

Table 1 Different classes of application as defined by ISA

Category	Class	Application	Description
Safety	0	Emergency action	Always critical
Control	1	Closed-loop regulatory control	Often critical
	2	Closed-loop supervisory control	Usually noncritical
	3	Open-loop control	Human in loop
Monitoring	4	Alerting	Short-term operational consequence
	5	Logging and downloading/uploading	No immediate operational consequence

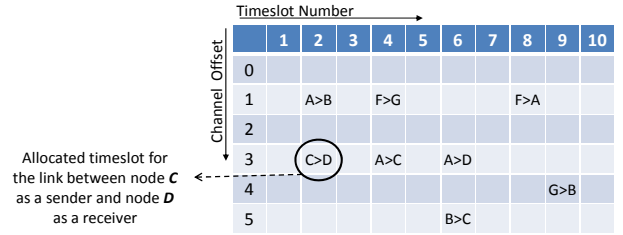
2. Motivation and application domain

During the last decade, lots of efforts have been undertaken to introduce suitable wireless technologies able to address the requirements of industrial monitoring and control applications. However, for improving these technologies or designing better ones, we still need a reference point. That way, we can evaluate those technologies based on certain metrics essential for large-scale industrial monitoring and control applications, including real-time capability, scalability, power consumption and robustness.

WirelessHART, as one of the first standards in the wireless sensor network domain that got approved by the IEC, can be considered as a reference point for evaluation based on different metrics. For instance, if we consider the real-time metric, based on the criticality and importance of the applications, the International Society of Automation (ISA) considers six classes of wireless communication, from critical control to monitoring applications. According to this classification, the WirelessHART standard can support industrial applications from class 2 to 5 [11].

The details of the classes are shown in Table 1, in which the importance of the message response time and Quality of Service (QoS) requirements varies [11]. In the more critical applications, process values need to be transmitted to the destination in a reliable, timely and accurate manner.

By having an accurate WirelessHART simulator, it is possible to achieve that reference point. That way it becomes feasible to improve the WirelessHART standard by modifying the WirelessHART stack or network system management algorithm. Furthermore by using different metrics, an evaluation of the other wireless protocols, through a comparison with WirelessHART, is also possible.

**Figure 1 Slot-Channel matrix**

3. Background

Before moving on to implementation details, the Time Synchronized Mesh Protocol (TSMP) and Time Slotted Channel Hopping (TSCH) need to be explained. Both are main concepts used in the WirelessHART protocol stack. Following this, the WirelessHART architecture is described.

3.1. Time Synchronized Mesh Protocol and Time Slotted Channel Hopping

WirelessHART is an IEEE 802.15.4 based standard which is designed to address the industrial process automation requirements, by using concepts derived from the TSMP [2]. TSMP, developed by DustNetworks, is a media access and networking protocol that is designed for low power and low bandwidth reliable communication. TSCH is a MAC scheme, which is a subset of TSMP. It enables robust communication through channel hopping, and high data rates through synchronization. TSCH is based on a time-slotted architecture, where a schedule dictates on what slot and which channel a node should transmit/receive data to/from a particular neighbor. Unlike TSMP, TSCH does not address routing issues but leaves this to the upper layers.

TSCH divides the wireless channel into time and frequency. Time is divided into discrete time slots. TSCH models the Radio Frequency (RF) space as a matrix of slot-channel cells. Figure 1 shows the TSCH approach.

TSCH uses the concept of a superframe: a collection of cells which repeat at regular intervals. For example, Figure 1 illustrates that a timeslot of a length of 10ms repeats once every 100ms, when the superframe consists of 10 slots. By scheduling each transaction (i.e. Tx-Rx operation) in one cell, the hidden terminal problem is prevented, as adjacent links never transmit simultaneously on the same frequency. A link is a transaction that occurs within a cell. It consists of a superframe ID, source and destination IDs, a slot number referenced to the beginning of the superframe, and a channel offset. The simplest version of a link contains one transmitter and one receiver. The two nodes at either end of the link communicate periodically once in every superframe. If only one transmitter is scheduled, the link

