Real World Testing of Aggregation in Publish/Subscribe Systems

Michael Schiefer, Christoph Steup and Jörg Kaiser
Department of Distributed Systems, O.-v.-Guericke-Universitaet Magdeburg,
Universitaetsplatz 2, 39106 Magdeburg, Germany

Keywords: Aggregation, Wireless Sensor Network, Publish/Subscribe, Network Topology, Ubiquitous Computing.

Abstract: Wireless sensor networks collecting data to monitor real life processes gain increasing attention from the scientific community. These systems promise ubiquitous computing in a digitalized world. However many problems need to be solved to achieve this dream. One of these problems is the limited battery power of the nodes and the limited bandwidth of the communication. Existing work tackle the problem of limited bandwidth by merging communication packets within the network. Based on this approach we investigate the use of application specific aggregation in publish/subscribe wireless sensor systems. We expect this approach to overcome overload situations in the network and decrease packet loss due to bandwidth exhaustion. This paper describes our architecture and evaluates the properties of our approach. We investigated specific topologies theoretically and through real-life experiments. Our results quantify the achievable event count and loss rate reduction.

1 INTRODUCTION

Sensor networks are an emerging field of technology which provides easy and cheap surveillance of complex and spacious systems. These networks consist of cheap nodes with limited computational power and energy storage. Therefore the collaboration of these nodes is crucial to the final service.

This technology is applicable to many scenarios ranging from forest fire detection to remote surveillance of industrial environments. This paper considers a typical wireless sensor network scenario including mobile and failing nodes. Therefore the resulting network includes dynamic changes in the topology as well as unreliable links between nodes.

The dynamic nature of such a network induces challenges for the reliable dissemination of the data. The constant changes of the topology complicates the identification of nearby stations through names or addresses. A solution to this problem is the publish/subscribe communication paradigm to decouple the nodes from each other as described by (Eugster et al., 2003).

Another problem is the limited processing capability and bandwidth of the hardware. This may cause simple strategies like ”forward-to-sink and process” to fail because of overload in the sink or during communication. There are two possible solutions for this problem: re-routing to nodes with free capacity or reduction of used bandwidth. Re-routing is a complex problem due to mobility and energy constraints. It also fails if only one path to the sink exists. The reduction of bandwidth can either be achieved by dropping or combining individual packets. Dropping always induces a loss of information and needs rules to describe which packets should be dropped (cf. (Chen and Kotz, 2005)), whereas combining may conserve the data but increases the dissemination delay.

A problem specific to event driven sensor systems occurs whenever multiple sensors observe the same real event. The sensors will create individual events shortly after the observation. These sensor are generally close to each other, which creates a short-term overload in the network due to the concurrent event dissemination. Additionally necessary forwarding of the created events may heighten the problem. This can either be overcome by spreading the event publication in time, which the CSMA/CA algorithms will do or by combining multiple events to prevent the individual event transmissions. However the spreading done by CSMA/CA is error-prone since it depends on statistical properties.

This paper tries to combine the publish/subscribe communication paradigm with an application-specific aggregation mechanism to lower the used bandwidth, overcome the short-term overload through concurrent event production and provide a load balancing mechanism within the wireless sensor network. Addition-

Schiefer M., Steup C. and Kaiser J..
Real World Testing of Aggregation in Publish/Subscribe Systems.
DOI: 10.5220/0004533401990206
In *Proceedings of the 10th International Conference on Signal Processing and Multimedia Applications and 10th International Conference on Wireless Information Networks and Systems* (WINSYS-2013), pages 199-206
ISBN: 978-989-8565-74-7

ally the properties of our approach will be evaluated on hypothetical topologies as well as real life experiments.

