# RMD: Reliable Multicast Data Dissemination within Groups of Collaborating Objects

Mihai Marin-Perianu and Paul Havinga
University of Twente, Enschede, The Netherlands
Email: {m.marinperianu, p.j.m.havinga}@utwente.nl

## Abstract

*Factory and industrial automation systems gradually start to incorporate wireless networks of smart objects and sensor nodes. In this context, one fundamental problem is the reliability of data dissemination, particularly in the case of total or partial network reconfiguration. We propose RMD, a reliable data dissemination solution targeting multicast groups of collaborating objects. Compared to previous work, our approach focuses on guaranteeing the data transmission rather than improving the delivery ratio. In addition, RMD scales better by utilizing the multicast support and proves more energy efficient due to cross-layer optimizations. To achieve these, we combine end-to-end aggregated acknowledgements with local error detection and recovery, and apply a selective listening scheme for reducing unnecessary radio operation.*

## 1. Introduction

Attaching wireless sensor nodes to everyday items enables the vision of smart collaborating objects. The inherent benefits (no wires, distributed intelligence, autonomy, fault tolerance) are particularly interesting for a large spectrum of emerging applications in factory and industrial automation systems [2]. Therefore, it can be envisaged that wireless collaborating objects will be able to provide complex services [12], beyond the traditional functionality of wireless sensor networks (WSNs).

In this paper we focus on reliable information dissemination in large networks of collaborating objects. The need for reliable dissemination stems from concrete applications in oil and gas industry, and transport and logistics processes, respectively [11]. In this context, the backend application often needs to access groups of nodes running certain tasks, in order to modify their parameters, or to deploy and run new tasks reflecting functional changes.

Solutions for WSN reprogramming have already been proposed, as well as several dedicated reliable transport protocols. Nevertheless, we can still observe two major problems that limit their applicability. First, due to the wireless medium, a mesh topology is typically assumed. Accordingly, many issues related to the data link (MAC) and routing are taken into account and solved at the transport layer. We argue that these approaches are either too specialised, or less efficient than a cross-layer solution, which can use the functionality (including topology control) of the most effective MAC and routing protocols. Second, end-to-end error control is most often disregarded, as being inefficient. However, the high degree of reliability required by industrial processes cannot be achieved without that. It is our goal to prove that a simplified end-to-end error control, relying on in-network aggregation of window-based acknowledgements, is possible without significant losses in the overall performance.

Distributing data or code to groups of nodes require a reliable multicast transport solution. The main requirement of such a protocol is to correctly deliver *all* the data to *every* intended recipient. The reliable multicast protocols developed for traditional networks are not directly applicable in the context of WSNs, due to the limited available resources and harsh operating environments. It follows that *energy efficiency*, along with *memory* and *bandwidth* constraints, represent difficult challenges. Mechanisms to deal with these issues include minimizing the amount of radio transmissions and receptions, as well as the time spent for idle listening, performing data aggregation, and using cross-layer interactions.

The paper is organised as follows. Section II presents the related work. Following this, Section III discusses the most relevant design considerations. Section IV describes the protocol operation. Sections V and VI provide analytical and simulation-based performance evaluation. Early experiences with implementing RMD are described in Section VII. Finally, Section VIII concludes the paper.

## 2. Related Work

We distinguish three broad related work areas: general-purpose multicast protocols, reliable transport solutions for WSNs, and sensor network reprogramming. For the first

topic, a comprehensive survey and taxonomy is given by Obraczka et. al [13]; the authors identify two major classes of applications that have motivated a whole suite of multicast protocols: multipoint interactive applications and data dissemination applications. Another interesting taxonomy [8] describes four generic approaches for reliable multicasting: sender-initiated, receiver-initiated, tree-based and ring-based. It is shown that *tree-based protocols* constitute the most scalable class of all reliable multicast protocols and also the best choice in terms of processing and memory requirements.