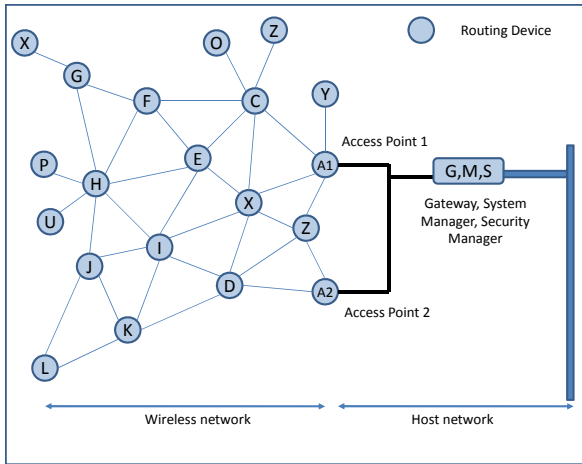


Figure 2 A sample of WirelessHART topology

is contention-free, but a slotted CSMA approach can be used if multiple transmitters are scheduled to use the same cell simultaneously. TSCH links hop pseudo-randomly over a set of predefined channels, one packet at a time. Each time a link is activated, both sides of the link calculate the radio channel of the communication by taking $(\text{Absolute Slot Number} + \text{Channel offset}) \% \text{Number of channels}$.

TSMP introduced graph-based routing. A graph is a routing structure that establishes directed end-to-end connection among devices. Each destination has its own graph, however several sources can share the same graph. Each source may have multiple graphs to communicate with each of its neighbours, so that in a single network, multiple graphs, some of which overlap, can be constructed. When a source node intends to send a packet, based on the QoS and final destination, the flow of the packet is identified and a fixed Graph ID is included in the packet header that routes it through the multiple paths to the destination. At any node in the path, multiple next hops could be specified in a mesh graph; path diversity is directly built-in [2].

3.2. WirelessHART architecture

A typical topology of the WirelessHART mesh network is shown in Figure 2. The basic types of nodes shown in this figure are:

Gateway, System/Network Manager, and Security Manager: These three node types are usually considered as a single component. This component connects the host network to the wireless network and is responsible for configuring the network, scheduling the communications and managing routes by using a centralized management algorithm.

Access Point: Two access points act as intermediate nodes thereby providing the interface between the wireless network and the Gateway.

Field Device: This type of node can be a sensor or an actuator, which is installed in the plant field.

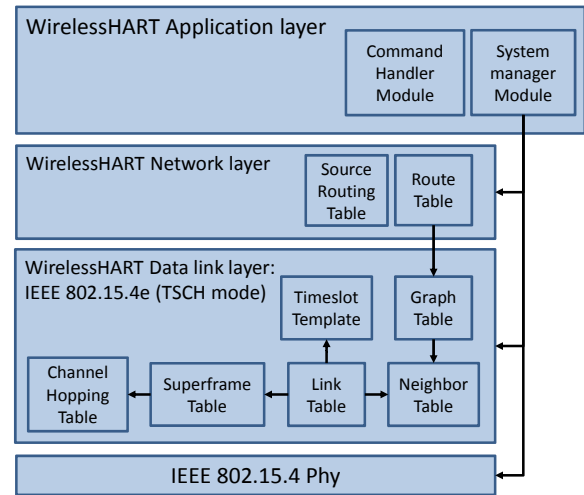


Figure 3 WirelessHART protocol stack

4. Implementation of WirelessHART

This section provides details about the implementation of WirelessHART protocol stack and the central network management algorithm.

4.1. WirelessHART protocol stack

In Figure 3 the protocol stack of the devices, and the access points is shown. All the field devices in the WirelessHART network have to support this stack.

The Physical layer of WirelessHART is the IEEE 802.15.4 standard physical layer which exists in the WPAN module of NS-2, so we did not modify this layer.

