Exploiting Publish/Subscribe Communication in Wireless Mesh Networks for Industrial Scenarios

André Herms, Michael Schulze, Jörg Kaiser, Edgar Nett
Department of Distributed Systems
University of Magdeburg Germany
{*aherms, mschulze, kaiser, nett*}@*ivs.cs.uni-magdeburg.de*

Abstract

This paper addresses questions of using wireless mesh networks (WMNs) in heterogeneous industrial infrastructures. This implies several problems like global addressing, handling of QoS requirements, interconnection with embedded networks. Our proposed solution is a publish/subscribe middleware. We discuss how it solves the mentioned problems. Real-world measurements and simulation results give an idea of QoS properties. A case study of controlling a mobile robot is presented. The results show that the middleware is well suited for non-critical control and monitoring tasks.

1. Introduction

Industrial installations with a large number of different plants are often spread across a wide area. Information exchange between plants as well as administration tasks like monitoring or configuring require a communication infrastructure. Traditionally, a wired backbone network is used. However, cabling involves high costs for dedicated cables and connectors, installation and maintenance. Changes in the production process can result in changes of the whole system, including the network. Mobile systems like autonomous transport robots cannot be included in a wired infrastructure.

An alternative communication infrastructure is provided by *Wireless Mesh Networks* (WMN). In contrast to the well known wireless communication with WLAN (IEEE 802.11), which consists of wireless access points (APs) connected to a wired backbone network, the communication is completely wireless. Stations automatically establish connections to all neighbours in communication range, which results in a meshed topology. Packets are forwarded between devices and thus communication can span areas much larger than the single wireless communication range.

Similar to cabling, there are also costs for buying and installing the needed equipment. Nevertheless, replacing a wireless mesh node in case of defect is much easier, quicker and mostly cheaper than to exchange a cable. Moreover, WMNs organise and optimise themselves, thus new devices are integrated automatically and only a minimal administration effort is needed. Additionally, the most important benefit of using WMNs is that the communication service is available spontaneously and everywhere in the covered area. In contrast to the standard infrastructure mode, the meshed topology contains in alternative routes, which increases the availability by redundancy. If one link breaks an alternative route is automatically chosen. This significantly improves the flexibility and allows integration of mobile systems.

Mobile systems in industrial applications have to exchange information with their environment via wireless technology. Such information are mainly sensory data or control data. However, the communication endpoints are embedded devices connected to industrial networks like CAN bus, ProfiNet, etc. The integration of WMNs leads to the following questions:

How can a device on an sub-network be globally addressed? How can we address dynamically appearing communication endpoints? How can we efficiently transmit data to devices via WMNs? How is the gateway functionality between networks provided? What kind of QoS can be provided by WMNs and how can it be improved?

We address these questions in this paper. A middleware is presented that allows network communication in heterogeneous industrial networks. We propose a subject-based publish/subscribe communication scheme for unifying the application-level interfaces. In this paper we focus on the details of implementing publish/subscribe communication in WMNs. Measurements and a case study of controlling a mobile system are presented and show the suitability of our approach.

The rest of the paper is structured as follows. In the following Sections 2 to 5 we give answers to the questions mentioned above. In Section 2 we explain the idea of using publish/subscribe. The following Section describes the integration of WMNs. Section 4 discusses QoS issues and section 5 how gateway functionality is provided. An evaluation based on real measurements and simulation is presented in Section 6. A case study of controlling a mobile robot is described in Section 9. The paper ends with related work and a conclusion with outlook.

2. Global addressing in heterogeneous industrial networks

Communication in industrial networks is not based on the classical client/server concept. Instead, we have mainly control loops. Sensors gather data and transmit them to the controller periodically. Controllers consume these and generate values for an actuator. Finally, actuators receive them and react accordingly.

If we want to exchange data with these embedded endpoints, e.g. for supervising sensory data, we require some kind of global addressing. One possibility is a global unique address for each device in the whole distributed system. Each communication endpoint has to know the address of its peer and a global hierarchical routing must be used, like done in IP networks. The drawbacks are that only point-to-point communication is possible, that routing information must be maintained, that global addresses must be assigned according to the network hierarchy, and that fixed peer addresses are difficult to handle in dynamic environments.