The problem of reliable transport in WSNs has received increasing concern lately. The proposed solutions consider performing hop-by-hop error recoveries (see PSFQ protocol [19]) or constructing a loss recovery infrastructure (see GARUDA [14]). Stann and Heidemann [17] propose RMST (Reliable Multi-Segment Transport), a transport layer designed for Directed Diffusion and discuss several design choices for MAC, transport and application layers. ESRT [16] and CODA [20] address mainly the problem of congestion detection and control.

Finally, the existing work on sensor network reprogramming can be split in two sub-categories: entire code delivery (see MOAP [18], Deluge [6] and MNP [7]) and difference-based update (see Maté [9]). The usual approach is to distribute the code or the updates in a hop-by-hop manner, with repairs being done within the local neighbourhood.

Our contribution consists in providing a cross-layer solution for reliable, multicast-based data dissemination (RMD). We conceive our work at the intersection of general purpose reliable transport protocols and network reprogramming. By using the multicast group support, the applications running in the network are more selective, closer to the real process and more scalable.

## 3. Design Considerations

In this Section, we discuss the considerations that underpin the design of RMD.

### 3.1. Tree-based Reliable Multicast

Tree-based reliable multicast protocols (TRMP) [8] are shown to exhibit good scalability properties by delegating responsibility to local groups (i.e., a node and its children in the tree). In addition, a considerable amount of network traffic can be saved by using local and aggregate ACKs. The local ACKs (or NACKs) trigger local retransmissions from a parent node to its children. The aggregate ACKs indicate correct overall reception to the source; a parent node will send an aggregate ACK only after it receives aggregate ACKs from all its children. Hence, the ACK-implosion problem is avoided and the source does not need to know the whole receiver set. The local behaviour of these protocols makes them suitable for the case of dense networks of collaborating objects and sensor nodes. Although these networks usually rely on a mesh model, a tree or cluster overlay organisation is often required for a proper, distributed operation. The reliable multicast protocol can thus directly use the available overlay structure. The construction and maintenance of such a topology are, however, beyond the scope of this paper.

### 3.2. Cross-layer Design

In our previous work [10], we reported on experiments with several reliable transport solutions in a sensor network testbed. The experimental results showed the following:

1. Traditional end-to-end error control cannot be applied directly, but has to be combined with local error detection and recovery.
2. Errors usually occur in burst and determine the loss of one or more packets. Therefore, a transport protocol should mainly handle packet losses, and rely on the MAC layer to deal with bit errors.
3. A cross-layer design, where the transport layer interacts with routing and MAC, can improve the energy and throughput performance.

According to these results, we design the tree-based reliable dissemination protocol to work jointly with the MAC layer. The functionality is split as follows:

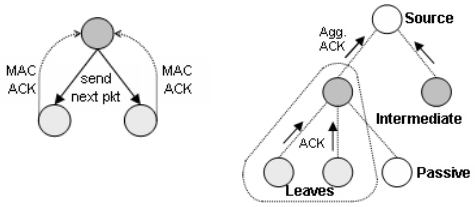
The MAC layer provides:

1. Neighbour information, from which a node can derive the identity of its parent and children in the tree.
2. Local (one-hop) ACKs and retransmissions (timeout or ARQ based).
3. Interaction points (callbacks) for signaling local ACKs to the dissemination layer.

The dissemination protocol:

1. Fragments the message into fixed size windows of packets (the packets are identified by sequences numbers).
2. Ensures end-to-end delivery through window ACKs.
3. Controls the MAC layer for the listening phase, in order to save energy (see the local multicast primitive defined in Section 5.3).

It follows that we distinguish between local ACKs (one hop, MAC-based, one for each packet), and window (or aggregate) ACKs (end-to-end, handled by the dissemination layer, one for each window). The reason is that MAC-based ACKs and retransmissions consume less time and energy than transport layer ACKs. Therefore, local ACKs, along with cross-layer listening, form the mechanisms to address the energy efficiency requirement. In order to guarantee the delivery, we use end-to-end aggregated ACKs that allow the source node to safely release the data from the memory. Finally, the memory and bandwidth limitations are addressed by reducing the necessary cache size and pipelining along the tree, respectively.