We modified the MAC layer of the IEEE 802.15.4 standard module of NS-2 to support network-wide time synchronization, channel hopping, dedicated slotted unicast communication bandwidth, link layer ACKs and concurrent link activation. In addition to the changes that are considered in IEEE 802.15.4 standard 2006, several new MAC layer management entity primitives (MLME), based on IEEE 802.15.4e¹ (TSCH mode), have been added to the existing IEEE 802.15.4 standard 2003 module in NS-2, as listed below: *mlme_set_slotframe*, *mlme_set_link*, *mlme_set_graph*, *mlme_tsch_mode*, *mlme_listen*, *mlme_advertise*, *mlme_keep_alive*, *mlme_join*, *mlme_activate*, and *mlme_disconnect*.

The collection of communication tables is implemented in the data link/MAC layer based on the IEEE 802.15.4e (TSCH mode) standard to enable communication and to control communication performance. These tables and their relationship are shown in the data link layer block in Figure 3. They are manipulated by the network manager through the previously mentioned MLME primitives.

¹ MAC amendment to the existing 802.15.4-2006 standard to support industrial requirements.

The network layer provides reliable end-to-end communication for network devices. In this layer we implement the Graph routing as well as Source routing. To do so the Route Table and Source Route Table structures are implemented in this layer. The tables and their relationships are shown in the network layer block in Figure 3. These tables are manipulated by the network manager and are used to deliver a packet to the destination.

The application layer of WirelessHART is a command based layer. Commands as the basis of HART communication, are sent from gateway or field devices and each command can be identified by a command number which determines the content of the message. The wireless commands are the collection of commands to support WirelessHART products. WirelessHART commands are in the range 768-1023 which can be used to support network management and gateway functions [8]. The WirelessHART commands that have been implemented in the simulator can be classified into several categories, including managing superframes and links commands, managing graph and source routes commands, bandwidth management commands, network health reporting and status commands.

4.2. WirelessHART network management algorithm

The WirelessHART System/Network manager uses centralized network management techniques for communication scheduling and to manage routes. However, this standard does not define the specific algorithms to be used by the network manager for allocating resources. In our work we implemented the network management algorithm introduced in [3], which is one of the few network management algorithms that addresses both routing and communication scheduling. According to [3] each time that a new node joins the network, the network management algorithm is executed and tries to find the new *Uplink graph*, *Broadcast graph*, *Downlink graph*, as well as defining communication schedules for the new device. This process is done incrementally until all the nodes have joined the network.

This section considers the implementation of the network management algorithm, by discussing the four most important parts: the joining procedure, graph and route definition, communication scheduling, and finally, service request procedure.

4.2.1. Joining procedure

In this part we provide a brief description of how a new device can join the network, and receive the activation command in this implementation. We implemented the joining procedure based on WirelessHART standard.

In Figure 4 the joining sequence of a new device is shown. The new device which intends to join the network listens on a physical channel for a period of

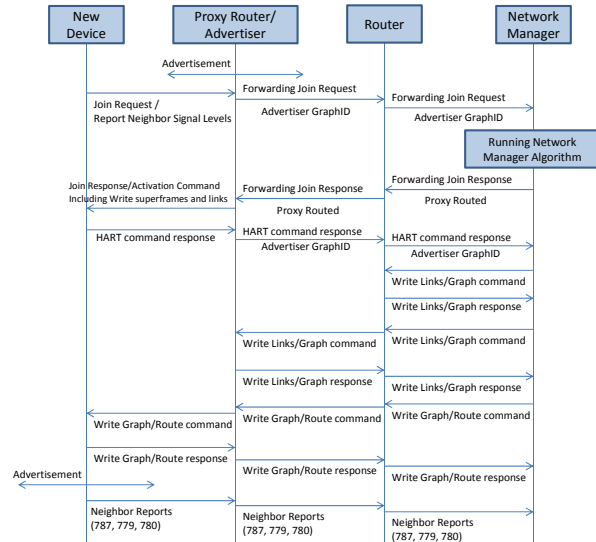


Figure 4 Joining Process

time and then continue this procedure on the next channel, until all the channels have been scanned. The new device selects the best advertiser/candidate according to certain predefined criteria and it sends the join request to the selected advertiser. The join request contains *Report Neighbour Signal Levels* (command 787) as well as other information. In this step the new device includes the advertiser Graph ID in the network header. The join request is forwarded toward the Gateway/ network manager, and the network manager who has received the join request uses its centralized algorithm to allocate the network resources (such as graphs and links). It then sends a join response/activation command to the device after all necessary network resources have been configured and reserved along the path.

