Evaluating and Bounding Operations Performance in Heterogeneous Sensor and Actuator Networks with Wireless Components

José Cecílio, João Costa, Pedro Martins, Nickerson Ferreira, and Pedro Furtado

University of Coimbra,
Coimbra, Portugal
{jcecilio,jpcosta,pmom,nickerson,pnf}@dei.uc.pt

Abstract. When wireless sensor networks are introduced in industrial settings, they will be part of a much larger heterogeneous sensor and actuation network that includes those sub-networks together with Ethernet cabled Programmable Logic Controllers (PLCs) and control stations. Performance issues arise in such systems. We propose mechanisms to measure, verify, control and debug expected operation time bounds in such heterogeneous sensor and actuator networks.

Keywords: WSAN, Heterogeneity, Operations Timing Guarantees.

1 Introduction

A sensor and actuation infrastructure is typically a heterogeneous environment, consisting of software running on computer-boards, control stations, and multiple networked sensors and actuators. It can be composed of multiple parts: wireless sensor sub-networks will do the sensing and actuation over parts of the plant, and they will be part of a much larger heterogeneous network that includes those sub-networks together with Ethernet cabled computer-boards and control stations.

Network and control engineers deploying and putting to work such a system will need to control and verify whether time bounds are within acceptable values. There are many options involved, and there is the need for both planning and testing. For instance, with the appropriate Time Division Multiple Access (TDMA) schedule, it is possible to implement simple closed-loops within a single wireless sensor network, with guaranteed behavior. On the other hand, it is also possible to have situations where a closed-loop is required with sensing happening in a wireless sensor network, decision logic in a computer, and actuation happening in another wireless sensor network. In this case the control loop will traverse multiple, most probably non-real-time hardware and software systems, nevertheless the control loop will still need to be under verifiable expected time bounds.

In this paper we define measures and metrics for surveillance of expectable time bounds and an approach for debugging, using tools and mechanisms to explore and report problems. This surveillance can be used in any distributed system to verify

J. Cichoń, M. Gębala, and M. Klonowski (Eds.): ADHOC-NOW 2013, LNCS 7960, pp. 111–122, 2013.

performance compliance. Assuming that we have monitoring or closed-loop tasks with timing requirements, this allows users to constantly monitor timing conformity.

We will also show experimental results concerning bounds and debugging tool. We create a simulation environment where we introduce some random delays in the messages, to demonstrate how the debugging tool works and its usability. These practical experimental results will show the values obtained in that environment, proving that the network and control engineers will have the adequate measures, metrics and tools to plan and debug adequately.

The rest of the paper is organized as follows: section 2 reviews related work; section 3 describes the heterogeneous monitoring and control architecture. It explains the architecture components considered in this work. Section 4 defines measures and metrics used by our approach to evaluate and bound operations performance in heterogeneous sensor networks. Section 5 describes the addition of debugging modules to the heterogeneous monitoring and control architecture. The debugging node component and operation performance monitor component are described. It is also described how the performance information is collected and processed. An example of operation performance monitor UI is presented, which allows users to evaluate the performance. In section 6 we define the experimental setup and section 7 shows results obtained and the conclusions. Lastly, section 8 concludes the paper.

2 Related Work

Our proposal is an approach to verify and control expected operation time bounds in a heterogeneous system such as Sensor and Actuator Networks (SANs) with wireless sensor sub-networks. Related work includes monitoring tools, studies on performance, latency and delay analysis.

Monitoring is crucial to control operations performance of a sensor and actuator network, and tools are needed to measure specific parameters of performance. These methods are used to monitor all necessary parameters that assure that all functionalities are working as expected.

There already exist some tools to monitor network performance in wireless sensor networks. Sympathy [1] is a monitoring system in which sensor nodes are supplied with specific monitoring software, and periodically send specific parameters to a sink node. The mechanism developed is aimed at detecting and debugging failures in sensor networks and is specifically designed to be used with data gathering applications. All evaluation is done at the sink. When a failure occurs, Sympathy triggers failure localization and reporting.