2 RELATED WORK

Plenty of work exists on aggregation of data in networks. The first approach especially tailored towards wireless sensor networks is ”Tiny AGgregation” (TAG) (Madden et al., 2002) on TinyOS which afterwards lead to TinyDB (Madden et al., 2005)(Madden, 2003). This work views the sensor network as a database, which executes SQL like queries. A tree based routing is used to spread the request ”from a powered, storage-rich basestation” as root to every node. Afterwards the data flows back from the leaves level by level. The parent nodes first gather the data of all children and aggregate them with their own data before they transfer the result one tree level up. Thus the nodes can sleep most of the time. Additionally each node carries only one message per request, which ”dramatically decrease[s] the amount of communication required to compute an aggregate” in most cases. This also provides an almost uniform energy consumption of the nodes. TAG is limited by its centralized routing tree, which makes it impossible to use two queries at the same time and always needs an external base station as the source of the request.

Another approach targets the known meta data of publish/subscribe middlewares. Application-Specific Integrated Aggregation for Publish/Subscribe Systems (ASIA) by Margera et. al(Frischbier et al., 2012) views the network as a topic-based publish/subscribe system instead of a distributed database. The publishers and subscribers are linked with brokers thus every message has to be transmitted over at least one broker. This generates meta data on the network itself or the sensor data. ASIA provides an aspect oriented interface to determine and aggregate those meta data in a application-specific way. However it is not intended to aggregate the data itself. The current design and implementation of ASIA is unsuitable for resource constraint systems.

To use aggregation in complex systems application-specific mechanisms are needed since primitive merging functions are not enough. Systems like ASIA allow the users to create customized merging functions only for meta data. On the other hand TAG and TinyDB allow user-specified application-specific aggregation functions, but are incompatible with the publish/subscribe communication paradigm due to their rigid time and routing scheme. Therefore this paper explores semantic aggregation in publish/subscribe systems and evaluates it theoretically and practically.

3 ARCHITECTURE

Our approach is based on Contiki(the Contiki project, 2012) as the underlying operating system and FAMOUSO(Zug et al., 2010)(Schulze, 2009) as publish/subscribe middleware.

Contiki provides hardware abstractions and thread implementations tailored towards deeply embedded systems like sensor nodes. Whereas the FAMOUSO middleware provides a topic-based publish/subscribe communication independent of the underlying OS through template meta-programming.

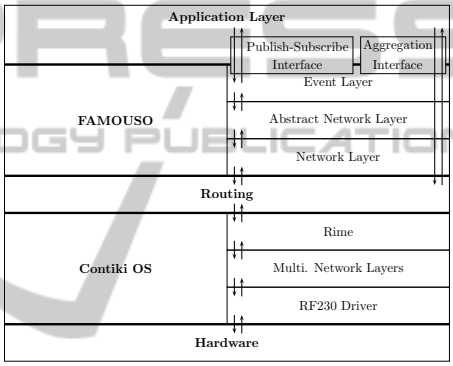


Figure 1: Architecture of the topic-based aggregation using the FAMOUSO middleware and CONTIKI.

As depicted in Figure 1 the middleware uses the Rime-Stack by (Dunkels, 2007) of Contiki to access the underlying wireless network. The Rime-Stack encapsulate the wireless hardware as well as the MAC-Layer of the IEEE 802.15.4 Standard(IEEE, 2011). It also provides a node id, which is used by the network layer of the middleware to translate publish/subscribe topics to network addresses in run-time.

Since Rime does not provide a routing mechanism FAMOUSO integrates its own, which is based on Directed Diffusion by (Intanagonwiwat et al., 2000). The subscribers periodically send their subscription through the network by flooding it. Every node saves the first neighbour sending this message, whichever is closer to the subscriber. Thus every node knows where received events need to be forwarded to. The interval of renewal of subscriptions is chosen depending on the mobility of the nodes. This provides an appropriate routing mechanism for content-based communication, which fits to the communication model of the middleware. Through periodic renewals of subscriptions limited mobility can be tolerated.

4 TOPIC BASED AGGREGATION

There are mainly two ways to achieve a semantic, application-specific aggregation in publish/subscribe systems.