After sending the join response, writing the superframe and links are the first commands to be sent to the new device. They are the only commands that can be proxy routed, in addition to the join response.

4.2.2. Graph and routes definition in the network

Three types of routing graphs are defined in a WirelessHART network to address different communication requirements.

Uplink graph is a graph connecting all devices to the gateway. This graph is used to forward both the devices' management data and process data to the gateway.

Broadcast graph connects the gateway to all devices. This graph can be used to broadcast either common data or control data to the entire network.

Downlink graph is defined per device. This graph is used to forward unicast messages from the gateway to each individual device.

To construct the reliable *Broadcast graph*, *Uplink graph*, and *Downlink graph*, the Algorithms 1, 2, 3, and 4 introduced in [3] are implemented in the network manager.

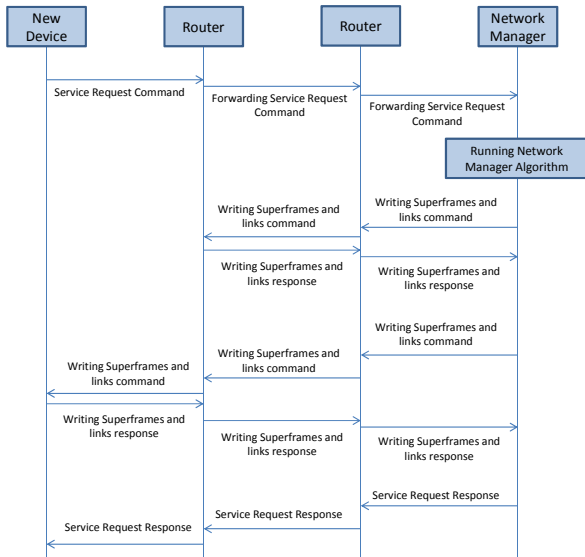


Figure 5 Service Request Process

4.2.3. Communication scheduling and channel management

After constructing the *Uplink graph*, *Broadcast graph*, and *Downlink graph* the Algorithms 7 and 8 in [3] define how the data communication schedules should be constructed and how links and superframes should be defined. By running these two algorithms in the network manager, new superframes and links are defined.

4.2.4. Service request procedure

We implemented the service request procedure based on WirelessHART standard. If a device needs to have a connection with the other devices, which could be an actuator, the requester sends out *service request* (command 799) to the network manager to request additional bandwidth. In that case the device may receive the response with some delay, because the network manager needs time to add links to a new route or an existing route, and then the network manager replies to the requester. The process of asking for more bandwidth is shown in Figure 5. The network manager allocates sufficient bandwidth along the uplink graph from the sensor to the gateway and then from the gateway along the downlink graph to the actuator. In Figure 6 a sample connection is shown in which the network manager has allocated the resources from the sensor node (37) to the actuator node (45).

4.3. WirelessHART Tcl interface commands

The WirelessHART simulation scenarios are implemented using Tcl scripts that comprise commands and parameters for simulator initialization, node creation and configuration such as *startWHGateway*, *startWHAccessPoint*, *startWHDevice*, or *requestService* commands. These scripts are similar to the existing scripts for WPAN module in NS-2.

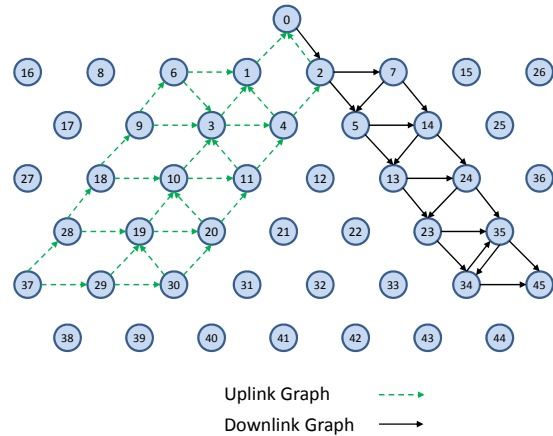


Figure 6 Connection establishment between nodes 37 and 45

The above-mentioned commands can be used respectively to start a Gateway/Network Manager, Access points, Field devices, and to request more bandwidth to communicate with the other devices. In the implementation we assumed that the connection between Access points and the Gateway is wireless.