(a) (b)

**Figure 1. Protocol operation.**

## 4. Protocol Description

We consider the case of a dense network of heterogeneous nodes, organised into multicast groups according to application specific criteria (for example, nodes with similar sensors or placed in the same area). In what follows, we first introduce the network and error model, then describe the protocol operation, and finally comment on additional extensions.

### 4.1. Network and Error Model

For simplicity, we assume that only one group is involved in the dissemination process at one time and that the initiator, also referred as *source* node, is the root of the multicast tree. The multicast tree comprises the group members and the additional forwarders needed for propagating the data. According to the position in the tree, we distinguish between *source*, *intermediate* and *leaf* nodes (see also Figure 1(b)). Moreover, the parent-children relation is important, since RMD relies on local error detection and recovery. We assume that the source and intermediate nodes have on average $B$ children, i.e. the average tree degree is $B$.

We consider the following types of errors (assumed to be independent events):

1. *Packet losses*. Packets are lost or corrupted with probability $p_1$ when transmitted from parent to children. We assume that all loss events at all receivers are mutually independent. This error type is detected at the MAC layer, when the corresponding MAC ACK is not received, and triggers an automatic retransmission.
2. *Faults*. By fault we mean an error that causes the retransmission of the entire current window. There are three possible reasons for faults: transient link breakdowns (a node loses contact temporary with its parent, and recovers within the current window), topology changes (a node loses contact permanently with its parent, and has to register to another parent) or hardware errors (due to imperfect battery contacts, harsh environments, watchdog behaviour, etc.). We assume that a fault can occur with probability $p_2$.

**Algorithm 1**: Intermediate node operation

```
// Init phase
switch packet received do
    case NEW_MESSAGE
        init variables (sequence no., message length, etc.);
        forward NEW_MESSAGE to children;
    case ACK
        mark child ACK;
        if all children have sent ACKs then
            decide role, compute τ2;
            send ACK to parent;
        end
end
// Running phase
switch event do
    // Timers
    case τ1
        send current packet to children;
    case τ2
        restart to send window at rate τ1;
        send RETRY to parent;
    // MAC ACK callback
    case MAC ACK callback
        if all children have sent MAC ACKs then
            advance to next packet;
            if end window or end message then
                start τ2;
            end
        end
    // Packet received
    case DATA
        if correct sequence number then
            store packet in cache;
        end
    case ACK
        if all children have sent ACKs then
            send ACK to parent;
            advance window;
        end
    case RETRY
        extend τ2;
        forward RETRY up to parent;
    case RECOVERY_REQ
        send RECOVERY;
        restart to send window at rate τ1;
        send RETRY to parent;
    case RECOVERY
        init variables (sequence no., message length, etc.);
end
```

### 4.2. Protocol Operation

Algorithm 1 describes the operation of an intermediate node. During the initial phase, the new message is announced and the nodes initialize the dissemination session as follows:

1. Set protocol parameters, such as sequence numbers, message length, etc.
2. Forward the message announcement to all children,
3. Based on the replies from the children, compute the height in the tree, the timeouts and decide the role,
4. Acknowledge the parent and piggyback the height and role.

During the running phase, the source starts sending the packets from the current window, which are further pipelined down the tree by the intermediate nodes. The sending rate is $\tau_1$. Each packet is sent until all the di-

rect children acknowledge it through MAC ACKs (see Figure 1(a)) or a maximum number of retransmissions is exceeded. In the latter case, the faulty nodes are considered lost and ignored for the rest of the dissemination session, unless they request a recovery within the current window (according to the definition of a *fault* from the previous section). The MAC ACKs are signaled through the callbacks defined by the dissemination layer.