The first one exploits the existing publish/subscribe API. One node subscribes to a topic of specific data, aggregates the data and publishes the results in another topic. All interested nodes subscribe for the new topic containing the aggregated data.

The second approach uses an application specified aggregation function and links it to the routing via a defined interface. If an aggregatable event arrives the routing forwards it up to the specified application function so that it may be merged with other events. Thus the events are aggregated on the fly on their way from publisher to subscriber.

The first approach has the drawback that the application-layer of the publish/subscribe system has no information on the topology of the network. Therefore explicitly placed aggregation nodes are needed. In case of a static network the optimal positions may be pre calculated. For dynamic networks however it is needed to reposition these nodes dynamically. The topology data for this repositioning needs to be deduced through additional mechanisms like service discovery. The second approach circumvent this problem and is used in our approach.

We force the application designer to convert the event data in an aggregation aware format before publishing it. This enables the subscriber to evaluate the received aggregated events before delivering them to the application. This induces no additional overhead to the designer since publishing and subscribing needs application-specific data anyway. Therefore we can assume that FAMOUSO either transfers data in an aggregation aware format or the data itself is marked as not aggregatable.

The application registers its specific merge function with the middleware as a channel attribute. The middleware passes this merge function down to the routing layer. Afterwards every received, aggregatable packet of the appropriate topic is passed to the merge function. To link the application event interface and the routing layer packet interface every packet is converted to an event. To support storing and merging of events each merge function owns an individual packet buffer. To avoid long hold times every data in the buffer is send after a specified period of time, which is changeable by the merging function. It is although possible to hold up events forever by periodically changing the time-out.

The resulting architecture can even handle partially deployed merge function. This might occur if nodes cannot aggregate data because of limited memory or computing time. This paves the way for a dynamic distribution of aggregation nodes within the network based on individual capabilities and the current topology of the network.

On the other hand the advantages and disadvantages heavily depend on the used routing function. An aggregation of data before the subscriber cannot be guaranteed, since parallel data flows like depicted in dark blue in Figure 2 are not aggregatable. A more aggregation aware routing tries to collapse the different event paths sooner and may be able to increase the benefit of the aggregation.

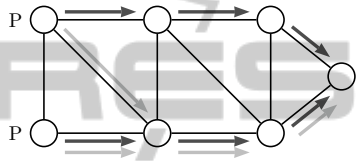


Figure 2: Example network with parallel flow (dark arrows) and an aggregation optimized flow (lighter arrows).

The routing tries to delay the cloning of events as long as possible to minimize the amount of packets to be transmitted to deliver a published event to the subscribers. Therefore a route must be found that enables early aggregation as well as late cloning.

Applications need to be aware that they may receive individual events multiple times as parts of aggregated events. We therefore force applications to handle such situation explicitly.

The publish/subscribe middleware creates individual routes for the events of each topic, as visible in Figure 3. Here two different topics exist with only one subscriber for both. At node R_5 events for channel one are routed to R_3 to enable aggregation. Events of topic two are routed through R_6 to be aggregated with events from $P_{2,2}$. Hence packets arriving at the same node with the same destination are routed differently depending on their content to enable optimal aggregation. If both channels would be routed through R_3 this would result in more sent packets in a global view. This shows that cost and benefit of aggregation and routing cannot be viewed independently. Useful estimations are only possible for specified scenarios with known routing mechanisms and topologies.

