Middleware for Adaptive Group Communication in Wireless Sensor Networks

Klaas Thoelen, Sam Michiels, and Wouter Joosen

IBBT - DistriNet, Katholieke Universiteit Leuven
3001 Leuven, Belgium
klaas.thoelen@cs.kuleuven.be

Abstract. While the size and heterogeneity of wireless sensor networks confirm the need and benefit of group communication, an intelligent approach that exploits the interaction pattern and network context is still missing. This paper introduces sensor middleware to dynamically select the most efficient alternative from a set of group communication mechanisms. The proposed solution leverages on an empirical analysis of the ODMRP multicast protocol and was evaluated by a proof-of-concept prototype running on the SunSPOT platform. Results show that network overhead is considerably reduced when using the sensor middleware for software deployment, reconfiguration and periodic data monitoring.

Keywords: Group Communication, Context-Awareness, Middleware, Wireless Sensor Networks, Multicast.

1 Introduction

As wireless sensor network (WSN) platforms mature and standardization in their networking stack advances, WSN deployments become larger and more heterogeneous. This heterogeneity is exposed by differences in hardware, software and even ownership of the individual sensor nodes. As a result, sensor nodes emerge with different responsibilities (provided services, quality levels), capabilities (processing power, sensor accuracy) and general properties (ownership, types of sensors, energy source). Paradoxically, this heterogeneity also leads to the opportunity of defining groups of sensor nodes with common characteristics.

Group communication support is needed so that a subset of nodes adhering to a certain commonality can easily be contacted. In the scope of large scale, long lived and dynamic WSN environments, we identify three interaction patterns that benefit from group communication [9,13]: (1) *software deployment*, (2) *dynamic configuration* and (3) *periodic actuation and monitoring*. Evolving requirements on deployed services and the network as a whole trigger both software deployment and configuration. The difference lies however in the frequencies of these interactions and the amount of data to be exchanged. (1) Software deployment involves the dissemination of relatively large chunks of data and should be kept to a minimum to extend the battery lifetime of the sensor nodes. (2) Reconfigurations can be performed more frequently than deployment, given that the

G. Parr and P. Morrow (Eds.): S-Cube 2010, LNICST 57, pp. 59–74, 2011.

network overhead is considerably smaller. (3) The exchange of monitoring and actuation data is highly frequent and often periodic in nature; in addition, the frequency of this periodic interaction is not static but susceptible to (changing) application preferences. The size of the data that is exchanged for monitoring however, is typically small in comparison with software deployment. These differences in interaction pattern characteristics influence the applicability of the various group communication mechanisms at hand.

In traditional networking, group communication is provided by multicast protocols that represent a group of nodes by a single address. The member nodes are not always bound by geographical properties and can be scattered throughout the network. The primary goal of multicast protocols is to deliver the intended packets to all members of the group with the least possible amount of transmissions per packet. This is typically achieved by the usage of an overlay tree or mesh in which all sources, members and needed intermediary nodes are connected. Creating these overlays, typically involves broadcasting the network in search of members and subsequent replies to the initiator of this so-called discovery phase. This exchange of control packets is commonly called *protocol overhead*, as it does not take part in delivering the actual data to the group members. In the face of mobility, changing connectivity and changing group memberships, these discovery phases need to be repeated at an appropriate frequency to keep the overlay up-to-date.

The problem that we address in this paper is that the protocol overhead introduced by multicast protocols is not always justifiable. Alternatives like broadcast and gossip mechanisms, introduce no or less protocol overhead [2,4] and although not group-aware, can be more efficient than multicast protocols. We argue that both the characteristics of the current interaction pattern and the network context need to be taken into account when performing group communication. Only in this way can we ensure that the most efficient communication mechanism is used.

The contributions of this paper are (1) the design of a context aware middleware layer that allows the selection of the most efficient mechanism at hand to perform group communication; to enable this, we performed (2) an efficiency analysis of a representative multicast protocol with regards to protocol overhead; (3) the evaluation of our prototype on the SunSPOT platform confirms the efficiency benefits of the proposed solution.

The remainder of this paper is structured as follows. In Section 2 we report on our efficiency analysis of the ODMRP multicast protocol and discuss the main results. We introduce our Communication Management Middleware in Section 3. We evaluate our prototype in Section 4 and elaborate on related work in Section 5. We conclude in Section 6.

2 Multicast Efficiency Analysis

With respect to the interaction patterns identified in the introduction, we compare a multicast and broadcast approach in terms of the amount of transmissions

needed to deliver data to all group members. This, of course, is dependent on the selected multicast protocol and how its overlay is created. In the following subsection we first discuss why we selected the ODMRP protocol and how it operates. We continue with a report on an efficiency analysis of ODMRP and conclude with defining a set of rules to follow in order to achieve efficient group communication.