Nodes that are group members store every correctly received data packet in non-volatile memory, in order to perform an update/reconfiguration at the end of the dissemination session. Nodes that are just data forwarders cache only the current window in RAM, so that they can perform local repairs. The leaves of the tree indicate the correct reception of an entire window through window ACKs. Further, the window ACKs travel back to the root by being aggregated at every intermediate node (see Figure 1(b)). If a node does not receive window ACKs from all direct children before the expiration of $\tau_2$, it starts resending the window from its cache. In addition, it informs the upstream nodes about the error through a *RETRY* packet, so that unnecessary retransmissions are prevented. Finally, a node that experienced a fault, can request a recovery operation within the current window through the *RECOVERY_REQ* packet.

### 4.3. Discussion

There are a number of remarks concerning the network and error model, as well as further extensions to the basic protocol. First, the source node is assumed to be the root of the tree. Nevertheless, any node in the tree can be the initiator, if the tree is re-hanged with that node as root. The tree re-hanging process can be easily embedded in the initial phase. In addition, it has the property of preserving the average tree degree $B$, which is relevant for the generality of the analysis from Section 5. Second, in the case of a hardware reset, a node can resume the dissemination because the incoming message is stored in non-volatile memory (e.g. EEPROM) until session completion. However, in the case of a node with permanent hardware problems, its children are required to register to a different parent. This situation is handled by the tree maintenance procedure and is beyond the scope of RMD. Finally, the assumption on transient link breakdowns holds only for fixed nodes. For future work plans on handling mobility, see Section 8.

## 5. Protocol Analysis

In this Section, we estimate theoretically the energy consumption and average latency. The results of this analysis are then further used to characterize the performance of RMD.

### 5.1. Energy Analysis

We consider network communication to be the most power consuming operation, and therefore compute the

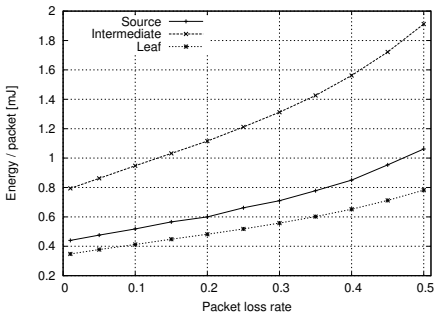



**Figure 2. Energy consumption on source, leaf and intermediate nodes.**

number of packets sent and received by a node during the transmission of one window. The MAC ACKs are considered to consume negligible time and energy compared to data packets and window ACKs.

Let us assume the simple case of one sender and one receiver. If $p_1$ is the probability of a packet loss, then the expected number of transmissions required for a correct reception is given by the geometric distribution

$$E[T_1] = 1/(1-p_1). \tag{1}$$

Now consider that the receiver can experience a fault with probability $p_2$ at time $t_i$ corresponding to the reception of the $i^{th}$ packet. As a result, the sender has to resend the window. We refer to this as a new *attempt*. The probability that a fault occurs during one attempt is $p_{2w} = 1-(1-p_2)^w$, where $w$ is the window size. The expected number of attempts until a successful window transmission is

$$E[N_2] = 1/(1-p_{2w}) = 1/(1-p_2)^w. \tag{2}$$

Since the number of faults is $E[N_2]-1$, the expected number of transmissions is

$$E[T_2] = \frac{E[N_2]-1}{p_2} = \frac{1/(1-p_2)^w - 1}{p_2}. \tag{3}$$

Let us analyse now the case of multiple receivers, where we denote by $B$ the average number of receivers (i.e., the average tree degree). As the errors at different receivers are independent events, Eq. 1 becomes [15]

$$E[T_1^B] = \sum_{k=1}^{B} \binom{B}{k} (-1)^{k+1} \frac{1}{1-p_1^k}. \tag{4}$$

The probability of a fault during one attempt becomes $p_{2w}^B = 1-(1-p_2)^{wB}$, and accordingly

$$E[N_2^B] = 1/(1-p_2)^{wB} \tag{5}$$

$$E[T_2^B] = \frac{1/(1-p_2)^{wB} - 1}{1-(1-p_2)^B}. \tag{6}$$

We can compute now the energy costs at the source, leaf and intermediate nodes, for sending a complete window and receiving the corresponding ACKs. For the source node, we have $E[X_S] = \xi_T\, E[T_S] + \xi_R\, E[R_S]$, where $T_S$, $R_S$ are