5. Usage of this implementation

This section lists some example usages of this implementation. The first section explains the usage of the simulator for network (or control) designers and the rest discuss about its usage for protocol designers.

5.1. Feasibility study of application scenarios

Our implementation can be used to check the feasibility of applying different application scenarios with different requirements in the WirelessHART network. By having predefined application scenarios, in which the number and position of nodes, and the sampling rates of sensor nodes are defined, we can study whether the network manager can allocate sufficient resources/bandwidth to those requests or not. If the network manager cannot allocate resources, the scenario is not feasible.

The network designers can also study the network coverage and connectivity. For example, by considering the network topology in the network manager, additional routers might be deployed in the network if the sensor and actuators are not covered or if the constructed graphs are not reliable. The control engineers can also study the possible data delivery delay in the controllers.

5.2. Simulating disturbance and changes

By using the Tcl scripts such as *WHnode-down*, *WHlink-down*, *WHnode-up*, or *WHlink-up* in this implementation, one can simulate node failure, disturbances or changes within the network and thereby evaluate the ability of WirelessHART protocol and network management algorithm to cope with the disturbance in the network.

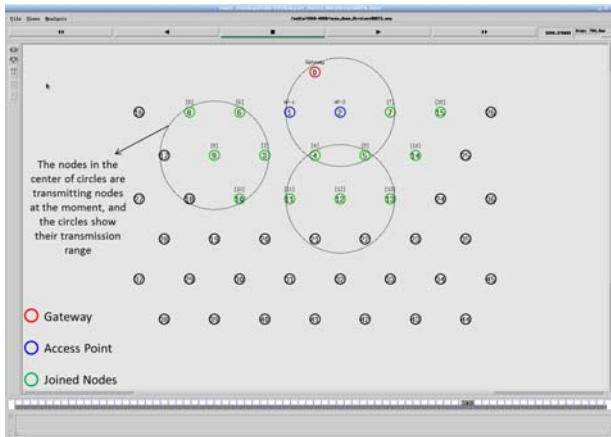


Figure 7 A sample of network topology from animation tool of NS-2 simulator (*nam*)

5.3. Evaluating other wireless protocols

This implementation can be used to compare the performance of other wireless protocols with that of WirelessHART. For instance, we are currently assessing whether a distributed management approach or the WirelessHART protocol can best cope with highly dynamic situations in a timely manner [12]. To do the comparison, we simulated node or link failures within the WirelessHART network and measured the time that the network manager needs to cope with that situation. Following this, we measured how long it takes the network to report the failure to the network manager. Subsequently, the time the network manager needs to schedule new communications and establish new routes was measured. Finally, the amount of time the network manager requires to release the previous resources and routes and define the new ones was calculated.

5.4. Modifying the WirelessHART stack

By modifying the open source WirelessHART stack of this implementation, different mechanisms in different parts of the WirelessHART stack can be applied. For instance, WirelessHART uses blind channel hopping and global blacklisting techniques to mitigate external interferences and multipath fading. Through applying other techniques, e.g. local blacklisting, we can evaluate the performance of this new method.

5.5. Evaluating different network management algorithms

WirelessHART standard does not specify the network management algorithm to be used by the network manager for allocating resources. This implementation can be used to evaluate the performance of all sorts of network management algorithms used for communication scheduling and managing routes. This means that, by applying any management algorithms instead of the management algorithm used in this implementation, and by using the implemented network

stack, the performance of these other network management algorithms can be evaluated.