2.1 Multicast Protocol Selection

We selected the On-Demand Multicast Routing Protocol (ODMRP) [14] because it is on-demand and source-initiated. Only when a source has a need to send data to a group, a multicast overlay is created, hereby reducing the protocol overhead to truly useful situations. This suits the identified interaction patterns as both deployment and reconfiguration require only short-lived overlays to fulfill their duties. Furthermore, periodic data publication only requires a constantly active overlay when the periodicity is high enough.

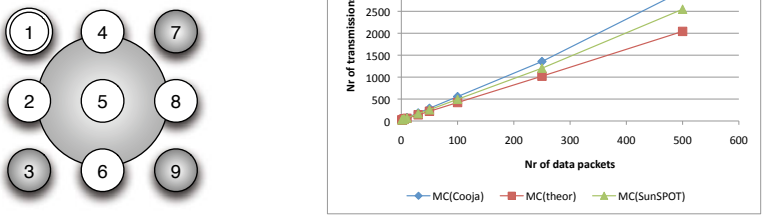
ODMRP is a mesh-based multicast protocol that uses the concept of a forwarding group in which only a subset of nodes forward data packets to the group members. The mesh overlay is created during a so-called discovery phase which involves broadcasting a Join Query packet to find all possible members of the group. At the reception of a Join Query, members of the group create a Join Reply and unicast it back to the source. Intermediate nodes receiving the Join Reply realize that they are on a path between the source and at least one member of the group and set a forwarding group flag before forwarding the Join Reply. In this way, a mesh of forwarding nodes is created that establishes routes between the source and the group members. Multicast data, transmitted by the source, is only retransmitted by nodes which have the forwarding flag set. Periodically, but only while the overlay is in effective use, discovery phases are repeated to refresh the mesh overlay. This period is defined as the discovery phase interval. The mesh overlay is maintained only in soft-state and fades away when no longer in use. No explicit control packets are thus required to leave the group.

Implementations. We implemented ODMRP on the SunSPOT [11] platform, the stable RED version of 090706. The SunSPOT version consumes 920 bytes or 0,17% of the total available RAM on a SunSPOT, compared to the 1040 bytes consumed by the built-in LQRP (Link Quality Routing Protocol) unicast routing protocol which is a derivative of AODV (Ad hoc On-demand Distance Vector) [7].

To be able to test the performance of our implementation in larger networks, we ported this implementation to a Cooja [6] compatible version. Cooja is a simulator primarily created to simulate sensor nodes running the Contiki operating system [1]. However, since it is developed using Java, it allows to simulate networks of Java-based application level sensor nodes.

2.2 Tests and Simulations

Driven by the interaction patterns introduced in Section 1, the variables in scope are the amount of data to exchange and the frequency of interactions with a



(a) (b)

Fig. 1. The 3x3 grid topology with an indication of the radio range around node 5 (a), and the accompanying graph of the number of transmissions needed to deliver a certain amount of data packets to the specified group (b)

multicast group. We define efficiency as the cumulative number of transmissions of all nodes in the network to deliver a certain amount of data packets to a group of nodes. This includes the transmissions of control as well as data packets. We compare the efficiency of the following cases: the SunSPOT implementation of ODMRP, the Cooja port of ODMRP, the theoretical optimum of ODMRP and a simple broadcast mechanism.

In our experiments, the discovery phase interval of ODMRP was kept constant at 30 seconds and the forwarding group refresh interval at 60 seconds. We consider node mobility as future work.

The theoretical optimum of ODMRP is the case in which ODMRP uses the minimum amount of protocol overhead to select the minimum number of forwarders. This case was used as a point of reference for the SunSPOT and Cooja implementations.

In case of the simple broadcasting mechanism, we calculated the number of transmissions executed when every node retransmits every packet it receives once. The number of transmissions per packet is thus equal to the number of nodes in the network.

The SunSPOT network under test was a 3x3-grid (see Figure 1(a)). Node 1 was used as the source of multicast data, with nodes 3, 7 and 9 being members.

In Cooja, we repeated the tests on a 3x3 grid with the same topology and radio range. As can be seen in Figure 1(b), the difference in efficiency was small and can be attributed to the difference in MAC and PHY layers used in both cases. This causes both setups to slightly favor different overlay meshes as members reply to the first JoinQuery they receive. The difference in comparison to the theoretical case can be attributed to the fact that in this case the most ideal mesh overlay is created with the least amount of forwarders. This is not always the case in the SunSPOT and Cooja versions.