DiMo [2] presents a distributed and scalable solution for monitoring the nodes and the topology, along with a redundant topology for increased robustness. The aim is to operate in safety-critical wireless sensor networks, where all sensor nodes must be up and functional. This tool provides two functionalities, network topology maintenance and network health status monitoring. Monitoring is done by using observer nodes that check the reception of heartbeats within a certain monitoring time. If not, the observer sends a node missing message to the sink. There is always one observer for each node. The sink is always aware of which nodes are being observed in the network, and therefore always knows which nodes are up and running.

Our work is related to these ones in what concerns network performance monitoring. However our approach is oriented towards high-level operations, where we propose operation time failure detection mechanisms.

There are also some works addressing latency and delays for WSNs [3, 4, 5, 6]. These works have considered the extension of the Network Calculus methodology [7] to WSNs. End-to-end delay bounds for real-time flows in WSNs have been studied in [8]. The authors propose closed-form recurrent expressions for computing the worst-case end-to-end delays, buffering and bandwidth requirements across any source-destination path in the cluster-tree assuming error free channel. They propose and describe a system model, an analytical methodology and software tool that permits the worst-case dimensioning and analysis of cluster-tree WSNs.

There also exist approaches to monitor latencies and delays in distributed control systems based on wired components. The authors of [9] and [10] show two studies on modeling and analyzing latency and delay stability of network control systems. The authors of [11] modeled end-to-end time delay dynamics for the internet using system identification tools. The study in [12] presents an analytical performance evaluation of the switched Ethernet with multiple levels from timing diagram analysis, and experimental evaluation from an experimental testbed with a networked control system. These works assume a wired network. However, a distributed control system may include wireless and wired parts. Our proposed approach includes application-level, end-to-end message losses, message delivery delays, commands delays, specification of bounds to provide high degree of performance in all components of the SAN (wired and wireless).

3 Heterogeneous Monitoring and Control Architecture

With the evolution and increased adoption of wireless sensor technology and networks, and their easier and much cheaper deployment, there is a current solution to partially replace or complement the existing infrastructure. When deployed in industrial settings, it will build a heterogeneous sensor and actuation network. The global network can be composed by wireless sensor networks (WSN), PLC, computers and control stations (Fig. 1).

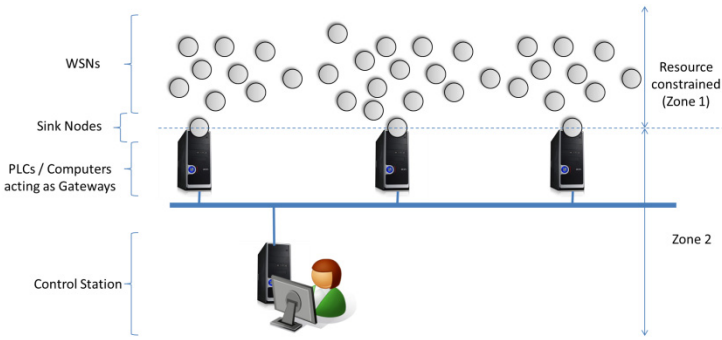

Fig. 1. General Architecture

One way to try to provide operation timing guarantees is to deploy WSN sub-networks with real-time specific algorithms that would include at least completely pre-planned synchronous time-division mechanisms. However, sensor and actuation infrastructures are typically heterogeneous systems. The system includes specific software parts, some possibly offering real-time, while others do not. In that context, deployment of sensor and actuator networks requires a high level of reliability control concerning measures such as latencies, delays and message losses. It is important to ensure that monitoring and control loops will still be under specified time bounds. For instance, control engineers can specify operation time bounds to deliver the data to or from supervision controllers.

4 Measures and Metrics

Operation timing issues in terms of monitor and closed-loops control can be controlled with the help of two measures, which we denote as Latency and Delay of Periodic Events.

Latency consists of the time required to travel between a source and a destination. Sources and destinations may be any site in the distributed system. For instance, the latency can be measured from a WSN leaf node to a sink node, or from a WSN sensing node to a computer, control station or backend application.

Delay of Periodic Events consists of the extra time taken to receive a message with respect to the predefined periodic reception instant.

Given the above time measures, we define metrics for sensing and control. The metrics allow us to quantify timing behavior of monitoring and closed-loops.

Monitoring Latency – the time taken to deliver a value from sensing node to the control station, for display or alarm computation. The following latency metrics are therefore all considered: Acquisition latency, Transmission latency, Control Station processing latency, End-to-end latency.

