pipes that make up the ref nery also provide a diff cult environment for radio communications. All of this together with the application scenario constraints, constitute a very challenging deployment.

II. Motivation

Environmental conditions, software errors and component failures are among the many causes for the poor performance of wireless sensor networks in real world deployments. During the development phase it may be necessary to provide extra infrastructure to help with detecting, diagnosing and f xing such problems. The GINSENG project, for example, uses embedded PCs running Linux connected to each node. This allows access to the serial logs of the nodes as well as the ability to reprogram them remotely. However, such a system is only feasible for a small number of nodes and can not be supported beyond the development phase.

Due to the distributed nature of WSNs each node must send a periodic update to the sink so that a representation of the status of the network is available to the operator. The accuracy of that representation is determined by the frequency of the updates that the nodes send. However, there is a large overhead associated with sending periodic performance updates to the sink.

By logging performance related data to the nodes' onboard Flash storage it is possible to carry out statistical analysis in the network [3]. Using the data logged to Flash the nodes can check for changes in routing or traff c patterns that often indicate performance anomalies. If a supplementary update is sent when a performance anomaly is detected, it is possible to reduce the frequency of periodic status updates to the sink without incurring any signif cant loss of accuracy in the available representation of network status. As well as the energy saving associated with fewer status updates there is also more bandwidth available to application messages.

III. Design

The frequency of performance messages sent by a node affects the accuracy of the representation of the network held at the sink. A high update rate provides a clearer picture of the network's current state for the operator but comes with overhead associated with additional network traff c, a low rate means less traff c but the condition of the network available to the operator may be out of date and of little use. The desirable scenario is a low update rate without the loss of accuracy.

It is possible to achieve this goal by getting the node to check for anomalous traff c behaviour based on stored data and send supplementary performance messages if an anomaly is encountered. Given that GinMAC [4] is TDMA based and the application has a periodic nature, it is to be expected that the number of messages sent and received by a node in a

978-1-4244-9289-3/11/\$26.00 ©2011 IEEE

given period should not vary considerably from one period to the next. Any sudden increase or decrease in traff c in such a system is a key indicator of performance problems or network changes. Interference, disconnected nodes and topology changes all result in a changed message sent/received rate. With this in mind, it was decided to periodically log the number of messages sent to and received by a node to Flash and use this log as a basis for the node's usual traff c conditions. Any large deviation from this norm generates an exception and a supplementary performance message to update the sink.

IV. Implementation

While we wish to reduce the overhead associated with sending performance monitoring data to the sink, some state information is required at the sink, so low-rate periodic updates are still necessary. The frequency of performance message transmission is reduced, with the new lower frequency sending only 1 message for every 5 messages normally sent. To ensure the sink is aware of changes in traff c conditions a check for anomalies is performed and supplementary messages are sent if necessary.

Upon initialisation a f le is created on the node's Flash using the Coffee [5] f lesystem and an *fsUpdate* event is scheduled. The *fsUpdate* event updates the f le with the number of slots, the number of messages sent and the number of messages received since the last time the event was triggered. A circular array is used to capture a recent snapshot of the node's traff c so that bursts can be detected easier. The standard deviation and average number of messages sent and received are then calculated based on the records in the f le stored in Flash. Finally a check is performed to see if the current traff c conditions are consistent with those of the stored data.

Equation 1 is for messages sent:

$$2SD_s < Abs(A_s - C_s) \tag{1}$$

where SD_s, A_s and C_s denote the standard deviation, average and current number of messages sent in a period respectively.

Similarly exceptions in the rate of messages received is checked using Equation 2

$$2SD_r < Abs(A_r - C_r) \tag{2}$$

where SD_r, A_r and C_r denote the standard deviation, average and current number of messages received in a period respectively.

If an exception is encountered then an exception message is generated and added to the queue. Exception messages are marked so the sink can distinguish easily between exceptions and regular messages sent at the lower frequency.

V. Evaluation

The evaluation was carried out using the GINSENG testbed at the Petrogal oil ref nery at Sines, Portugal and a laboratory testbed in University College Cork (UCC). The Sines deployment consists of 16 TelosB nodes running a version of Contiki 2.4 [6] that has been heavily modif ed for the GINSENG project including GinMAC, overload control and performance debugging etc. The network includes a section where nodes are widely spread with long communication ranges and a section where there is a high concentration of nodes with short communication ranges. Nodes operate at different heights from the ground in areas with large metal structures, which can seriously affect wireless communication.