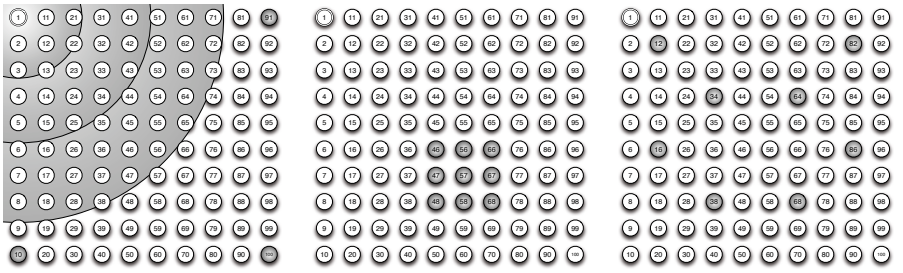


Fig. 2. Topologies of the 10x10 grid networks. The different sets of members (shaded circles) were chosen to represent a very dispersed group (a), a very condensed group (b) and an evenly spread out group (c). In (a), also the radio ranges are shown which were used during subsequent simulations.

Additional simulations were performed on a 5x5 grid and a 10x10 grid. In the first case we used the same radio range and the same network configuration as previous (one corner node as a source and all other corner nodes as members of the group). The 10x10 grid was however used to perform more extensive simulations with various network configurations. Figure 2 shows the radio ranges and sets of group members used. Node 1 was again used as the source of multicast data and the different sets of members were chosen to represent (a) a very dispersed group, (b) a very condensed group and (c) a evenly spread out group.

To indicate that our ODMRP implementation is satisfactory and that the measured transmissions effectively deliver the data packets to the group members, we include Table 1. This table shows the delivery ratio, or the average ratio of the number of data packets effectively received by the group members to the number of data packets sent by the source. The theoretical multicast and broadcast cases obviously hold a perfect delivery ratio of 1, but the experimental SunSPOT and Cooja cases come very close.

2.3 Multicast vs. Broadcast: Protocol Overhead

In this subsection we evaluate how the two alternatives of multicasting and broadcasting relate to each other in terms of protocol overhead.

Table 1. Delivery ratios of the tests and simulations

Network	Implementation	Delivery Ratio
3x3	SunSPOT	0,976
3x3	Cooja	0,994
5x5	Cooja	0,999
10x10	Cooja	0,980

Figure 3(a) and Figure 3(b) show the measured (or required) number of transmissions per number of data packets sent out by the source, respectively with and without the broadcast results. These specific figures are the average results from measurements in the 10x10 grid network, with medium radio range and evenly spread group members. All other cases resulted in similar graphs however.

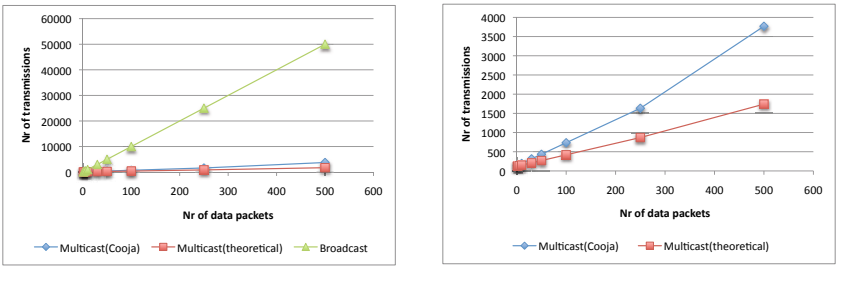
The simple, yet wasteful, broadcast mechanism has no protocol overhead. Every data packet transmitted by the source is retransmitted exactly once by every node when it is received. This is represented by the linear function nx in which n is equal to the number of nodes in the network and x to the amount of data packets transmitted by the source.

In the multicast case, the protocol overhead introduces a variable factor. A discovery phase is executed before the first data packet is transmitted and periodically repeated. Such a discovery phase requires an amount of transmissions equal to O.

$$O = JQ_N + JR_N + Ack_N \tag{1}$$

JQ_N, JR_N, Ack_N are respectively defined as the total amount of Join Queries, Join Replies and Acknowledgements transmitted by all nodes in the network. JQ_N is equal to n or the number of nodes in the network. JR_N and Ack_N are however dependent on the actual overlay mesh that is created at execution time. They are both functions of the size of the network, the position and the amount of members, the selected forwarders and the radio range of the nodes. The total amount of transmissions in the multicast case, during a single discovery interval, can be represented by the function $O + kx$, in which k is equal to the number of nodes which are selected as a forwarder.