A better alternative is the publish/subscribe (P/S) communication paradigm, where information instead of devices are addressed. The exchanged information are termed events, which contain a unique *subject* that describes the type of data globally. A *publisher* produces data that are published under the appropriate subject and a *subscriber* subscribes subjects of interest. Publishers and subscribers only have to agree on the used subject to exchange data.

The P/S communication is applied to every network, embedded networks like Field Bus as well as a the WMN. A lightweight middleware is used to provide a common P/S interface to the application and to map the P/S communication to the underlying specific network (see Figure 1). The middleware takes care of delivering events of all publishers to all corresponding subscribers. The used network, the source of events, and the eventually required routing are transparent for the application. Interactions are spontaneous, autonomous and many-to-many. The communication is decoupled and no control

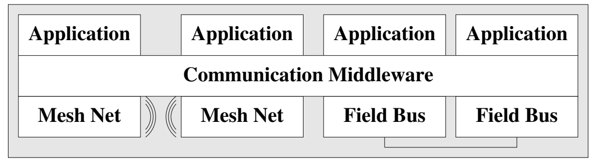

Figure 1. Wireless Mesh

flow dependencies arise [1, 2].

3. Mapping publish/subscribe communication to WMNs

The communication middleware provides a mapping component for every supported network that binds the P/S communication appropriately to the specific network addressing mechanism. In this paper we focus on the integration of WMNs. Hence, we describe in this Section the mesh routing mapper component (MRM).

In general, the mesh routing provides two kinds of communication services: *unicast transmission* and *flooding*. The unicast service directly routes packets from one station to another one on multiple hops. Flooding is a broadcast mechanism for the whole network. The source station broadcasts the packet. All stations that receive the packet the first time rebroadcast it exactly once. Thus, all stations receive the data and every station will transmit the packet. In a network with n stations this results in n transmissions. The flooding range can be limited by setting a *maximum hop count value* (MHC) for a packet. This value is decreased with every retransmission and packets are not forwarded when its reaches zero.

3.1. Maintenance of Subscriptions

For an efficient communication, the MRM of a publishing node has to know all of its subscribers. To get the subscription information, the MRM on a subscribers node sends their subscription information (all subscribed subjects) via the subscription management component (SMC) periodically. The SMC propagates the subscription information under a specific subject by flooding it in the network. Hence, every station receives and stores this information. Further, the sent event contains its initial MHC value. By subtracting the value of the MHC field at the receiver, every station can estimate the hop distance to the initiator. This is stored together with a time-stamp and the list of subjects. A lease mechanisms is used to remove old subscriber information, e.g. when a subscriber has crashed or has been removed from the network. A typical value for the update period is 60 seconds, but this parameter can be

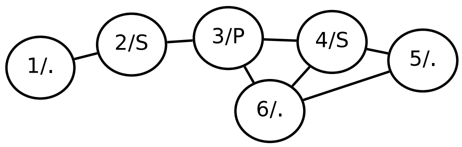

Figure 2. Example for reducing the flooding range

further adjusted to individual needs.

3.2. Efficient Propagation of Events

We investigated two approaches for optimizing the costs of P/S communication in WMNs. As a first approach, we use an efficient way of propagating events in the MRM. Bandwidth can be saved if none or only a few subscribers exist for an event. In general, events are flooded in the network and filtered at the receiver. It should be noted that flooding is very expensive, especially for the high-frequent event packets. A publisher is able to minimise the costs by looking at its list of subscribers. If there are no subscribers for a subject, the events will not be published. If there are only a few subscribers (less than a parameter k), the events are sent directly via unicast. Their addresses are known from the subscription packets.

3.3. Reducing the Flooding Range

The flooding as described in Section 3 is a very expensive operation in the network. As a second approach, we investigated ways to reduce the flooding costs. Very often, the subscribers are located topologically close to their publisher. A flooding through the whole network is not required in these cases. Based on the distance estimation from the subscriber messages (Section 3.1) we can determine the number of hops to the most distant subscriber. This value can be used as maximum hop count to limit the range and the number of transmissions.