So why is the aggregation neglecting routing algorithm of FAMOUSO used? Most publish/subscribe systems do not incorporate an aggregation aware routing and thus using aggregation would lead to similar results. Furthermore by designing the network topology it is possible to create worst case but also best

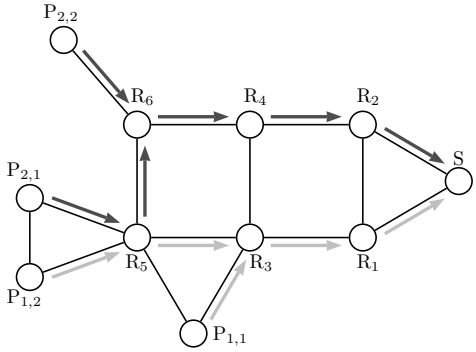


Figure 3: Example network with different routes based on the topics of the data. $P_{i,k}$ is the k^{th} publisher for topic i. R_j are forwarding relay nodes. The topic data is displayed through the lighter and darker arrows respectively.

case scenarios. These results can be validated with practical experiments.

5 EVALUATION

This section first analyse the possible cost and benefits of an aggregation in a theoretical way. Afterwards the real world test cases are described.

5.1 Topological Analysis

Unless otherwise stated we consider the routing as well as the links between nodes to be optimal. The goal of the analysis is an estimation of bandwidth reduction as well as induced latency for some general topologies.

We found no typical topology, which approximates WSN well enough. Therefore we will discuss some topologies with special properties and try to reason their relevance towards real networks.

5.1.1 Worst Case

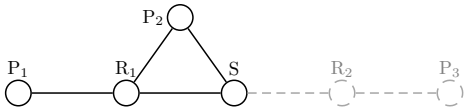


Figure 4: A topology not benefiting from aggregation. The only extension conserving the worst case property is shown in dashed grey.

To achieve worst case packet counts disjoint routing paths are needed. Worst case delays can only be observed if no parallel communication to multiple aggregating nodes are possible.

The black graph in Figure 4 shows such a topology without any possible time and event reduction. This topology enables aggregation only at node R_1, which induces an additional packet from P_2 to R_1. However the gain is only one packet less from P_2 to S. Therefore no reduction of packet count is possible. Additionally P_1 and P_2 may not transmit data to R at the same time, which lengthens the delay of the events.

This worst case scenario is extensible, Figure 4 including the grey parts. However the worst case property only holds if each publisher has its own disjoint routing path to the subscriber. In the end the topology may only be extended through single publisher chains connected directly to S.

5.1.2 Best Case

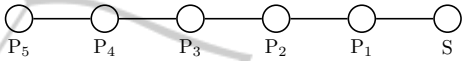


Figure 5: The best topology for aggregation.

The best case for aggregation is a chain of nodes consisting of a subscriber at one end and publishers (see Figure 5). Under the assumption that all *n* publishers publish at nearly the same time a minimal delay is to be expected. The rightmost publisher P_5 transmits its event to its neighbour. The neighbour instantly aggregates it with its own event and forwards it to the next one. In the end the complete aggregated event arrives at the subscriber. The packet count in this example is reduced from 15 to five. In general aggregation could decrease the number of transferred packets from $(n \cdot (n+1)/2)$ to *n*. This is the optimum concerning delay and packet count.

If on the other hand the leftmost publisher P_1 starts to transmit followed by P_2 and so on, only one event arrives at the subscriber after five sent packets. Because of the time-out the left publisher P_1 sends the saved event of P_2 to the subscriber etc. Depending on the time of publication and the chosen time-outs it is possible for every event to arrive at the subscriber individually. Thus even this best case topology may provide no benefits and introduce additional delays. This is a general problem. For every aggregation node in every topology such a bad case could be generated. It is therefore necessary to chose waiting time-outs depending on application's constraints.

5.1.3 Simple Aggregation

The simplest non linear topology enabling aggregation consists of multiple publishers and subscribers connected to one relay node (cf. Figure 6). If publishers transmit their events shortly after each other the relay node may send the aggregated event to the subscriber instantly. Thus the delay is small and the event count is minimized.

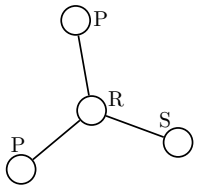


Figure 6: A topology with the possibility of aggregation.