6. Result demonstration

This section demonstrates our implementation results for a sample WirelessHART topology and scenario. The first part describes the simulation model, network topology and the related parameters. The nodes' joining delay and the number of required communications (number of messages sent) for these nodes are discussed in the second part. The connection establishment between different pairs of sensors and actuators are analysed in the third part. The fourth part evaluates the number of required communications for data delivery from sensors to actuators, and in the fifth part the behaviour of the system in case of nodes/links failure is discussed.

6.1. Simulation model, parameters and network topology

In this demonstration, the simulation area is 150m×150m, the neighbours distance is around 10 meters and the transmission range is 15 meters. The network consists of one gateway, two access points, and 43 field devices. In Figure 7 we take a snapshot from *nam*, which is a Tcl/Tk-based animation tool for viewing network simulation traces in NS-2, in order to show the simulated network topology.

In this demonstration we assume the length of management superframe to be 2sec. All the results reflect the average values achieved after running the program twenty times.

6.2. Node Joining

This part evaluates the nodes' joining delay and the number of required communication for joining. As shown in Figures 8 and 9, we have categorized the nodes based on their hop distance from the gateway to six groups in our demonstration. We measured the delay and the number of required communications as caused by the way in which the new device joins the network.

Figures 8 and 9 show respectively the number of required communications and the delay for nodes joining. In doing so they show both the results based on using the aggregation command² (78) and results obtained without aggregation command.

The figures also show that as the node distances increase, more delay and communications are needed for joining. Furthermore, it is noticeable that using the HART aggregation commands leads to a decrease in node joining delay as well as the required communications.

² Aggregation command allows transmission of multiple commands to one field device within one transaction.

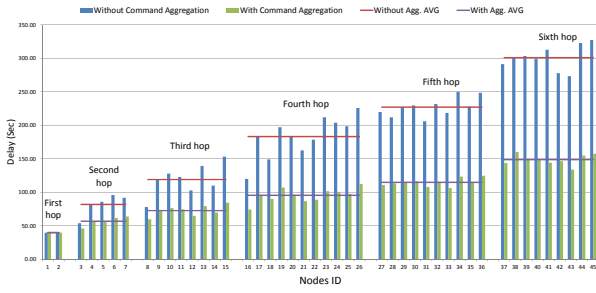


Figure 8 Nodes joining delays

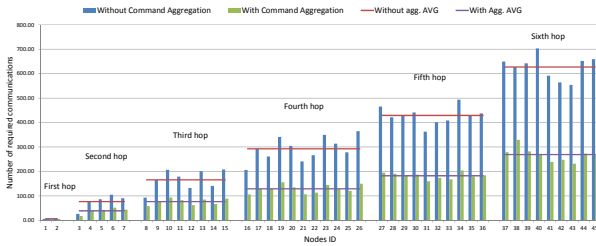


Figure 9 Number of required communications for nodes joining

6.3. Connection establishment between sensors and actuators

We defined 29 pairs of sensors and actuators. These pairs are chosen in such a way that the total hop distance of sensor to the gateway and of gateway to actuator are spread in different hop levels. Each sensor sends out a *service request* (command 799) to the network manager including the actuator address, service/connection ID, and the period of communication. By receiving the *service request*, the network manager allocates the requested resources along the uplink graph from the sensor to the gateway and from the gateway, along the downlink graph, to the actuator. In Figures 10 and 11 the delay and number of required communications for each pair based on its unique service/connection ID is displayed. In these figures we classified the connections based on the total hop distance of sensor to the gateway and the gateway to actuator. In short, as the total hop distance of a pair increases, the delay and number of required communications for establishing the connection by the network manager also increase.

6.4. Data delivery from sensor to actuator

In this part we assess the data delivery latency based on the number of required communications from first the sensor to the gateway and then toward the actuator. In Figure 12, the required number of communications is shown for 29 pairs. As in previous graphs we group the connections based on the total hop distance from sensor to gateway and then toward the actuator.