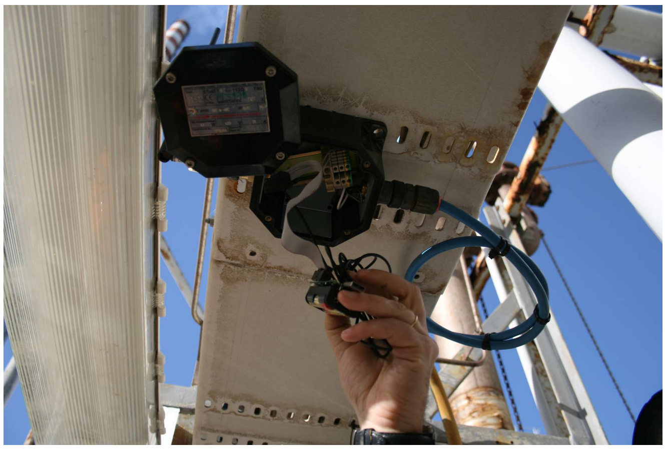
Fig. 1. A node in the Sines tesbed.

Each node (excluding the Sink) is contained within an ATEX box as a safety precaution and then individually connected to existing sensors located throughout the testing area of the ref nery, as shown in Figure 1. Nodes are connected to a number of different sensor types including pressure, temperature and f ow transmitter. During the current development and evaluation phase the nodes are also connected to embedded PCs running Linux to allow for remote reprogramming.

A PC running Ubuntu Linux is connected to one of the nodes and is used as a sink. This sink is located in a small portable off ce in the ref nery. The other 15 nodes make up a network with a static 3-2-1 topology as shown in Figure 2.

The UCC testbed also conists of 16 TelosB nodes that use the same software and the same topology as the Sines deployment. Apart from the physical environment, the laboratory testbed is as close a representation of the Sines testbed as possible.

Each node generates an application message once per second and performance messages once every 25 seconds at the default high frequency. The low frequency for sending performance messages is $^1/_5th$ of the high rate, i.e. one message every 125 seconds.

A. Evaluation Results

Of interest is the difference in accuracy of the sink's representation of the nodes' state that comes from reducing the performance message update rate. This is for the low update rate, both with and without exceptions. The cost of sending the updates at both rates is also of interest, as is the

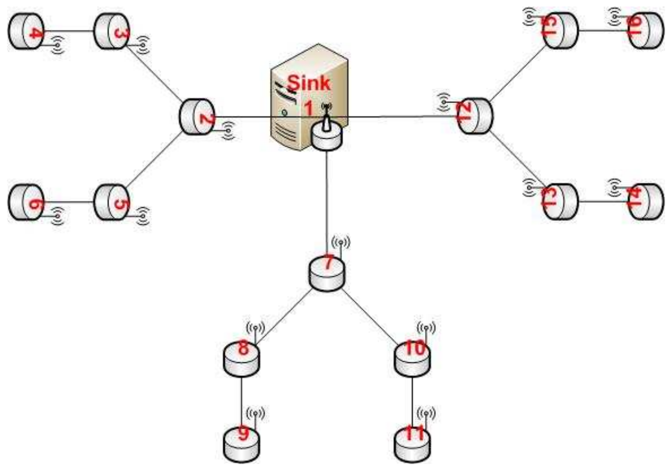

Fig. 2. The GINSENG test bed 3-2-1 topology.

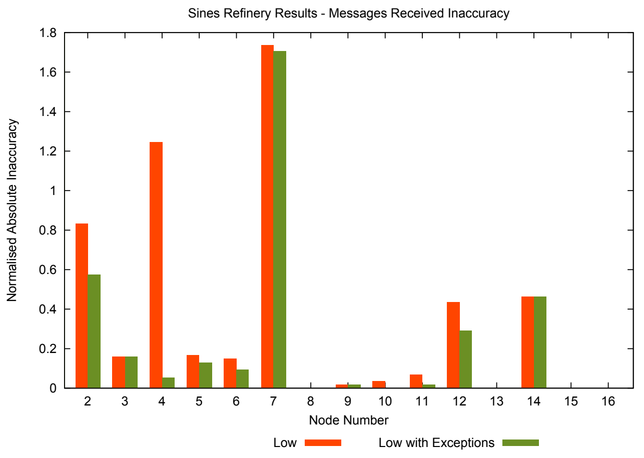

Fig. 4. Detection Inaccuracy Summary - Messages Received.