We can compare both functions with the graphs in Figure 3(a). As the number of forwarders in the multicast case is typically much smaller than the total amount of nodes and thus $k < n$, we see that the slope of the multicast graph is



(a) (b)

Fig. 3. The amount of transmissions needed per number of data packets sent out by the source, with (a) and without (b) the broadcast results for the 10x10 grid network, with medium radio range and evenly spread group members

a lot less steep than the broadcast graph. A second discovery phase, performed between 250 and 500 data packets causes the multicast graphs to bend a little. In the theoretical case, the optimal set of forwarders is reselected and thus the graph only bends lightly because of the additional protocol overhead. In the practical case however, the selection of a different set of forwarders causes a greater bend in the graph. Subsequent discovery phases influence the slope to a lesser extent since after two discovery phase intervals forwarders are deactivated.

From Figure 3(a), we can conclude that in general multicast requires far less transmissions than broadcast, especially when the amount of data to disseminate is large. For a small amount of data packets however, the benefit of multicast is not that clear. Two of the three interaction patterns we identified involve the dissemination of small amounts of data. In the following subsection we therefore investigate the influence of protocol overhead on the amount of transmissions needed to disseminate an increasing number of data packets.

2.4 The Efficiency Threshold

In this section, we empirically determine the amount of data packets that need to be disseminated by the source in order to compensate multicast protocol overhead. This is defined as the *efficiency threshold* and visualized in Figure 4. The horizontal axis shows the amount of data packets disseminated by the source. On the vertical axis, we depict the total amount of transmissions (control and data packets) in percentages of the broadcast constant. To compensate for the various network sizes, we normalize the broadcast constant to 100 %. Only a single discovery phase is executed in the experiments. This happens before the first data packet is transmitted, but the respective protocol overhead is divided over the amount of data packets disseminated by the source. For a single data packet, the total amount of transmissions is thus the sum of all protocol overhead transmissions and the data packet transmissions. For a larger number

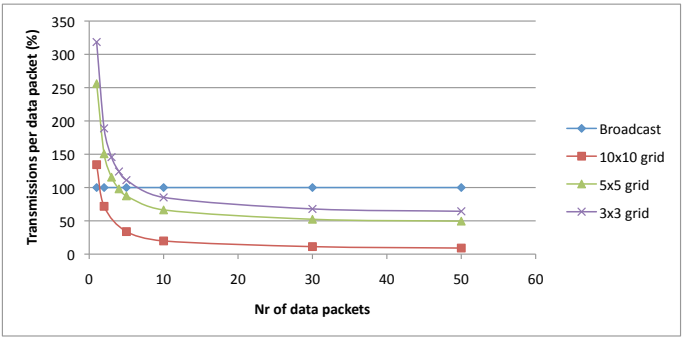


Fig. 4. The normalized amount of transmissions needed per data packet, for an increasing number of data packets sent by the source

Table 2. The average efficiency threshold for the various grids under test.

Network	**Threshold** (in number of data packets)
3x3	7
5x5	5
10x10	2

of data packets, the protocol overhead transmissions are divided over all the data packets, hereby reducing their impact.

In Figure 4, the curves represent the average of all our tests and simulations on the various grids (see section 2.2). We see that for a few packets, the multicast protocol overhead is not justifiable and broadcast is the better option. The multicast curves however quickly drop below the broadcast constant and level of to the average amount of forwarders selected by the protocol in each of the three grids. The amount of data packets on the horizontal axis that corresponds with the intersection of the multicast curve with the broadcast constant, is the efficiency threshold (see Table 2). The variation is explained in the following subsection.

2.5 Network Influences on the Efficiency Threshold

In this subsection we look at how various network parameters influence the efficiency threshold and explain its variation between the various grids.

The *discovery phase interval* and *periodic data interval* have no influence on the efficiency threshold. Any change in one of them, doesn't affect the amount of data packets required to justify the multicast protocol overhead.

For a fixed group of members, the *radio range* also has no practical influence. It affects the number of forwarders selected and thus the overall total of transmissions, but in most practical cases, this results in only a few more transmissions per data packet. Due to the steep decline of the curves in Figure 4, this practically has no influence on the efficiency threshold.

The *member distribution* and *member ratio* (the percentage of members in the network) in general also do not influence the efficiency threshold. The contribution of the broadcasting of Join Queries is too large for the variation in members and forwarders to have a clear impact on the total amount of transmissions. The amount of transmissions per data packet does change lightly in most cases, but not enough to influence the efficiency threshold.