An example is depicted in Figure 2. Here, the publishing station 3 must deliver events to its two subscribers, station 2 and 4. A flooding in this network would result in six broadcast transmissions, because every station repeats the packet once. A unicast transmission to every subscriber would require two transmissions. Reducing the flooding range to one hop results in only a single broadcast transmission.

We have implemented and tested this approach. While succeeding in simple simulation scenarios, it does not work well in real-world setups. The reason is the idealised model of wireless links. It often happens that a broadcast packet is received by a station out of communication range. This allows flooding packets to take a *shortcut*. Hence, the distance estimation returns too small distances. This leads to the flooding process being stopped too early, when the outermost stations have not received the packets. Only in rare cases when the event packets also take a shortcut, they are received. As one essential requirement of the communication service is the reliable delivery of events, we rejected this mechanism.

From these negative results we conclude: Firstly, experimental evaluation is necessary to get insights in the real problems of an approach. Simulation studies are helpful, but should not be used as the only method of evaluation. Secondly, we can see how using topological properties for optimising the flooding process does not give reliable results. Similar problems are known from the MPR optimization used by the OLSR routing protocol [3].

4. Handling of QoS Requirements

Application in industrial scenarios typically have QoS requirements. Mostly, periodic data streams with fixed small payload are used. Relevant QoS properties are realtime capability and reliability, which manifest themselves in terms of end-to-end packet loss and end-to-end packet delay.

Mesh routing protocols normally provide no QoS guarantees. Packets can be lost due to the unreliable physical layer. If there is to much load in the network, packet delays increase significantly because of the long waiting time in the interface queues. Hard QoS requirements of applications cannot be met with this technology. However, we have addressed these problems and improved the QoS properties of mesh routing.

End-to-end packet loss is the accumulated packet loss of all links on a path. Especially in industrial environments links are subject to disturbing influences like multi-path propagation and signal fading. Opposite to other protocols we do not use every link, but only the ones suitable for reliable communication. A strict link selection identifies weak and asymmetric connections that are not used. Resulting end-to-end packet loss is usually very low. Details are discussed in [4].

Hard realtime requirements are difficult to fulfil with standard WLAN network interfaces. Randomized medium access and automatic retransmissions on packet loss lead to unpredictability of packet propagation times. However, soft realtime propagation with low jitter and low average delays are achievable.

The most critical reason for packet delays are congestion effects in the network. Too much load causes packets to be delayed in the interface queues. Traditional congestion control mechanisms like the one of

TCP/IP are not suitable. These require the adaptation of the transmission rate which is not applicable to our fixed-periodic data.

Instead we use a reservation of medium time for end-to-end connections. Publishing applications specify there requirements of network resources in terms of period and packet size. A reservation protocol either ensures that enough network capacity for a newly established data stream is available, or the reservation is rejected. A protocol reserves the resources for every link on the path. This can be done by using decentralised mechanism, as we describe in [5]. However, for this architecture we use a centralised bandwidth manager. It maintains the reservations of all established data streams and allows reserving new ones. Details are discussed in [6].

5. Integration of heterogeneous networks

In Section 2 we describe the global addressing across heterogeneous networks. To enable communication spanning different networks, we added gateway functionality on nodes that are connected to more than one physical network. To applications, gateways are transparent due to the decoupled subject-based communication.

The main tasks of a gateway are forwarding, adaptation of attributes and filtering. As mention before, the subject of an event has a meaning across all connected networks. To get the knowledge about subscribed subjects, gateways use the SMC (see Section 3.1). If a new subscription is signaled via that component from subnet *A*, the gateway will create subscriptions in all other connected network segments and a publish announcement to sub-net *A*. Therefore, the gateway acts as proxy subscriber/publisher. The transformation of network-level packet formats is done automatically, as the gateway uses the application-level publish/subscribe interface to the individual networks.