On the other hand if the transmissions of the publishers are widely spread in time the relay node will withhold events until timed-out. As described in Section 5.1.2 this is a general limitation. The simplicity as well as the extensibility of this topology are indications for its occurrence in real networks

5.1.4 Conclusion

As seen worst case scenarios are difficult to build and unlikely in reality. Topologies with expectable good aggregation results, e.g. chain like structures in mine drifts(Harms, 2011), are more likely. Regular networks will be composed of positive and negative structures like Figures 4 to 5. In summary benefits are expected with respect to routing parameters and timing configuration.

5.2 Test Cases

To validate the theoretical results of Section 5.1 real life experiments were conducted aiming at bandwidth saving.

5.2.1 Configuration

As physical motes the RCB128RFA1 and the deRFmega128-22A platforms by dresden elektronik ingenieurtechnik gmbh(Pietschmann, 2013) are used. They are quite similar in hardware with the main difference being the antenna. As software our architecture as described in Section 3 is used. Contiki is configured to use ”nullmac” as MAC-Protocol resulting in no retransmissions and ”sicslowmac” as ”Radio Duty Cycling”-Protocol underneath it. Furthermore the maximum tx power is reduced and the rx treshold is increased to achieve a smaller and more controllable test area.

Each test contains one additional control node, which is not represented in the figures. No packet is ever routed through this node. It is used to initialize and verify the topology at the beginning and controls the tests.

The topology is configured statically in the EEPROM of the node. Even though nodes may receive packets via links not part of the topology, these will be dropped before being processed. To achieve reproducible test results a verification of the routing paths between publishers and subscriber is done in the beginning of each test.

5.2.2 Aggregation Function

The used aggregation function is the mean of 16 bit values. To enable identification of individual events within the aggregated events a list of individual events is attached to the resulting value. The forwarding of stored events(aggregated or individual ones) happens in three situations.

The first one is a time-out. After five seconds the stored event is forwarded to the next node. This value is a trade-off between delay of event dissemination and time to aggregate incoming events.

The second situation consists of a maximum number of individual events incorporated within one aggregated event. The maximum number is inferred from the publishers active within the network.

The third situation occurs if an event is received from a publisher already contained in the publisher list of the stored aggregated event. This results in forwarding the stored event and using the newly received event as the new aggregated event. This mainly happens in high throughput tests when a packet is lost and the original publisher of the lost event produces the next data.

5.2.3 Bandwidth Tests

Our test considers only the best case topology for bandwidth reduction to quantify the possible benefit of using aggregation. This enables us to estimate the possible benefit for different types of real life topologies. Thus the best case topology, see Figure 5, is used.

We considers four different configurations: high and low network throughput with a size of five as well as ten publishers. To quantify the benefit of the aggregation each configuration is tested with and without aggregation. Network throughput will be emulated by the transmission of pseudo events, which contain randomly generated sensor data. This generated sensor data will be aggregated by the application specific mechanism described in Section 5.2.2 attached to the used publish/subscribe channel. The high throughput scenario publishes repeatable randomly two to four pseudo events per publisher per second, whereas the low throughput test publishes only one pseudo event per publisher per minute. The nodes are configured to publish at nearly the same time. The experiments with high network throughput will run 15 minutes and the low throughput ones 30 minutes.

The bandwidth test is used to measure transmitted as well as received packets for individual nodes and published and received events for the whole test. Each node measures the data on its routing layer. After the test ends the statistics are saved in the EEPROM of the node and later transmitted to pc via a serial data connection. Thus the data gathering is guaranteed to not interfere with the measurements. To prevent result-biasing additional packets there is no retransmission mechanism activated in the network.

To evaluate the problem of concurrent event production we additionally conducted low throughput tests with an id-based delay before the publication in each node. The results should approximate a network not observing the same phenomenon and therefore not publishing at the same time.

6 RESULTS

The following sections will discuss the results of our real-life experiments as described in section 5.2. We will use abbreviations like H10AD, which stands for a high throughput test (H) of size ten (10) with aggregation (A) and id-based delay (D). Another example would be L5, which represents a low throughput test of size 5 without aggregation and without delay.