The 3x3 and 5x5 grids however, expose extreme cases which do clearly affect the efficiency threshold. The small size of the grid, inherently causes the member ratio to be higher. In combination with the small radio range used, this causes nearly all nodes to be either a member or a forwarder. This drastically augments the amount of transmissions during both the discovery phase and the data dissemination afterwards. As a result, the efficiency threshold increases. In larger grids, this effect is also present, but only appears in extreme situations with a

very low radio range, combined with a very high member ratio. The efficiency thresholds in Table 2, were empirically determined and do not take into account such extreme situations.

2.6 Lessons Learned

With respect to the results described in the previous subsection, we deduce a number of rules which guide us towards efficient use of multicast. For this purpose, we define the following variables:

d = discovery phase interval
e = the efficiency threshold
p = periodic data interval

Using these variables, we can describe the following rules:

1. For a number of data packets below the threshold e, the protocol overhead is inefficient. In this case simply broadcasting is more efficient.
2. For a number of data packets above the threshold e, during a single discovery interval, the multicast protocol overhead is justifiable and multicast becomes the most efficient alternative.
3. For periodic transmissions to a multicast group, we can select a threshold for p below which multicast is the most efficient communication mechanism. In the other case, broadcast is the more efficient alternative. p has to obey to the following equation:

$$p < \frac{d}{e} \tag{2}$$

3 Communication Management Middleware

In this section we introduce our Communication Management Middleware (CMM) which leverages on the conclusions presented in the previous section. It efficiently delivers application data to a specified destination taking into account the interaction pattern and network context.

The CMM is situated above a traditional networking layer and below any application functionality (see Figure 5). The networking layer provides an extensible set of communication mechanisms like unicast, broadcast and multicast.

The need for a middleware layer arises from the fact that neither the applications, nor the multicast protocols should be held responsible for the selection of the most efficient communication mechanism. Applications might target their data to a group of interested nodes, but should not be concerned about how the data is delivered to this group. Additionally, multicast protocols allow for dissemination of data to a group of nodes but are not, and shouldn't be, concerned about whether they are to be used or not.

We continue this section with an overview of the key CMM building blocks and a discussion on the adaptations required at the networking layer.

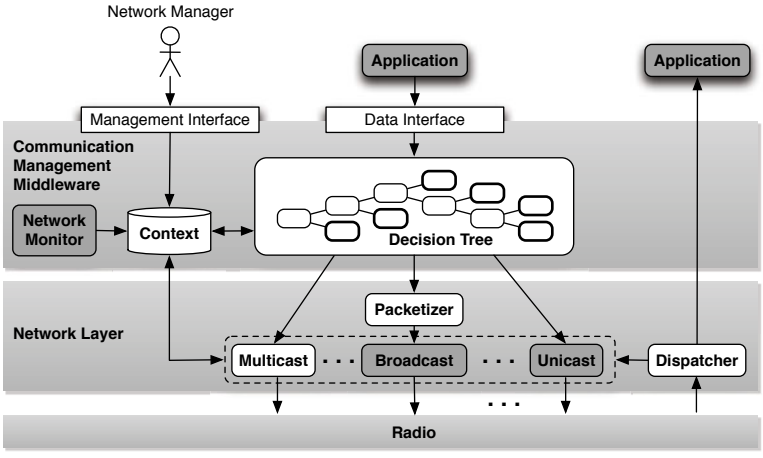


Fig. 5. Architectural representation of the Communication Management middleware in the network stack. The white blocks depict the contributions of this paper.

3.1 Component View

The Communication Management Middleware consists of a Data and Management Interface, a Network Monitor, a Context representation and a Decision Tree.

The *Data Interface* (see Table 3) is used by applications to pass data and the recipient's address to the middleware. The address can either be a unicast, broadcast or multicast address; further delivery of data is taken care of by the CMM. Through the same interface, applications can also notify the middleware of periodic transmissions by registering themselves together with the multicast address concerned and the sending interval time value.

The *Management Interface* and *Network Monitor* are both used to update the *context representation*. The latter contains a small database with tuples for various network parameters like the network size, the group size and the radio range. These parameters are used to determine the efficiency threshold and can be manually updated by a network manager through the Management Interface (see Table 4). The context representation is also interfaced with the various communication mechanisms in the network layer to query on their state or update various settings. With regard to the multicast mechanism, for instance, it can check for active multicast overlays or update the discovery phase interval.