Beside forwarding, the gateway adapts attributes, in particular QoS properties. It is known that different networks have different features and it is impossible to offer the same QoS everywhere. Therefore, gateways adjusts the QoS attributes of the proxies to the best level possible. However, in most cases the QoS or realtime class decreases, e.g. the hard-realtime properties of events on a Field-Bus are redefined as soft-realtime when forwarded into a WMN.

Furthermore, the gateway appends additional attributes on each forwarded event, e.g. a geographical coordinate or identifier of the part of the plant. Thereby, additional filtering and grouping of the information on application level is possible.

The third task is filtering. It is mostly done automatically at the network boundaries by the middleware, because forwarding would take place only if a subject is subscribed in another sub-net. Additionally, the gateway filters on the attributes of the events. That allows separating out events, where attributes forbid the forwarding over network boundaries or where validity of the event is expired.

To sum up, we can say that events are disseminated selectively from one network to another and thus the gateway functionality guarantees the economical utilisation of the provided bandwidth.

6. Evaluation

In this paper we present an evaluation that determines charactaristic QoS parameters of our system. We measure the end-to-end packet loss and the end-to-end packet delay depending on the number of hops. We use a simple scenario that allows to investigate the relation between these parameters. However, the topology is kept simple to be easy to reproduce.

We use nine stations in a ring topology, as depicted in Figure 3. All stations are Linux-based workstations with 2.4 GHz processors and Atheros-based WLAN interfaces. We use the madwifi driver [7] (r3350) in AdHoc mode on a free IEEE 802.11a channel. A packet filter between them prevents the establishing of unintended links. Timestamps are taken via small indication packets transmitted over a low latency GigaBit Ethernet (accuracy about 0.04 ms).

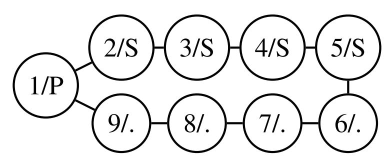

Figure 3. Topology of scenario

The publisher application is located on station 1. The subscribed stations are 2–5, which create routes with one to four hops. A 32-bit sequence number is used as payload, which allow easy detection of packet losses. We take time-stamps in the subscriber and the publisher application, which are off-line correlated afterwards. The results contain the minimum, maximum and average packet delay of 25000 events.

We consider two variants of the scenario. In the first one the four subscribers are active at the same time, which causes the broadcast publishing to be used (the parameter according to Section 3.2 is $k = 2$). In the second variant only one subscriber is active at a time and the events are delivered via unicast transmissions. This allows us to compare both publishing methods.

6.1. Measurements in a Real-World Scenario

The end-to-end packet delay is shown in Figure 4. A logarithmic y-scale has been chosen for better differentiation of minimum and average values. The average delay is in an order of 0.1–1.5 ms. Delays in our target applications should not exceed 100 ms, e.g. for realtime monitoring of systems. In average, this is fulfilled. However, we can also observe maximum values of above 100 ms. These outliers can have several reasons, the probabilistic medium access, the use of a non-realtime operation system, or some problems in the time synchronisation of the measuring stations. A look at the distribution of the outliers in Figure 5 shows that theses are very rare (again a logarithmic y-scale is used). We can conclude that hard realtime is not achievable because of the underlying WLAN standard, but soft realtime requirements can be fulfilled.

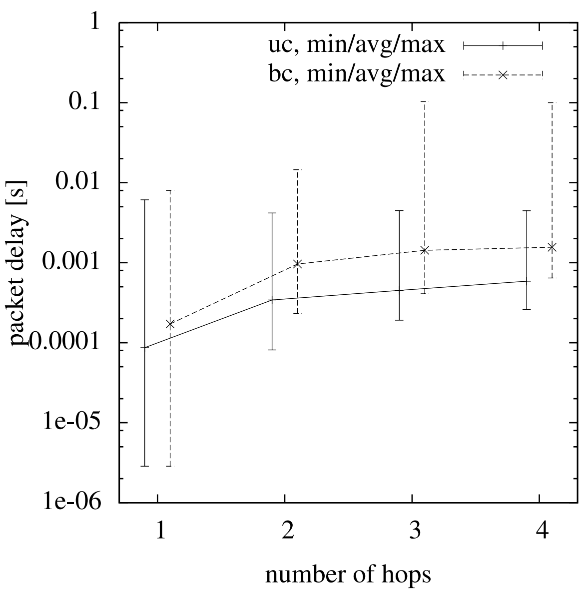

Figure 4. Measurement of packet delay, unicast (uc) and broadcast (bc)