6.1 Packet Count

The Figure 7a depicts the bandwidth results of the H10A experiment showing an increasing amount of packets related to the distance of the publishing node P_i to the subscribing node S. This is caused by the routing based retransmission of each packet of the predecessor of each node additionally to its own data. Consequently a linear growth is to be expected. However the diagram shows a non-linear growth from publisher P_{10} to P_1. This is caused by the high loss rate, which will be discuss in Section 6.4.

The nodes of the H10A experiment, as visible in Figure 7b, transmit a nearly constant amount of packets only. This is caused by the combination of the received packets to a single event. Node S transmits some packets to establish the channel and initialize the routing. Publisher P_{10} only receives these propagating through the network. Therefore both nodes have small values for transmitted and received packets respectively.

Figure 7c shows the bandwidth results of the L10AD experiment. In this experiment the publishers additionally had an ID dependent delay before publishing. Here the expected linear growth of packets based on the distance to the subscriber is visible. Compared to the basic test, this is due to the additional delay between publications. The delay is significantly larger than the initial delay of the CSMA/CA of 802.15.4. This spreads the publications in time and lessens significantly the collision rate of the packets within the network. This shows the problem of short-term overload on event generation of multiple nodes.

6.2 Aggregation Efficiency

Aggregation efficiency can be measured through the amount of individual events contained in aggregated events. In the experiment H10A 2372 aggregated events contained all possible ten individual events. Only 49 aggregated events contained less then ten events. Therefore during the experiment in 95% of all cases an optimal aggregation was possible. This is to be expected since the publishers transmitted shortly after each other.

Experiment L10AD created quite different results, as visible in Figure 7d. The x axis shows the number of individual events the received aggregated event is build of. The y axis corresponds to the occurrence of an aggregated event containing a specific number of individual events. This diagram shows that $85,5\%$ of all events were part of aggregated events, but none aggregated event contained the maximum number of ten individual events. This is caused by the additional delay destroying the alignment of the publishers. This disabled the aggregation in some nodes, due to time-outs.

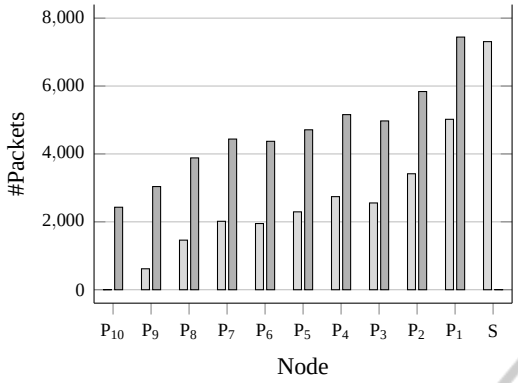
6.3 Total Event Counts

The Figures 7e and 7f illustrates the total number of transmitted events within the network.

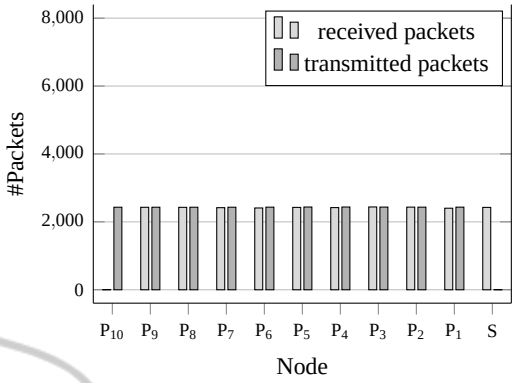
The total number is the sum of all events generated by publishers as well as all the forwarded events while routing. The numbers clearly show the correlation of the saving with the chain size. This supports the theoretical evaluation in Section 5.1. The H10 and H5 experiments showed an event reduction of $50,4\%$ and $32,1\%$.The L10 and L5 experiments showed similar values with $45,3\%$ and $34,1\%$. Therefore it can be assumed that the event reduction does not depend on the throughput of the network. This fits to the theoretical evaluation.