Table 3. The Data Interface

void send(NetworkAddress address, byte[] data)
void registerPeriodicSender(NetworkAddress address, int interval, int ApplicationID)
void unregisterPeriodicSender(NetworkAddress address, int ApplicationID)

Table 4. The Management Interface

void setNetworkSize(int size)
int getNetworkSize()
void setGroupSize(NetworkAddress address, int size)
int getGroupSize(NetworkAddress address)
void setRadioRange(int range)
int getRadioRange()
void setEfficiencyThreshold(int value)
int getEfficiencyThreshold()

We conceptually include the Network Monitor, which provides a future automated alternative to the Management Interface. This alleviates the burden of manually updating the context representation from the network manager. Further research is however required on how the network parameters can be extracted from the network in an automated manner. We do foresee however that the Network Monitor will not be installed in the same form on all nodes of the network. A full-fledged Network Monitor will operate at the more powerful nodes like for instance gateways, while light-weight monitors or proxies will be deployed on the more resource constrained nodes.

The logic of the CMM is defined in the *Decision Tree*. The current interaction pattern at use is reflected by the data received via the Data Interface. In combination with the information retrievable in the context database, the Decision Tree evaluates which communication component at the network layer will deliver the data in the most efficient manner.

A detailed representation of the Decision Tree is presented in Figure 6. Depending on the destination address specified, the broadcast or unicast mechanisms of the network layer are activated, in which case the middleware layer simply passes the data and destination address to the network layer. The real logic comes into play when the destination address is a multicast address. In this case, first the size of the message is compared against the product of the efficiency threshold e and the available payload size of the network packets. The

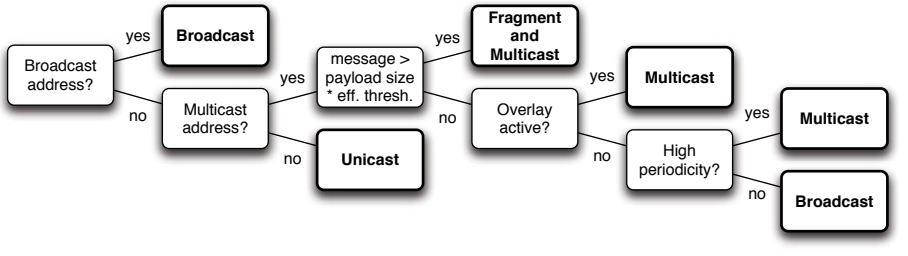


Fig. 6. The Communication Management Middleware decision tree

latter product indicates the amount of data required to result in enough data packets to be transmitted to justify the protocol overhead of a discovery phase. If the message is large enough, a discovery phase is executed if no multicast overlay is active and the message gets fragmented and transmitted via the multicast mechanism. In case the message is smaller than the mentioned product, we check whether an overlay is active, in which case we multicast anyway. Be it that no multicast overlay is active, we further check if the multicast group is registered as a group towards which periodic communication is targeted with a high enough periodicity. If this is the case, we multicast as well, if not we broadcast the message over the network and let the recipients detect whether they should process the message or not.

3.2 Network Layer Adaptations

The network layer provides an extensible set of communication mechanisms. Besides the unicast, broadcast and multicast mechanisms depicted in Figure 5, other mechanisms can be provided such as gossip protocols and anycast mechanisms.

To allow multicast data to be broadcasted, the broadcast mechanism has to be adapted to reflect the multicast membership of nodes. The Packetizer component adds a small header to all broadcast traffic. This header consists of a *multicast-flag* and an optional multicast address. If the broadcast data to be sent, is targeted to a multicast group, the multicast-flag is set to 1 and the multicast address is added to the header. In case of a normal broadcast, the multicast-flag is set to 0 and no multicast address is added.

At the receiving end, the *Dispatcher* component checks the destination address of all incoming packets. Packets destined to the node's unicast address or one of its multicast addresses are dispatched to the upper layer applications for further processing. In case a broadcast packet is received, its multicast-flag is inspected. If this is not set, the packet is dispatched to the upper layers as a normal broadcast packet. If the multicast-flag is set however, the packet is only dispatched to the upper layers if the node itself is a member of the specified multicast group. The multicast-flag thus allows receiving nodes to refrain from processing a broadcast packet when it is intended for a multicast group they are not a member of. The Dispatcher furthermore appropriately makes use of Unicast, Broadcast and Multicast mechanism for packet forwarding to the (other) destination(s).

4 Prototype Evaluation

We implemented a prototype version of the Communication Management Middleware on the SunSPOT platform (version Red-090706) and ported it to the Cooja simulator.