The packet loss for communication over different numbers of hops are depicted in Table 1. The unicast

Table 1. Measurement of pack loss rates

	1 hop	*2 hops*	*3 hops*	*4 hops*
unicast	0.1%	0.2%	0.1%	0.4%
broadcast	0.4%	2.6%	2.3%	2.8%

variant shows a very low loss rate of below 0.5%. This is a very good result and is consistent with previous measurements in other more complex topologies [4]. It

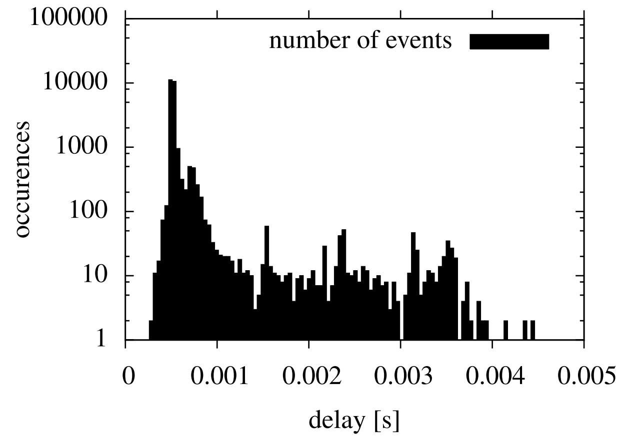

Figure 5. Histogram of packet delays, four-hop unicast scenario

is mainly caused by the very strict link selection mechanisms (see also [4]). The packet loss in the broadcast variant has a loss rate one magnitude higher. This is not a typical result. Normally, the flooding mechanism is very robust against packet loss due to the multiple reception of packet by a station. However, in the used topologies only two redundant routes to every station exist. Further, the WLAN standard uses automatic retransmissions on a single hop, but not for broadcast packets. In other topologies we observed packet losses of below 1%.

We can see that the wireless communication is not suitable for critical messages, when every packet must be received. However, for monitoring or periodic update of control settings it is acceptable. Such applications can tolerate packet loss up 10%.

6.2. Simulation Studies

Packet delays from the measurements show some serious outliers. These might be caused by the operation system, by the insufficient time synchronisation, or by the P/S middleware itself. To clarify this point we repeated the same experiment in a simulated environment. The same topology was set up and the test applications and middleware were used. Only the runtime environment was changed. This is possible due to the abstraction layer GEA[8]. We used the NS-2 simulator (version 2.32) [9]. Two configuration parameters were relevant—the log-normal shadowing model was used as propagation model and a data rate of 2 Mbit/s was configured.

End-to-end delays are depicted in Figure 6. We can see that even the maximum delays of a four-hop propagation are below 10 ms. This suggests that long delays

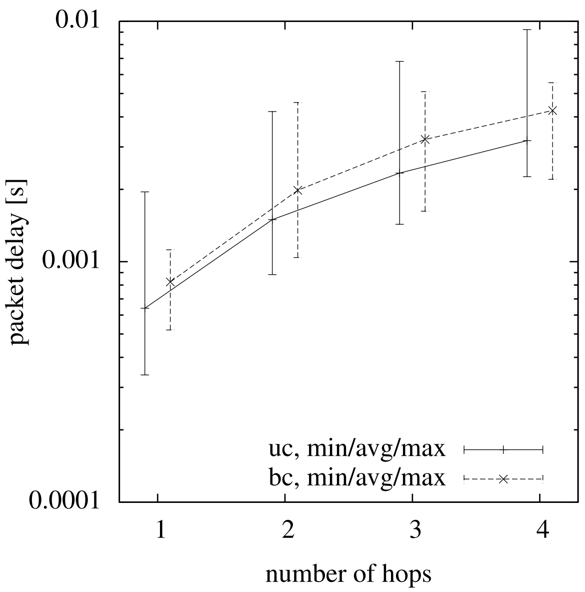

Figure 6. Simulation of packet delay, unicast (uc) and broadcast (bc)