Monitoring Delay – the amount of extra time from the moment when a periodic operation was expected to receive some data to the instant when it actually received. When users create a monitoring task, they must specify a sensing rate. The control station expects to receive the data at that rate, but delays may happen in the way to the control station, therefore delays are recorded.

Event-Based Closed-Loop Latency – the time taken from sensing node to actuator node passing through the supervision control logic. It will be the time taken since the value (event) happens at a sensing node to the instant when the action is performed at the actuator node. The following latency metrics should be considered to determine the closed-loop latency: Acquisition latency, Upstream transmission latency, Control Station processing latency, Downstream transmission latency, Actuator processing latency, End-to-end latency.

Periodic Closed-Loops Latency – periodic closed-loops can be associated with two latencies: Monitoring latency and Actuation latency. The Monitoring latency can be defined as the time taken from sensing node to the supervision control logic. The Actuation latency corresponds to the time taken to reach an actuator. We also define

an end-to-end latency as the time from the instant when a specific value is sensed and the moment when an actuation is done which incorporates a decision based on that value. The following latency metrics should be considered to determine the closed-loop latency for synchronous or periodic closed-loops: Acquisition latency; Upstream transmission latency; Wait for the actuation instant latency; Control Station processing latency; Downstream transmission latency; Actuator processing latency; End-to-end latency;

Closed-Loop Delays – In periodic closed-loop operations, actuation is expected within a specific period. However, operation delays may occur in the control station and/or command transmission. The closed-loop delay is the excess time.

In event-based closed-loops, there can be monitoring delays. This means that a sample expected every x time units may be delayed.

5 Addition of Debugging Modules to Monitoring and Control Architecture

In the previous sections we defined measures and metrics useful to evaluate operation performance. In this section we will discuss how to add debugging modules to the monitoring and control architecture.

The architecture can be divided into two main parts: nodes and control station. To add debugging functionalities, we need to add a Debugging Module (DM) to nodes and a Performance Monitor Module (PMM) to the control station. The debugging module collects information from operations in nodes, then formats and forwards information to the Performance Monitor module. The Performance Monitor gathers the status information coming from nodes, stores it in a database and processes it according to bounds defined by the user.

In the next sections we describe how the Debugging module and Performance Monitor module work.

5.1 The Debugging Module

The Debugging module (DM) stores all information concerning node operation (e.g. execution times, battery level) and messages (e.g. messages received, messages transmitted, transmission fails, transmission latencies). This information is stored inside the node. It can be stored either in main memory, flash memory or other storage device.

DM is an optional module that can be activated or deactivated. It generates a debugging report, either by request or periodically, with a configurable period.

DM has two modes of operation:

- Network debugging – the DM runs in all nodes and keeps the header information of messages, where it adds timestamps corresponding to arrive and departure instants. After, this information is sent periodically or by request to the Performance Monitor (described in the next sub-section), which is able to calculate metrics. This operation mode may be deactivated in constrained devices, because it consumes resources such as memory and processing time.

- High-level operation debugging – instead of collecting, storing and sending all information to the Performance Monitor, the DM can be configured to only add specific timestamps to messages along the path to the control station.

Assuming a monitoring operation in a distributed control system with WSN sub-networks, where data messages are sent through a gateway, the DM can be configured to add timestamps in the source node, sink node, gateway and control station. Fig. 2 illustrates nodes, gateways and a control station in that context.

The approach assumes that WSN nodes are clock synchronized. However, they may not be synchronized with the rest of the distributed control system. Gateways, computers and control stations are also assumed clock synchronized (e.g. the NTP protocol can be used).

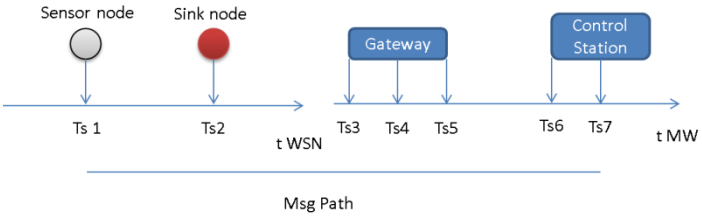

Fig. 2. Message path – example

In Fig. 2, the DM starts by adding a generation timestamp (source timestamp) in the sensor node (*Ts1*). When this message is received by the sink node, it adds a new timestamp (*Ts2*) and indicates to the gateway that a message is available to be written in the serial interface. Upon receiving this indication, the gateway keeps a timestamp that will be added to the message (*Ts3*), and the serial transmission starts. After concluding the serial transmission, the gateway takes note of the current timestamp (*Ts4*) and adds *Ts3* and *Ts4* to the message.