6.4 Loss Rate

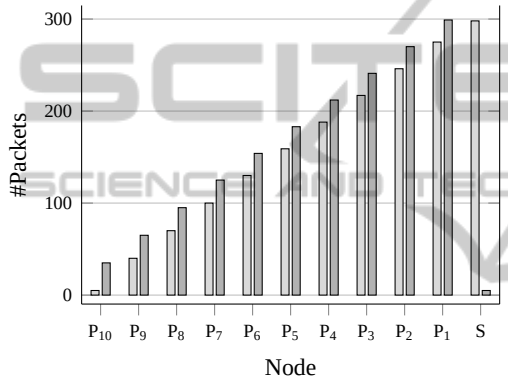
The loss rate of the application is calculated as the fraction of events received by the subscriber of all published events. Aggregated events are multiplied by the number of individual events they contain. The



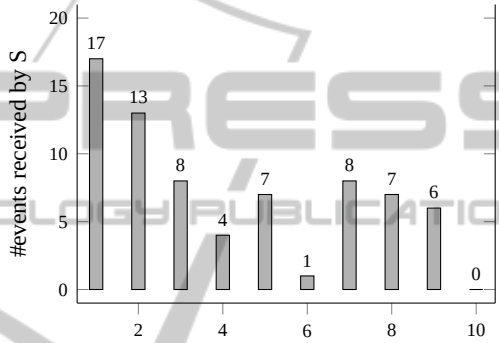
(a) Packet count of experiment H10.



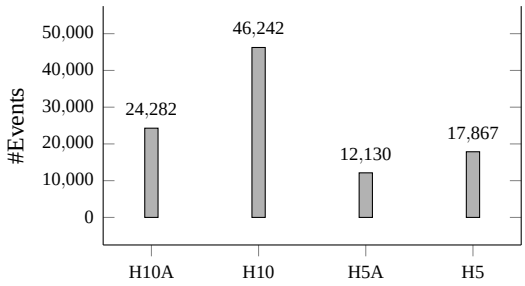
(b) Packet count of experiment H10A.



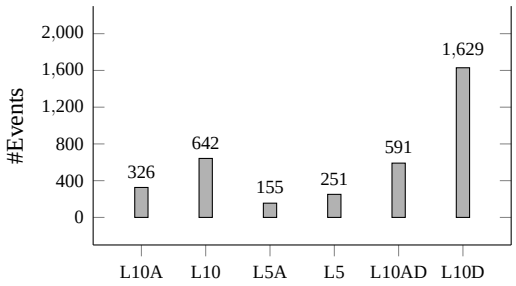
(c) Packet count of experiment L10D.



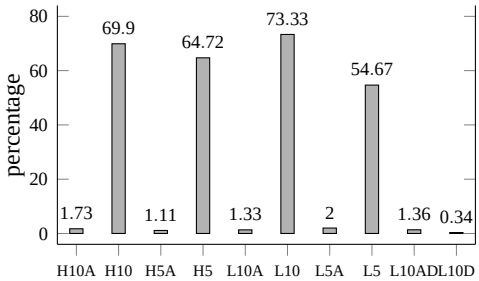
(d) Histogram of number of aggregation events received by S containing x events in experiment L10AD.



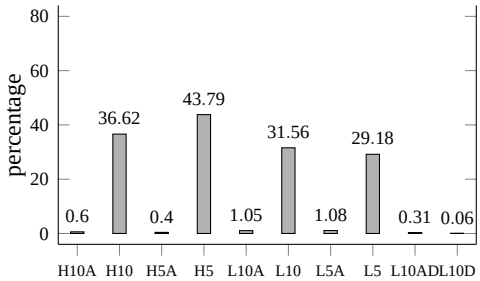
(e) Total Event Count of high throughput test cases.