from the measurements are caused by the runtime environment (Linux), which is not modelled in the simulator. We can conclude that the middleware allows much lower latencies than measured. However, the use of a simple general purpose operation system can add a significant amount of processing delay.

During the simulation no packet losses occurred. We could inject artificial loss in the model and would measure the same, but this would give us no further insights.

7. Overhead Considerations

Features of the P/S middleware impose costs, which depend on the underlying network. The number of packets allows a rough estimation of the communication overhead in WMNs. Announcements of subscribers use flooding, which causes n packets to be transmitted in a network of n stations. This is done for each subscriber and in each update period of about 60 seconds. The overall overhead of this is in the same order as the routing protocol and adds no significant load to the network.

The overhead of publishing events depends on the mechanisms used. The broadcast propagation is the most expensive one. The overhead has fixed costs of n packets for one event. Costs of the unicast variant depend on the distance from the publisher to the subscriber. One packet per hop is required. This must be summed up for all publisher-subscriber pairs.

A theoretical optimal event propagation could use less transmissions by using an optimal propagation tree. Events could be propagated along multicast trees from a publisher to all its subscribers. However, the problem of calculating an optimal tree (Steiner-tree) is a well known NP-complete problem, which means we must use some kind of heuristics.

Processing overhead of the mesh routing is insignificant. The routing table lookup costs $\mathcal{O}(\log n)$ for the used sorted list. The calculation of routes requires $\mathcal{O}(m + n \log n)$ (m is number of links) for the used Dijkstra algorithm. Our recent profiling results show that most CPU time is spent with processing of packets in the WLAN drivers, which is not part of our middleware.

To get an idea of the resource utilisation of our middleware on micro-controllers, we present values of the needed flash, RAM and also an estimation of the required CPU time for the AVR platform. The current realization for the AVR uses the CAN field bus for communication. The middleware implementation provides a CAN configuration protocol for automatic assignment of unique node IDs, a binding protocol for mapping the subject to a short network-specific name (event tag), a fragmentation protocol and local event propagation. The functionality of this implementation takes roughly 7 kBytes of flash and approximately 200 Bytes of RAM memory. The required CPU time for actual event propagation is in same order of systems that define the CAN identifier for the whole system in a static kind. In difference, our approach allows a dynamic configuration and dynamic binding, which takes place before events are communicated and thus it is outside of the critical communication path.

8. Related Work

Many publish/subscribe systems use TCP/IP or UDP/IP for communication. Many mesh routing protocols, like AODV [10] or OLSR [3], provide an IP routing service. By combining an IP-based mesh network with an existing P/S software, we could easily provide a publish/subscribe layer. However, only our integrated approach enables us to access internals of the routing, for controlling the flooding mechanisms (see sec. 3.3) and reserving network resources for QoS.

P/S systems like SIENA [11], READY [12], or HERMES [13] provide a scalable P/S services, based on a static broker overlay network with reliable connections. In WMNs, central brokers create traffic hot spots, which limit overall network throughput. In contrast, our approach is decentralised, which leads to a homogeneous distribution of the load.

The many-to-many communication is highly related to multicast protocols in WMNs. Several approaches [14, 15, 16] try to create multicast trees, which allow an optimal use of the network resources. These systems look very promising and a substitution of our simple broadcast with a tree-based multicast will be an objective in our future work.

Wireless Sensor Networks (MSNs) like MIRES [17] often provide the propagation from sensor nodes to a single sink. More important than the reliable propagation of all information is the energy efficiency for such networks. In industrial applications we can assume sufficient power supply, but require more reliable communication.