Upon concluding this process and after applying any necessary processing to the message, the gateway adds another timestamp (*Ts5*) and transmits it to the control station. When the message is received by the control station, it adds a timestamp (*Ts6*), processes the message and adds a new timestamp (*TS7*), which indicates the instant of message processing at the control station was concluded. After that, at the control station, the Performance Monitor module (described in the next section) receives the message and, based on the timestamps that come in the message, it is able to calculate metrics.

If there is only one computer node and the control station, there will only be *Ts1*, *Ts6* and *Ts7*.

5.2 The Performance Monitor Module and UI

In this sub-section we describe the Performance Monitor module (PMM), which debugs operations performance in the heterogeneous distributed system. The PMM stores events (data messages, debug messages), latencies and delays into a database. It collects all events when they arrive, computes metric values, classifies events with respect to bounds, and stores the information in the database. Bounds should be configured for the relevant metrics.

Assuming the example shown in Fig. 2, PMM collects the timestamps and processes them it to determine partial and end-to-end latencies.

The following partial latencies are calculated:

- WSN upstream latency (*Ts2 – Ts1*)
- WSN to Gateway interface latency (*Ts4 – Ts3*)
- Middleware latency (*Ts6 – Ts5*)
- Control station latency (*Ts7 – Ts6*)
- End-to-end ((*Ts2 – Ts1*) + (*Ts4 – Ts3*) + (*Ts6 – Ts5*) + (*Ts7 – Ts6*))

After concluding all computations, PMM stores the following information in the database: Source node id, Destination node id, Type of message, MsgSeqId, [Timestamps], partial latencies, end-to-end latency. This information is stored for each message, when the second operation mode of debugging is running. When the first operation mode of the debugging component is running, a full report with link-by-link information and end-to-end information is also stored.

The PMM user interface shows operations performance data, and alerts users when there is a problem detected by metric exceeds bounds. Statistical information is also shown and is updated for each event that arrives or for each timeout that occurs.

Fig. 3 shows a screenshot of PMM. We can see how many events (data messages) arrived in-time, out-of-time (with respect to defined bounds), and the corresponding statistical information. This interface also shows a pie chart to give an overall view of the performance.

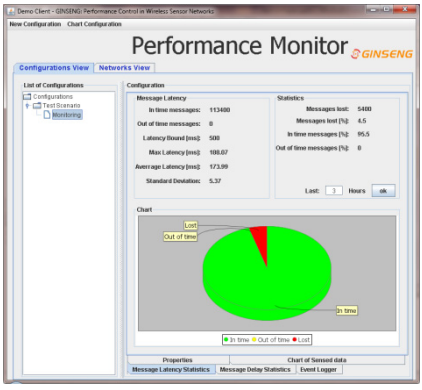

Fig. 3. PMM user interface

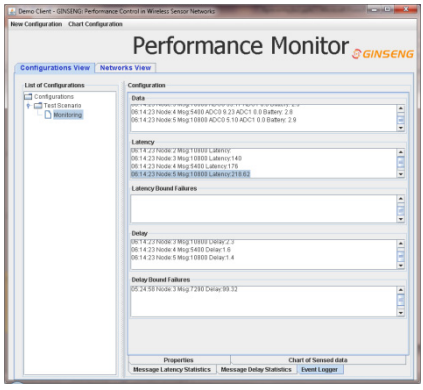

Fig. 4. PMM user interface – event logger

Fig. 4 shows the event logger of PMM. This logger shows the details on failed events. A list of failures is shown and the user can select one of each and see all details, including the latency in each part of the distributed control system.

When a problem is reported by the PMM, the user can explore the event properties (e.g. delayed messages, high latencies) and find where the problem occurs. If a problem is found and the debugging report is not available at the PMM, nodes are requested to send their debugging report. If a node is dead, the debugging report is not retrieved and the problem source may be due to the dead node. Otherwise, if all reports are retrieved, the PMM is able to detect the message path and check where it was discarded or where it took longer than expected.