The SunSPOT prototype consumes 124 bytes of RAM and 2997 bytes of flash memory on a SunSPOT. This is respectively 0,02% of the 512 kbytes available RAM and 0,07% of the 4 MB available flash memory on a SunSPOT. We consider

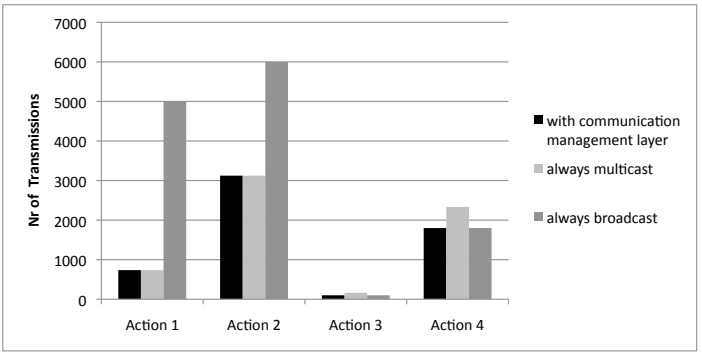


Fig. 7. The number of transmissions needed to perform the actions of the described use case

this very low with regards to the efficiency advantage that is added. Furthermore it indicates the applicability of our approach on more constrained sensor devices.

The following use case is used as a practical evaluation. In the 10x10 grid as depicted in Figure 2(a) node 1 is used as a management gateway to the WSN, through which deployments and reconfigurations are performed. The small radio range as depicted in the same figure is used and the discovery phase interval is set to 30 seconds. We performed the following set of actions using the Cooja prototype.

1. Deploy a temperature monitoring service to the group of opposite corner nodes 10, 91 and 100. The size of the component is 4KB which is fragmented over 50 packets.
2. During a 200 second period, the temperature monitoring services publish temperature readings to a group consisting of nodes 5 and 51. The temperature readings are published with a periodic time interval of 10 seconds and small enough to fit in a single packet.
3. A reconfiguration is executed which changes the periodic interval of the publication of temperature readings from 10 seconds to 30 seconds. This reconfiguration is performed on nodes 10, 91 and 100 and can be performed by the dissemination of a single data packet.
4. For another 200 seconds, the temperature components publish their temperature readings. This time using the new periodic interval of 30 seconds.

We compare the results of our middleware layer to the alternatives of always multicasting and always broadcasting. Figure 7 shows the total amount of transmissions in the network needed to execute this set of actions.

During the first action, the evaluation of the decision tree results in the selection of the *Fragment and Multicast* leaf. The temperature monitoring component is thus multicasted to the group members, which requires an average of 736 transmissions or about 15% of the broadcast alternative.

During the second action, the periodicity of the temperature readings is small enough to justify the protocol overhead of multicasting. When using the Communication Management Middleware, the components register their periodic publication to the defined group and the evaluation of the decision tree results in the selection of multicast as the most efficient alternative. This results in an average of 3124 transmissions or about 52% of the broadcast alternative.

Concerning the third action, the single packet needed for reconfiguration causes the Communication Management Middleware to refrain from using the multicast mechanism. The efficiency threshold e for a 10x10 grid is 2 as shown in Table 2 and thus the evaluation of the decision tree results in the selection of a *Broadcast* leaf. This results in a 100 transmissions as compared to an average of 161 transmissions in case multicast is used. This equals to 62% of the multicast alternative and is a reduction of 38%.

During the fourth action the temperature components are registered with the relatively high periodic interval of 30 seconds. This means that in case multicast is used, a discovery phase has to be executed for each transmission of a temperature reading. The evaluation of the decision tree, however results in the selection of a *Broadcast* leaf. This results in 1800 transmissions which is about 77% of the transmissions needed when multicast would have been used.

We can conclude that the Communication Management Middleware always selects the most efficient mechanism to perform group communication, and this at a very low cost with regards to memory consumption. The last two actions indicate the real additional value of our middleware. While a multicast mechanism can be provided, and will in some situations be the most efficient alternative, it is rather wasteful to use it as a default to perform group communication. Certainly when the overall task of WSNs is considered, which generally is periodic gathering of sensor data. The introduction of a middleware layer which takes the current context and interaction patterns into account, shows that considerable reductions of transmissions are possible.

5 Related Work

The work described in this paper is a first step towards more elaborated group communication support in the field of WSNs. While considerable work has been done on multicast routing in WSNs [10], few attention has been given to runtime selection of the efficient application of these protocols. In this section we discuss related work that adds to the functionality of the Communication Management Middleware.

First of all, the *functionality* of the Communication Management Middleware can be expanded to include group definition and creation. This would allow the middleware to create groups outside of the awareness of the applications. In [12], a reconfigurable group management middleware is presented. Group management is defined as the combination of identifying the need for new groups and discovering their members. The authors argue that different strategies need to be employed, dependent on the user services requiring group communication

and the system conditions (power level, connectivity, bandwidth etc.). A similar mechanism can be included in our context aware middleware.