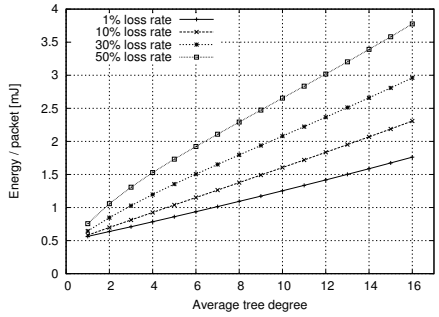



**Figure 3. Energy consumption on intermediate nodes, as a function of the tree degree.**

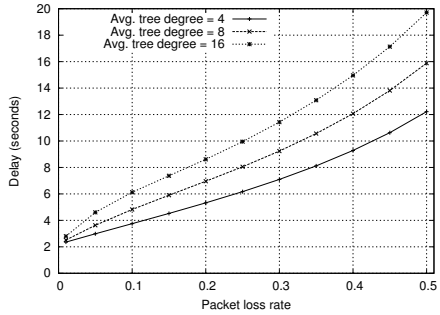



**Figure 4. Average latency.**

the number of transmissions and receptions, and $\xi_T, \xi_R$ represent the energy costs for sending and receiving a packet, respectively. Analogous equations can be written for intermediate and leaf nodes.

After successfully sending a window, the source node receives $B$ ACKs. Using Eq. 4 and 6

$$E[X_S] = \xi_T \, E[T_2^B] \, E[T_1^B] \; + \; \xi_R \, B. \qquad (7)$$

A leaf node sends one ACK per window and receives all the packets sent by the parent through local broadcasts, except its own packet losses. Therefore

$$E[X_L] = \xi_T \; + \; \xi_R \, E[T_2^B] \, E[T_1^B] \, (1 - p_1). \qquad (8)$$

Finally, an intermediate node carries out the double task of distributing packets and aggregating ACKs to and from its children. In addition, it is actively involved in the dissemination process, by maintaining the current window in the cache, in order to perform local repairs. The energy costs of a source node and a leaf node are therefore combined:

$$\begin{aligned} E[X_I] = \quad & \xi_T \, (E[T_2^B] \, E[T_1^B] + 1) \\ + \quad & \xi_R \, (E[T_2^B] \, E[T_1^B] \, (1 - p_1) + B). \end{aligned} \qquad (9)$$

Dividing the results from Eq. 7 - 9 by the window size $w$ yields the average energy costs per packet.

### 5.2. Average Latency

In order to examine the average latency, we derive an approximation of the transmission time required for the correct delivery of one window. We assume that the transmission time is mainly affected by the delay on the longest branch of the tree. We compute therefore the expected delay $E[D]$ along the branch of length $h$, where $h$ is the level of the tree. In the absence of errors, as the transmission process is pipelined, a window of size $w$ requires $\tau_1(w + h - 1)$ time to arrive at the leaf node. Additionally, the aggregated ACKs require $\tau_1 h$ time to travel back to the source. Any retransmission occurring along the pipeline increases the delay with one time unit $\tau_1$. Since the source nodes sends on average $E[T_S]$ packets per window and there are $h - 1$ intermediate nodes along the branch, each sending on average $E[T_I]$ packets, we have

$$E[D] = \tau_1 \, (E[T_S] + (E[T_I] - w)(h-1) + 2h - 1). \qquad (10)$$

The average latency per packet is given by $E[D]/w$.

### 5.3. Local Multicast Primitive

In Section 3.2, the design of RMD included cross-layer efficient listening. In what follows, we show how this can improve both latency and energy consumption.