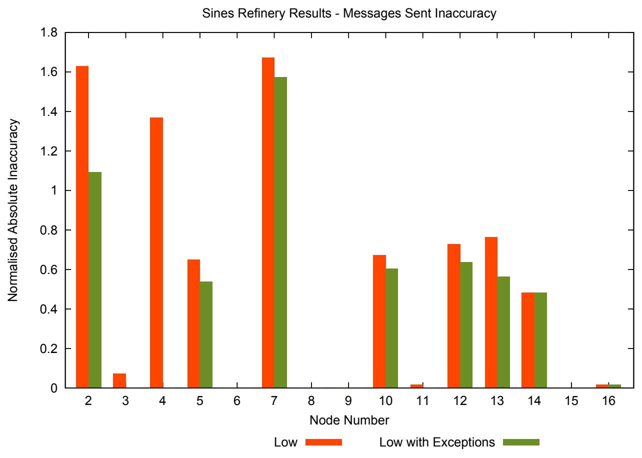

Fig. 3. Detection Inaccuracy Summary - Messages Sent.

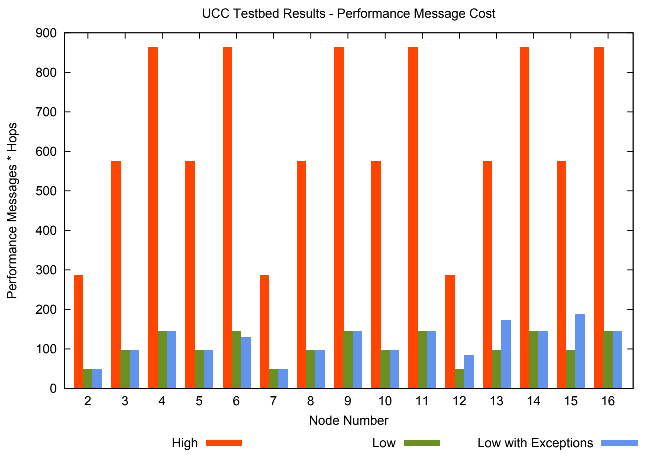

Fig. 5. Performance Debugging Overhead per Node (Communication).

additional cost of monitoring for performance changes and sending supplementary updates if a threshold is breached.

1) Accuracy: This experiment was carried out on the Sines testbed. We wanted to measure how much the performance data at the sink differed from that of the high update rate when we use the low update rate with and without exceptions. Since the performance data at the high rate was being used as our *normal*, it was necessary to use the high update rate for all the messages. Exception messages and those sent at the low rate were marked, so they could be distinguished from the high rate messages and each other.

In order to determine the accuracy we took the number of messages sent/received at the high frequency as normal and then calculated how much this number differed from that of both the low frequency and the low with exceptions. The average inaccuracy for each node is shown for messages sent and messages received in Figure 3 and Figure 4 respectively, with a decrease in inaccuracy ref ecting an increase in accuracy. The accuracy for most nodes is improved; this improvement is considerable for nodes 2 and 4.

2) Cost: This experiment was carried out on the UCC testbed. Three two hour runs were done, f rst with the high performance update frequency (every 25 seconds), then with the low frequency (every 125 seconds) and f nally the low frequency with exceptions. The number of messages required to send each node's performance updates to the sink is shown in Figure 5, including the cost associated with forwarding the performance updates. The difference between the high and low is signif cant indicating a considerable energy saving is possible by reducing the update frequency. However, the lower frequency including exceptions has only a slightly higher cost than the low frequency.

As well as the overhead associated with sending the exception messages, the cost of storing performance data in Flash must be taken into account. Figure 6 shows a breakdown of the energy used during the run with the low performance update rate and exceptions. On average Flash read and write operations added just $0.02mW$ in two hours, accounting for under 10% of the nodes total energy usage.

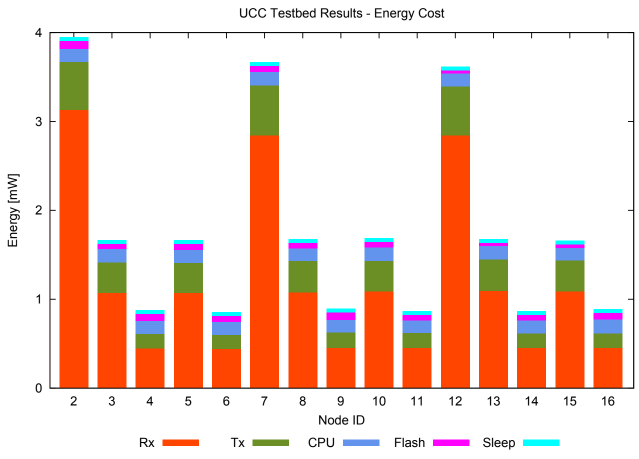

Fig. 6. Energy Breakdown per Node.