We measured the maximum, average, and minimum number of required communications for delivering the data for each pair. The difference between the maximum and minimum number of required communications was

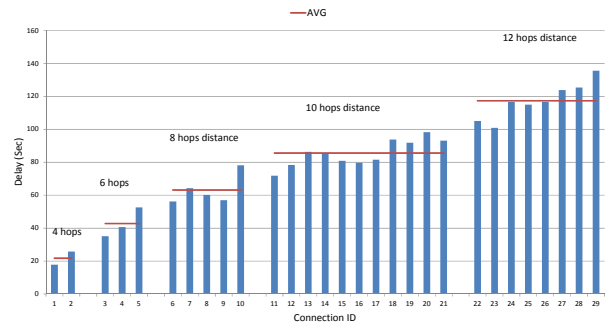


Figure 10 Connection establishment delays

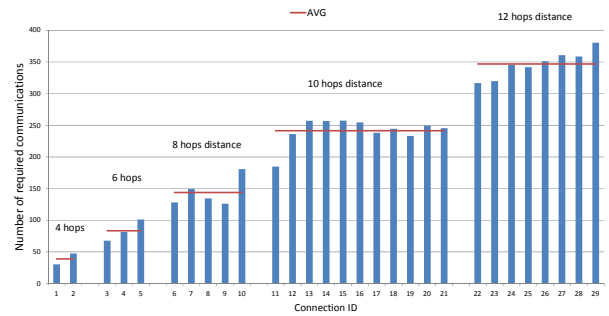


Figure 11 Number of required communications for connection establishment

caused by selecting the next hop in each node to forward the traffic toward the destination. This jitter can be significant for real-time applications.

6.5. Node and link failure in the network

In this part, we demonstrate the behavior of the system in case of link and node failure. Figure 6 showed a sample downlink graph toward node 45. Figure 13 shows the node 24 failure in the network as well as how the system manager copes with this node failure by defining new links and deleting unnecessary links.

Figures 14 and 15 show how the system behaves in the case of differing numbers of link failures, which are chosen randomly on different hop levels. We increased the number of link failures from 1 to 10 and measured the delay in, and the number of required communications for, coping with the links failure. Even though the network may still work when the graphs are unreliable, the implemented system management algorithm is set to establish a reliable graph and construct a new schedule. This causes the relatively high delay and number of required communications.

7. Future work and conclusion

As the next step we intend to validate the correctness of this implementation. One possibility is to use the WirelessHART evaluation kit to assess the correctness of the implementation. In order to assess the accuracy of this implementation three steps need to be taken: defining the same scenario for the evaluation kit and the simulator, verifying the correctness of the data and

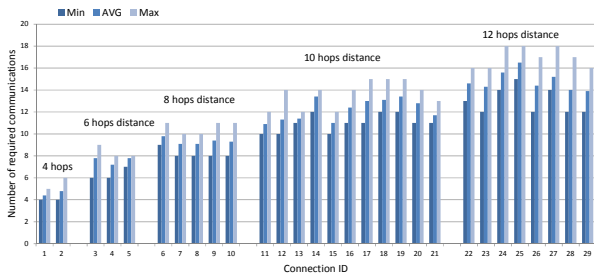


Figure 12 Number of required communications for data delivery

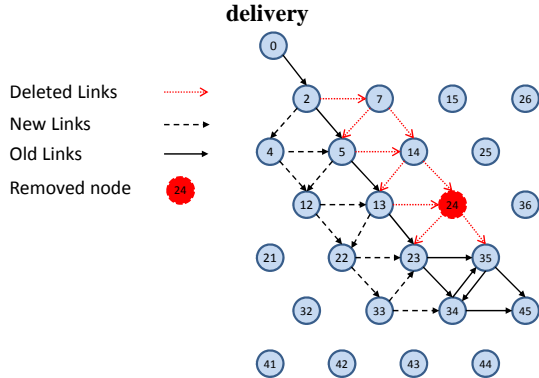


Figure 13 Node failure sample

timing compliance of the response packet, and, finally, performing the basic functionality test.

We found that the network management algorithm greatly affects the performance of the WirelessHART network, during node joining, the connection establishment, data delivery latency, and when coping with node/link failure. Consequently, when applying other system management algorithms result may differ.