Typically, the MAC layer provides two types of local (one-hop) communication primitives: unicast and broadcast. Let us assume that a parent node has $n$ neighbours, out of which $B$ are children involved in the dissemination process. The total energy spent for sending and receiving a packet from parent to children is $X_{ucast} = B\xi_T + B\xi_R$ for unicast, and $X_{bcast} = \xi_T + n\xi_R$ for broadcast packets, respectively. If we assume $\xi_T \approx \xi_R$, it follows that broadcasts are a good choice only for dense multicast groups ($B \geq (n+1)/2$). However, in the case of unicasts, the delay is $B$ times higher ($D_{ucast} = BD_{bcast}$) and the sender performs $B$ transmissions for each packet.

Controlling the MAC layer during the listening phase can offer a better alternative. More specifically, RMD can instruct the MAC to listen only for those packets sent by the parent node and intended for the specific multicast group. In this way, unnecessary listening is avoided, and the delay and energy consumption are minimal: $D_{mcast} = D_{bcast}$, $X_{mcast} = \xi_T + B\xi_R$.

## 6. Performance Evaluation

In this section, we examine the energy and latency performance of RMD. First, we analyse the numerical results from Section 5, then we compare via simulation with a well known transport solution, Pump Slowly Fetch Quickly (PSFQ) [19]. Throughout the rest of this section, we use the following values for the various parameters: $w = 5$ packets, $p_2 = 1\%$, $\tau_1 = 1$ s, $\xi_T \approx \xi_R = 0.2$ mJ.

### 6.1. Numerical Results

Figure 2 shows the average energy costs for source, leaf and intermediate nodes, as a function of the packet loss rate, for an average tree degree $B = 4$. As expected, the intermediate nodes are more loaded than the source or leaf nodes, since they are involved in both downstream and upstream traffic, and perform local repairs. An analysis on the separate transmissions and receptions shows that the intermediate nodes spend comparable energy on both operations, whereas the source and the leaf nodes execute mostly one operation, according to their role. Figure 3 reflects the effect of the average tree degree on the energy consumption observed at the intermediate nodes. We can note that the local recovery and ACKs aggregation mechanisms ensure a linear performance degradation, even under high error rates.

We further evaluate the average latency of RMD, computed as the average delay per successfully transmitted packet. Figure 4 shows the average delay variation with the loss rate, for several values of the tree degree. Other parameters that affect the dissemination latency are the sending rate at the source node and the tree height.

### 6.2. Comparison

We compare via simulation RMD with PSFQ. PSFQ is one of the first protocols for WSNs that aims at providing reliable transport with minimal energy expenditure. It has some similar properties to Scalable Reliable Multicast (SRM) [4], such as local recovery through request/repair packets and NACK suppression scheme. The key idea is to slowly inject messages (*pump* operation) into the network, while quickly performing NACK-based repairs from direct neighbours (*fetch* operation). The ratio between the pump and the fetch timers provide a number of possible retransmissions of the missed packets. In the simulation of PSFQ performed by Wan et al., the scenario concerns the re-tasking of 13 nodes, regularly placed along the hallway of a building, with 50 packets of program data. We compare both protocols in a more general setting, using OMNeT++ simulation environment [3]. We simulate a network of 100 nodes, randomly placed in an area of 200m x 200m, into which a source node attempts to disseminate 10000 packets. The radio range is set to 25 meters and the channel error (packet loss rate $p_1$) varies between 1% and 50%. For every value of $p_1$, we average over 10 random topologies.

In the following, we discuss a number of design issues that make a fully objective comparison difficult. First, RMD relies on a tree structure, and the characteristics of the tree affect the performance of the protocol. Since PSFQ is solely a transport solution, decoupled from the routing substrate, we are interested to compare only the reliable delivery portions of the two protocols. Therefore, we evaluate RMD using trees that are easy to build, rather than optimal. For every random topology generated by the simulator, we construct a hopcount-based spanning tree as follows: each node selects randomly as its parent one of the direct neighbours with a lower hopcount to the source.