VI. Related Work

Sympathy is a popular tool that provides network-monitoring and is particularly suited to periodic data gathering WSNs [7]. The sink gathers performance related data from all nodes in the network for analysis. PAD and PerDB employ a different approach, these schemes include a small amount of performance information in each application message [8], [9]. The causal diagrams and belief networks used in PAD are like Sympathy's decision tree. Weighted decision trees are used to calcuate the Visibility metric, which aids in the design of network protocols suited for debugging [10].

Our scheme, along with Sympathy, PAD and PerDB are among many that monitor for changes in traff c patterns and routing to detect faults and anomolies in networks. While the others rely on the sink to perform the analysis and fault detection, we use the storage and computational capabilities of the nodes to perform the detection on the nodes themselves. One of the main goals of this work is to reduce the overhead of performance debugging, as in PAD and PerDB. However, unlike those systems that discard data not included in application messages, we log data to the nodes' Flash allowing it to be used in detection of faults.

Statistical analysis can be used for application-level data integrity, rather than checking for performance anomolies in the communication layers as in this work. For example, invalid data from faulty or compromised nodes are detected using outliers by Ganeriwal et al. [11]. Suelo analyses sensor data to look for predef ned features that can indicate problems with the sensors [12].

Like this work, Envirolog employs a node's Flash memory to log data in order to improve performance [13]. The authors' aim is to be able to later replay logged events, allowing analysis of network's performance. We use the logged data to detect performance anomolies in the network itself as it happens.

VII. Conclusion

Periodic updates of performance meta-data is commonly used to provide WSNs with performance monitoring and debugging capabilities. Frequent updates can be costly in terms of both energy and bandwidth. We present a scheme where nodes store performance data in Flash, which is used to check for anomolous traff c conditions. Using this scheme it is possible to reduce the rate at which performance updates are sent to the sink without a major loss of accuracy. This work is part of the GINSENG project that aims to develop WSN systems for industrial environments where performance is critical. Part of the evaluation was carried out on the GINSENG testbed in an operation oil ref nery in Sines Portugal.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 224282.

References

[1] "GINSENG," Web page, visited 2011-07-01. [Online]. Available: http://www.ict-ginseng.eu/

[2] "ATEX," Web page, visited 2011-07-01. [Online]. Available: http://ec.europa.eu/enterprise/sectors/mechanical/documents/guidance/atex

[3] T. O'Donovan, N. Tsiftes, Z. He, T. Voigt, and C. J. Sreenan, "Detailed diagnosis of performance anomalies in sensornets," in *Proceedings of the Sixth ACM Workshop on Hot Topics in Embedded Networked Sensors (HotEmNets2010)*, June 2010.

[4] P. Suriyachai, J. Brown, and U. Roedig, "Time-critical data delivery in wireless sensor networks," in *Proceedings of the Sixth IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '10)*, June 2010.

[5] N. Tsiftes, A. Dunkels, Z. He, and T. Voigt, "Enabling large-scale storage in sensor networks with the coffee f le system," in *Proceedings of the Eighth ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN 2009)*, April 2009.

[6] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - a lightweight and f exible operating system for tiny networked sensors," in *Proceedings of the IEEE Workshop on Embedded Networked Sensor Systems (EmNets)*, November 2004.

[7] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," in *Proceedings of the Third International Conference on Embedded Networked Sensor Systems (SenSys '05)*, November 2005.

[8] K. Liu, M. Li, X. Yang, and M. Jiang, "Passive diagnosis for wireless sensor networks," in *Proceedings of the Sixth ACM Conference on Embedded Networked Sensor Systems (SenSys '08)*, November 2008.

[9] V. Pejovic and C. J. Sreenan, "Perdb: Performance debugging for wireless sensor networks," in *Proceedings of the Sixth European Conference on Wireless Sensor Networks (EWSN 2009), Poster/Demo session*, February 2009.

[10] M. Wachs, J. I. Choi, J. W. Lee, K. Srinivasan, Z. Chen, M. Jain, and P. Levis, "Visibility: a new metric for protocol design," in *Proceedings of the Fifth International Conference on Embedded Networked Sensor Systems (SenSys '07)*, November 2007.

[11] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, June 2008.

[12] N. Ramanathan, T. Schoellhammer, E. Kohler, K. Whitehouse, T. Harmon, and D. Estrin, "Suelo: human-assisted sensing for exploratory soil monitoring studies," in *Proceedings of the Seventh ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*, November 2009.

[13] L. Luo, T. He, G. Zhou, L. Gu, J. Stankovic, and T. Abdelzaher, "Achieving repeatability of asynchronous events in wireless sensor networks with envirolog," in *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM '06)*, April 2006.