PMM allows users to select one message and see all details, including the latency in each part of the distributed control system.

6 Experimental Setup

In this experimental section we use a testbed prototype to show that our approach is useful to monitor and debug operations performance in the whole SAN (WSN nodes, gateways, control stations and end-user applications).

In the testbed there is a WSN sub-network, which is a planned network designed to provide performance control, gateways (computer nodes) and control stations. Fig. 5 shows a sketch of our setup.

The WSN sub-network includes 12 TelosB nodes organized hierarchically in a 1-1-2 tree and one sink node (composed by one TelosB node and one computer that acts as gateway of the sub-network). The setup also includes a control station that receives the sensor samples, monitors operations performance using bounds, and collects data for testing the debugging approach.

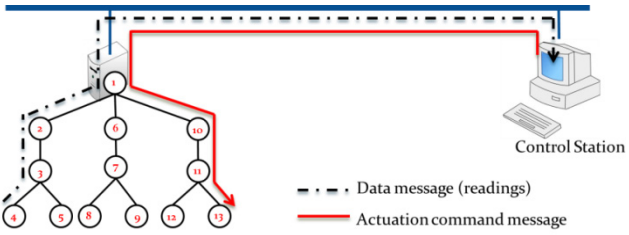

Fig. 5. Setup

The control station is a computer with an Intel Pentium D, running at 3.4 GHz. It has 2 GB of RAM and an Ethernet connection. The gateway connecting the WSN sub-network to the cabled network is another computer with similar characteristics.

All computer nodes (gateway and control station) are connected through Ethernet cables and GigaBit network adapters. The WSN sub-network is connected to the gateway using the serial interface provided by TelosB nodes. That interface is configured to operate at 460800 baud/second.

All computers run Linux OS and have specific components developed using Java to do specific tasks. For instance, the gateway computer has a gateway software component to read data from the serial interface and send it to the control station, and to receive messages destined for the WSN and deliver them through the serial interface. The control station has a software component which implements remote configuration and performs functionalities such as monitoring and closed-loop control.

The WSN sensor nodes run Contiki OS and generate one message per time unit with a specified sensing rate. Each message includes data measures such as temperature and light. GinMAC [13] is used at the MAC layer by the WSN nodes.

7 Testing Bounds and the Performance Monitoring Tool

To exercise the use of bounds and debugging, we created a monitoring operation and introduced a "liar" node which injects 10 ms of delay in the first of every two consecutive messages that travel through it. Using the setup shown in Fig. 5, we will replace node 3 by the "liar" node (Fig. 6).

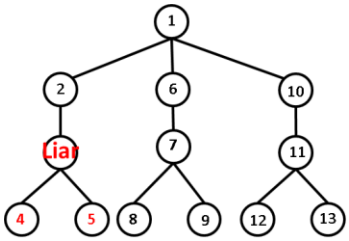

Fig. 6. Setup with "liar" node

Moreover, to simulate some losses in the network, we changed the node 4 configuration to consecutively send one message and discard the next message. This allows us to simulate 50% of message losses.

Fig. 7 and Fig. 8 show the results concerning message delays. Fig. 7 reports values concerning delay without the "liar" node, while Fig. 8 reports delays after replacement of node 3 by the "liar" node.

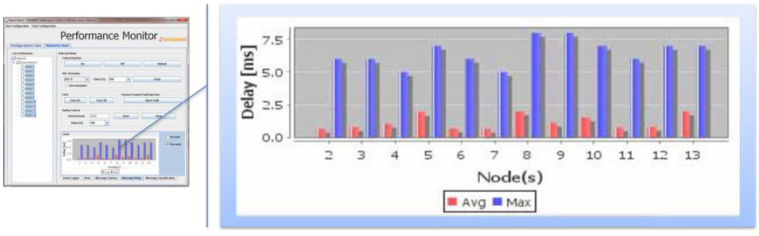

Fig. 7. Message delay without "liar" node

From Fig. 7 we can conclude that consecutive messages arrive at the control station, in average, within 0.5 to 2 ms. This value can grow up to a maximum of 8 ms.