This paper has shown an implementation of the WirelessHART standard and how this implementation can be used as a reference point to evaluate other wireless protocols, and to improve the WirelessHART stack and network management algorithm. In addition, it was discussed how this implementation can be used to study the feasibility of applying different application scenarios with different requirements in the WirelessHART network. This implementation can be used for research purposes, and will become available as an open source implementation in the near future.

References

[1] Industrial communication networks - Fieldbus specifications, WirelessHART communication network and communication profile IEC/PAS 62591 Ed. 1.0, Publication date: January 22, 2009.

[2] K. S. J. Pister and L. Doherty, TSMP: Time Synchronized Mesh Protocol, Proceedings of the IASTED International Symposium on Distributed Sensor Networks (DSN08), Orlando, Florida, USA, November 2008.

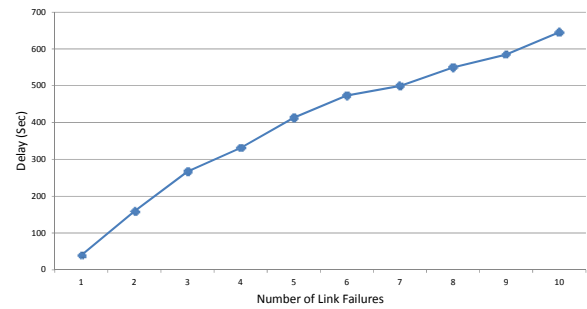


Figure 14 Network maintenance delays

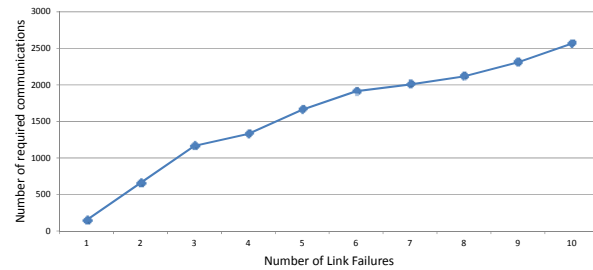


Figure 15 Number of required communications for network maintenance

[3] S. Han, X. Zhu, D. Chen, A. K. Mok, M. Nixon, "Reliable and Real-time Communication in Industrial Wireless Mesh Networks," RTAS 2011, 3-12, Chicago, IL, 2011.

[4] C. M. De Dominicis et al. Investigating WirelessHART coexistence issues through a specifically designed simulator. The Intl. Instrumentation and Measurement Technology Conference, 2009.

[5] M. De Biasi, C. Snickars, K. Landernas, A. Isaksson, "Simulation of Process Control with WirelessHART Networks Subject to Clock Drift," The 32nd IEEE Intl. Computer Software and Applications Conference, 2008

[6] K. Shah, T. Seceleanu, M. Gidlund, "Design and implementation of a WirelessHART simulator for process control," SIES 2010, pp. 221 - 224, 7-9 July 2010

[7] The network simulator ns-2. <http://www.isi.edu/nsnam/ns/>.

[8] D. Chen, M. Nixon, A. Mok, "WirelessHART: Real-Time Mesh Network for Industrial Automation" Springer, 2010, ISBN: 1441960465

[9] M. Nobre, I. Silva, L.A. Guedes, P. Portugal, "Towards a WirelessHART module for the ns-3 simulator," ETFA 2010, pp. 1 - 4, 13-16 Sept. 2010

[10] OMNeT++. <http://www.omnetoo.org/>

[11] R. Zurawski, Ed., "Network Embedded Systems". Boca Raton, FL: CRC, 2009.

[12] P. Zand, S. Chatterjea, J. Ketema, and P. Havinga, "A Distributed Scheduling Algorithm for Real-time (D-SAR) Industrial Wireless Sensor and Actuator Networks". ETFA 2012, 17-21 Sept. 2012.

[13] S. Petersen and S. Carlsen, "Performance Evaluation of WirelessHART for Factory Automation". ETFA 2009, pp. 1-9, 22-25 Sept. 2009.