Zhou and Sing present a *content based multicast* in [18]. Sensors push their data in the network, where it is stored. Data sinks similar to our subscribers pull the data from locations they are interested in. Self localisation of all station is required, e.g. by using GPS. However, GPS cannot be used in indoor scenarios and other localisation systems impose additional efforts.

Mahrenholz [19] discusses a possible architecture for P/S communication in dynamic ad-hoc networks. He focuses on networks with a high rate of connectivity changes (mobility). Bandwidth reservation mechanisms are suggested, but we have shown in [5] that these suffer from unreliability even in static networks.

9. Control of Mobile Robots

As an prove-of-concept implementation we use the presented middleware for wireless control of mobile robots. We use the Volksbot hardware [20], which is a cheap rapid-prototyping platform for robot construction (see Figure 7). A Linux-driven laptop serves as main controller. Various sensors and actuators like distance sensors, cameras and motor controllers are attached. Heterogeneous embedded communication links are used (USB, RS-232, I^2C-Bus, CAN-Bus). For external communication a wireless network interface based on WLAN is included in the main controller. The communication between sensors, actuators, and controllers is based on the publish/subscribe middleware.

For wireless control of the robot we deploy a mesh network in our office environment (see Figure 8). It covers all areas where the robot moves around. Control messages are periodically published from a separate control unit and are received by the subscribed robot. Sensory data is internally published by the robot and automatically received by the subscribed control unit via WMN. It is able to visualise the distance sensors and ground sensors as well as the camera view. Hence, we are able to tele-operate the robot while directly perceiving the effects of packet delay and packet loss. Because camera data is not compressed yet, we use a low resolution to prevent excessive bandwidth requirements.

Figure 7. Volksbot platform

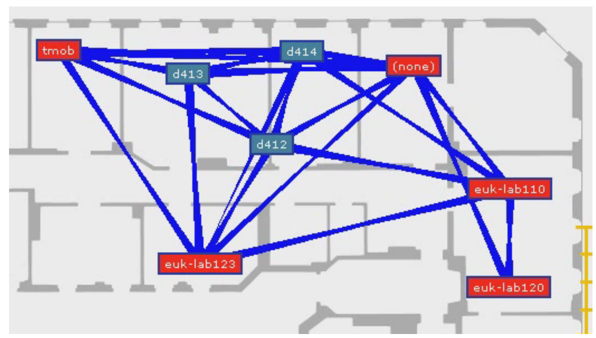

Figure 8. Live topology view of mesh network, robots are labeled d41* (blue boxes)

Still it imposes high bandwidth costs. Nevertheless, we are able to directly control the robot. The decoupled publish/subscribe communication enables us to do this without caring about the address of the robot or the bus the sensors are attached to. Even multiple robots can be included, which are distinguished by attributes of the events. Therefore, the WMN gateway appends a robot identifier attribute to published events. Control messages are distinguished by the same attribute and filtered by the subscribing robots.

This prototype shows the following benefits of using our middleware. The end-to-end delay and packet loss are low enough for control and monitoring tasks. Sensors and actuators are directly interfaced by the middleware and can be accessed remotely via wireless multi-hop communication. Further, the mesh topology ensures that during a link break still alternative routes are available. This ensures seamless availability of the communication and uninterrupted control of the robot.

10. Conclusion

The middleware presented in this paper enables communication in heterogeneous industrial communication infrastructures. With the integration of WMNs new fields of application become available. The presented solution allows interaction with embedded systems on standard industrial communication networks via cheap and flexible wireless technology. P/S communication enables dynamic and easy communication by providing a global addressing of information.

Due to the observed possible packet loss and jitter in the delays, it is not suitable for critical control tasks. Instead, monitoring, sporadic setting of system parameters and other non-critical task will be the targeted applications.

The current system has many points of possible optimisations. The propagation of events should be done via multicast trees. The impact of the runtime environment could be minimised with real-time operating systems. Further performance evaluations are required. The availability of a real implementation gives us the opportunity to detect open problems, which do not appear in artificial simulated scenarios. By further real world studies we will be able to get further insights in the performance of WMNs in industrial applications.