After introducing our "liar" node and running the monitoring operation for 24 hours, we obtained the chart of Fig. 8. This figure shows that the delay of nodes 4 and 5 increased. These nodes send their messages to the control station passing through the "liar" node, which is node 3. In this case, we can see that the delay of two consecutive messages increased, in average, to 12 ms, and up to a maximum of 20 ms.

Using the PM proposed in this paper, we can also define bounds for the message delay. Assuming that each message should arrive at the control station within a maximum delay of 10 ms, we can define a delay bound and analyse the results.

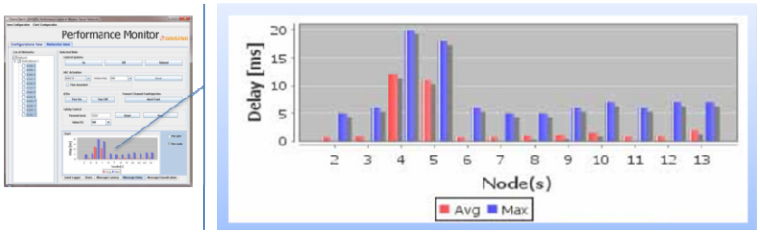

Fig. 8. Message delay with "liar" node

Fig. 9 shows the percentage of messages classified as in-time, out-of-time and lost, according to a delay bound of 10 ms and lost timeout of 1s (the timeout when a message is considered lost).

From Fig. 9 we conclude that 88.6% of the messages are delivered within the bound, but 6.8% are delivered out of bounds and 4.5% are lost. These numbers are as expected:

- Messages lost – there are 12 nodes sending data messages, node 4 fails one in every two messages. That results in $\frac{1}{12*2}$ losses, which agrees with the result of 4.5% losses that was obtained.
- Messages out of bound – there are 12 nodes sending data messages, node 4 sends only half of its messages and half of them arrive delayed. Concerning node 5, half of its messages arrive delayed. That results in $\frac{1}{12*3}+\frac{1}{12*2}$ messages out of bound, which agrees with the result of 6.8% out of bound that was obtained.
- Messages in time – 88.6% of messages are delivered in time, that results from the total number of expected received messages minus the number of losses and out of bound $\left(1-\left(\frac{1}{12*2}+\frac{1}{12*3}+\frac{1}{12*2}\right)\right)$.

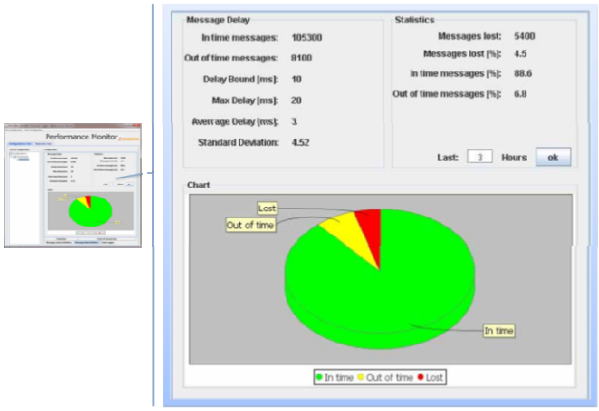

Fig. 9. Message classification according delay bound

The user can also explore event properties (in this case, delays) and find where the problem occurred. For instance, the user interface includes per node evaluation such as the one in Fig. 10, which shows which node(s) is failing.

From Fig. 10 we can conclude that node 4 is responsible for the losses represented in Fig. 9. It is losing 50% of the messages, as expected.

Fig. 10 also shows that the delay bound is not met by nodes 4 and 5. In this case, further debugging allows the user to identify the path of each message and to check where it took longer than expected.

For instance, if we explore the path and delay parts of node 5, we can conclude that messages sent by that node are waiting, in average, 10 ms in the transmission queue of node 3 (our "liar" node), which is greater than the expected average delay value of 2 ms for node 3 sending messages to the control station seen in Fig. 7.