Secondly, as possible outcomes of the decision tree in Figure 6, broadcasting is a frequent alternative to multicasting. The *efficiency* of group communication, and broadcasting in general, can be further improved by supporting more intelligent broadcast mechanisms like gossiping [2] or clustering [4]. A further efficiency gain can be realized by adopting a framework like ManetKit [8]. While we consider the multicast routing protocols to be black boxes, ManetKit provides reusable components and supports composition, decomposition and hybridisation of possibly multiple MANET routing protocols. It allows for runtime transition between unicast routing protocols while retaining reusable protocol state. The combination of a multicast version of ManetKit and our Communication Management Middleware would enable finer grained mechanism selection by replacing the fixed set of communication mechanisms with runtime composable alternatives which are better adapted to the current usage and network conditions.

In the third place, improving the *context awareness* of the Communication Management Middleware requires more elaborate studies on how multicast protocols are affected by the network context. Additionally, further work on network monitoring that extracts this information from the network is needed [5]. Node mobility is just one of the context parameters that need to be studied.

6 Conclusion

This paper presented the Communication Management Middleware which enables the selection of the most appropriate group communication mechanism based on the current interaction pattern and network context. The Communication Management Middleware incorporates rules that are the conclusions of an efficiency analysis of the ODMRP multicast protocol. We evaluated the middleware by performing a set of deployment, reconfiguration and periodic data monitoring actions. In all cases, the middleware selected the most efficient alternative, hereby considerably reducing the network overhead.

Acknowledgement. This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy, and by the Research Fund K.U.Leuven. It is conducted in the context of the IWT-SBO-STADiUM project No. 80037 [3].

References

1. Dunkels, A., Grönvall, B., Voigt, T.: Contiki - a lightweight and flexible operating system for tiny networked sensors. In: Proceedings of the First IEEE Workshop on Embedded Networked Sensors, Tampa, Florida, USA (November 2004)
2. Haas, Z.J., Halpern, J.Y., Li, L.: Gossip-based ad hoc routing. In: Proceedings of IEEE INFOCOM 2002, The 21st Annual Joint Conference of the IEEE Computer and Communications Societies, June 23-27. IEEE, New York (2002), ISBN 0-7803-7477-0

3. IWT STADiUM project 80037, Software Technology for Adaptable Distributed Middleware (2010), http://distrinet.cs.kuleuven.be/projects/stadium/ (visited June 2010)
4. Kwon, T.J., Gerla, M.: Efficient Flooding with Passive Clustering (PC) in Ad Hoc Networks. ACM SIGCOMM Computer Comm. Rev. 32(1), 44–56 (2002)
5. Lee, W.L., Datta, A., Cardell-Oliver, R.: Network Management in Wireless Sensor Networks. In: Handbook of Mobile Ad Hoc and Pervasive Communication (2007)
6. Österlind, F., Dunkels, A., Eriksson, J., Finne, N., Voigt, T.: Cross-level sensor network simulation with COOJA. In: SenseApp 2006, Tampa, Florida, USA (2006)
7. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (2003)
8. Ramdhany, R., Grace, P., Coulson, G., Hutchison, D.: MANETKit: Supporting the Dynamic Deployment and Reconfiguration of Ad-Hoc Routing Protocols. In: Proceedings of IFIP/ACM/USENIX Middleware, vol. 09 (2009)
9. Sá Silva, J., Camilo, T., Pinto, P., Ruivo, R., Rodrigues, A., Gaudêncio, F., Boavida, F.: Multicast and IP Multicast support in Wireless Sensor Networks. Journal of Networks 3(3), 19–26 (2008)
10. Simek, M., Komosný, D., Burget, R., Sá Silva, J.: Multicast Routing in Wireless Sensor Networks. In: Telecommunication and Signal Processing (2008)
11. SunSPOT, http://www.sunspotworld.com/ (visited June 2010)
12. Vieira, M.S., Rosa, N.S.: A reconfigurable group management middleware service for wireless sensor networks. In: Proceedings of the 3rd International Workshop on Middleware for Pervasive and Ad-Hoc Computing, Grenoble, France (2005)
13. Wagenknecht, G., Anwander, M., Brogle, M., Braun, T.: Reliable Multicast in Wireless Sensor Networks, 7. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze, Berlin, Germany, September 25-26, pp. 69–72, Freie Universitt Berlin, Fachbereich Mathematik und Informatik, Tech. Report B 08-12 (2008)
14. Yi, Y., Lee, S., Su, W., Gerla, M.: On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks. IETF Draft, draft-ietf-manet-odmrp-04.txt (February 2003)