References

[1] A. Casimiro, J. Kaiser, and P. Verissimo, "An architectural framework and a middleware for cooperating smart components," in *ACM Computing Frontiers conference (CF'04)*, Ischia, Italy, April 2004, pp. Pages: 28 – 39.

[2] J. Kaiser, C. Brudna, C. Mitidieri, and C. Pereira, "COSMIC: A middleware for event-based interaction on CAN," in *9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2003)*, Lisbon, Portugal, September 2003.

[3] T. Clausen, P. J. (editors), C. Adjih, A. Laouiti, P. Minet, P. M. ler, A. Qayyum, and L.Viennot, "Optimized link state routing protocol (olsr)," RFC 3626, OCT 2003, network Working Group.

[4] A. Herms, G. Lukas, and S. Ivanov, "Realism in design and evaluation of wireless routing protocols," in *Proceedings of First international Workshop on Mobile Services and Personalized Environments (MSPE'06)*, vol. P-102. Aachen, Germany: Lecture Notes in Informatics (LNI), November 2006, pp. 57–70.

[5] A. Herms, S. Ivanov, and G. Lukas, "Precise admission control for bandwidth reservation in wireless mesh networks," in *Proceedings of 4th Intl. Conference on Mobile Ad-Hoc and Sensor Networks MASS'07*, Pisa, Italy, October 2007.

[6] A. Herms, G. Lukas, and S. Ivanov, "Measurement-based detection of interfering neighbors for qos in wireless mesh networks," in *16th IST Mobile and Wireless Communications Summit 2007, Proceedings of*, 2007.

[7] "MadWiFi project homepage," 2008. [Online]. Available: http://madwifi.org

[8] A. Herms and D. Mahrenholz, "Unified development and deployment of network protocols," in *Proceedings of MeshNets Workshop in conjunction with First International Conference on Wireless Internet (WICON)*, Budapest, Hungary, July 2005.

[9] "NS-2 project homepage," 2008. [Online]. Available: http://nsnam.sf.net

[10] C. E. Perkins, E. M. Belding-Royer, and I. Chakeres, *RFC3561 - Ad hoc On-Demand Distance Vector (AODV) Routing*, Request for Comment, July 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3561.txt

[11] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, no. 3, pp. 332–383, 2001.

[12] R. Gruber, B. Krishnamurthy, and E. Panagos, "The architecture of the READY event notification service," 1999. [Online]. Available: citeseer.ist.psu.edu/gruber99architecture.html

[13] P. R. Pietzuch and J. Bacon, "Hermes: A distributed event-based middleware architecture," in *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 611–618.

[14] P. M. Ruiz, F. J. Galera, C. Jelger, and T. Noel, "Efficient multicast routing in wireless mesh networks connected to internet," in *InterSense '06: Proceedings of the first international conference on Integrated internet ad hoc and sensor networks*. New York, NY, USA: ACM, 2006, p. 26.

[15] H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc networks," in *MSWIM '00: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*. New York, NY, USA: ACM, 2000, pp. 61–68.

[16] S.-J. Lee, M. Gerla, and C.-C. Chiang, "On-demand multicast routing protocol," *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, pp. 1298–1302 vol.3, 1999.

[17] E. Souto, a. Germano Guimar G. Vasconcelos, M. Vieira, N. Rosa, and C. Ferraz, "A message-oriented middleware for sensor networks," in *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*. New York, NY, USA: ACM, 2004, pp. 127–134.

[18] H. Zhou and S. Singh, "Content based multicast (cbm) in ad hoc networks," in *MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*. Piscataway, NJ, USA: IEEE Press, 2000, pp. 51–60.

[19] D. Mahrenholz, "Providing QoS for publish/subscribe communication in dynamic ad-hoc networks," Doctoral Dissertation, University of Magdeburg, May 2006.

[20] "Volksbot project hompage." [Online]. Available: http://www.volksbot.de/index-en.php