(f) Total event count of low throughput test cases.



(g) Event loss rate of all experiments.



(h) Packet loss rate of all experiments.

Figure 7: Results of different test cases as described in Section 5.2.

loss rates for events and packets can be seen in Figures 7g and 7h respectively.

Tests with aggregation showed a quite low loss rate compared to tests without. A cause may be the nearly synchronous event production of the publishers. This causes a temporal overload in the network, which increases packet loss. The aggregation overcomes this problem by combining multiple packets to a single event. Therefore the short-term overload of the network is circumvented and the additional collisions are prevented.

This assumption is supported by the loss rate of experiments L10D and L10AD. In this case the loss rate of aggregation is higher, even though both experiments showed a very low loss rate for packets as well as events. The negative benefit might be caused by the additionally delays of the forwarding, which shifts forwarded events closer to the next published event increasing collision probability.

7 CONCLUSIONS

In this paper we presented an extended publish/subscribe architecture to support application specific aggregation. We described our architecture based on the publish/subscribe middleware FAMOUSO and the embedded operating system Contiki as well as the topic based aggregation.

To estimate the benefits of general aggregation mechanisms in publish/subscribe systems we evaluated different possible network topologies theoretically. Additionally we have done experiments on best case topologies to evaluate the possible benefit. This evaluation showed that in chain like topologies an event count reduction of 32% for five nodes and 50% for ten nodes is possible. This supports the theoretical evaluation, which promises an increasing benefit depending on the size of the chain. However the theoretical benefits were higher because they omitted loss rate. Additionally application-specific aggregation emerged as a solution for the temporal overload of the network during concurrent event production of close nodes. Finally we reasoned why aggregation beneficial topologies will be increasingly likely depending on the size of the WSN.

Following this work we want to evaluate the influence of an aggregation aware routing, which promises additional benefits in non-optimal topologies. Furthermore we want to extend the evaluation to other metrics like energy and resource consumption.

REFERENCES

Chen, G. and Kotz, D. (2005). Policy-driven data dissemination for context-aware applications. In *In Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, page 283–289.

Dunkels, A. (2007). Poster abstract: Rime — a lightweight layered communication stack for sensor networks.

Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A. (2003). The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131.

Frischbier, S., Margara, A., Freudenreich, T., Eugster, P., Eyers, D., and Pietzuch, P. (2012). Asia: application-specific integrated aggregation for publish/subscribe middleware. In *Proceedings of the Posters and Demo Track*, Middleware '12, pages 6:1–6:2, New York, NY, USA. ACM.

Harms, H. (2011). Sensornetzwerke im Bergwerk als Untertage-Ortungssystem. Diploma thesis, Otto-von-Guericke University Magdeburg.

IEEE (2011). IEEE standard for local and metropolitan area networks–Part 15.4: Low-rate wireless personal area networks (LR-WPANs). Technical Report 802.15.4-2011, IEEE Computer Society.

Intanagonwiwat, C., Govindan, R., and Estrin, D. (2000). Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *MOBICOM*, pages 56–67. ACM.

Madden, S. (2003). Tinydb: A declarative database for sensor networks. website http://telegraph.cs.berkeley.edu/tinydb/index.htm, date: 2012-07-25.

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2002). Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146.

Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2005). Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173.

Pietschmann, L. (2013). dresden elektronik webseite. http://www.dresden-elektronik.de.

Schulze, M. (2009). Famouso – family of adaptive middleware for autonomous sentient objects. website http://famouso.sourceforge.net/, date: 17.07.2012.

the Contiki project (2012). Contiki: The open source operating system for the internet of things. website http://www.contiki-os.org/, date: 2012-07-17.

Zug, S., Schulze, M., Dietrich, A., and Kaiser, J. (2010). Programming abstractions and middleware for building control systems as networks of smart sensors and actuators. In *Proceedings of Emerging Technologies in Factory Automation (ETFA '10)*, Bilbao, Spain.