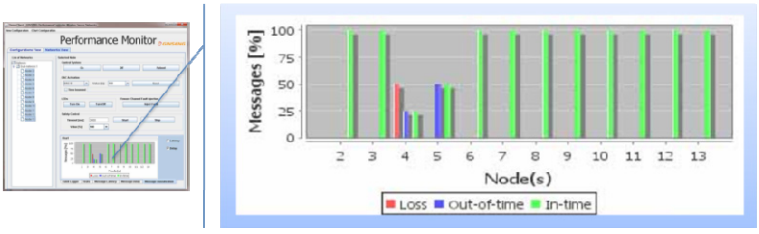

Fig. 10. Message classification according to delay bounds per node

8 Conclusion

In order to make sensor and actuator networks (SANs) more reliable in practical contexts with constraints such as latencies and delay bounds, there is a need for approaches to help a user to correctly evaluate and debug the performance problems in those SANs. This is especially true in heterogeneous systems with non real-time components, where it is very important to determine where and why problems occur.

The introduction of wireless sensor networks into industrial sites has created one heterogeneity case in which it is important to be able to track and debug problems.

We proposed an approach to monitor and debug latency and delay problems. The approach collects information on every part of the paths followed by messages and also on delays, and it combines detailed link-level information with timing information to allow users to track where the problems occurred and why. The approach also classifies messages according to bounds and provides feedback about how many messages arrived at the control station in-time or out-of-time. Our experimental results consisted in injecting timing failures into a testbed and showing that the approach allowed the users to detect the problems and track them until they reach the originator of the problem. Future work will consist in providing automated detailed problem tracking reports.

References

1. Ramanathan, N., Chang, K., Kapur, R., Girod, L., Kohler, E., Estrin, D.: Sympathy for the Sensor Network Debugger. Presented at ACM Sensys (2005)
2. Meier, A., Motani, M., Siquan, H., Künzli, S.: DiMo: Distributed Node Monitoring in Wireless Sensor Networks. Presented at 11th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2008), Vancouver (2008)
3. Koubaa, A., Alves, M., Tovar, E.: Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks. In: 27th IEEE International RealTime Systems Symposium, RTSS 2006, pp. 412–421 (2006)
4. Lenzini, L., Martorini, L., Mingozzi, E., Stea, G.: Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree networks. Performance Evaluation 63(9-10), 956–987 (2006)
5. Roedig, U., Gollan, N., Schmitt, J.: Validating the Sensor Network Calculus by Simulations. In: Network 2006 (2006)
6. Schmitt, J., Zdarsky, F., Roedig, U.: Sensor Network Calculus with Multiple Sinks. In: Proceedings of IFIP NETWORKING 2006 Workshop on Performance Control in Wireless Sensor Networks, pp. 6–13 (2006)
7. Thiran, P., Le Boudec, J.-Y. (eds.): Network Calculus. LNCS, vol. 2050. Springer, Heidelberg (2001), http://ica1www.epfl.ch/PS_files/netCalBookv4.pdf
8. Jurcik, P., Severino, R., Koubaa, A., Alves, M., Tovar, E.: Real-Time Communications Over Cluster-Tree Sensor Networks with Mobile Sink Behaviour. In: 14th IEEE International Conference on Embedded and RealTime Computing Systems and Applications, pp. 401–412 (2008)
9. Sato, K., Nakada, H., Sato, Y.: Variable rate speech coding and network delay analysis for universal transport network. In: Seventh Annual Joint Conference of the IEEE Computer and Communcations Societies Networks Evolution or Revolution, IEEE INFOCOM (1988)
10. Wu, J., Deng, F.Q., Gao, J.: Modeling and stability of long random delay networked control systems. In: International Conference on Machine Learning and Cybernetics, vol. 2, pp. 947–952 (2005)
11. Kamrani, E., Mehraban, M.H.: Modeling Internet Delay Dynamics Using System Identification. In: IEEE International Conference on Industrial Technology, pp. 716–721 (2006)
12. Lee, S., Lee, K., Lee, M., Harashima, F.: Integration of mobile vehicles for automated material handling using Profibus and IEEE 802.11 networks. IEEE Transactions on Industrial Electronics (2002)
13. Suriyachai, P., Brown, J., Roedig, U.: Time-Critical Data Delivery in Wireless Sensor Networks. In: Rajaraman, R., Moscibroda, T., Dunkels, A., Scaglione, A. (eds.) DCOSS 2010. LNCS, vol. 6131, pp. 216–229. Springer, Heidelberg (2010)