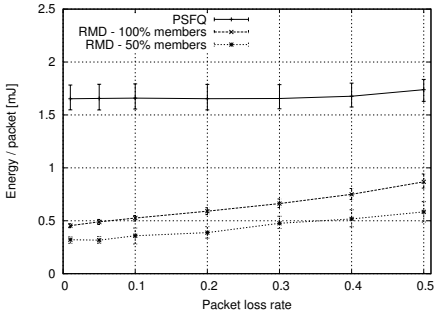



**Figure 5. Energy consumption comparison.**

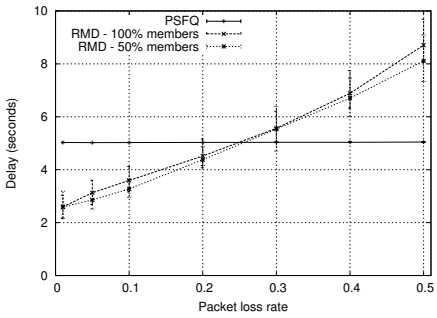



**Figure 6. Average latency comparison.**

Second, PSFQ utilizes broadcast and NACK suppression, in order to reduce redundant broadcasts and avoid NACK implosion. For this, the nodes listen always to the medium. In contrast, RMD interacts with the MAC to prevent unnecessary listening (this difference is not reflected in the simulations).

Third, the rebroadcast mechanism in PSFQ demands buffering along the way, otherwise the delivery ratio plummets with the error rate. There is no bound on the size of the buffers. Accordingly, in our simulation of PSFQ, the nodes behave ideally by storing all the packets. Note that in RMD, the intermediate nodes maintain a local cache with the current window.

Fourth, PSFQ attempts to batch up all message losses in a single NACK packet by looking for gaps in the sequence numbers. However, one packet cannot accommodate any number of gaps, and there is a limited time for recoveries between two consecutive pump operations. We set the number of possible retransmitted packets equal to the window size of RMD (5 packets).

Finally, the timer values affect significantly the performance of both protocols. For PSFQ, we set $T_{fetch} = 0.1$ s,

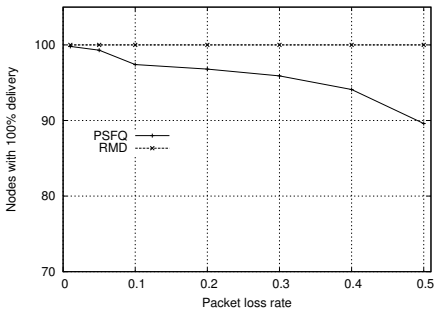



**Figure 7. Nodes with 100% delivery ratio.**

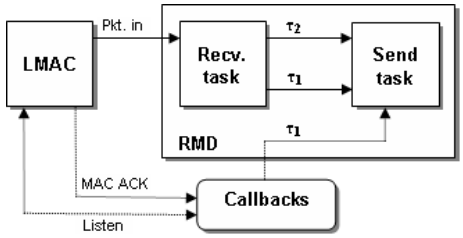



**Figure 8. RMD implementation.**

| | **Code** (FLASH) | **Data** (RAM) |
|---|---|---|
| **Dissemination** | 2.7 kB | 188 B |
| **LMAC** | 3.6 kB | 296 B |
| **OS kernel+drivers** | 21.6 kB | 400 B |
| **Total available** | 48 kB | 10 kB |

**Table 1. Code and data memory footprints**

$T_{pump} = 0.5$ s, i.e. 5 recoveries between two consecutive pump operations. For RMD, as mentioned, $\tau_1 = 1$ s.

Figure 5 compares the energy spent on average by a node in the network, under various loss rates. PSFQ incurs costs 2-3 times bigger than RMD, mainly because of the broadcast approach. An analysis over the data recorded from the simulation reveals that nodes running PSFQ consume 5-6 times more energy on receiving packets than on sending. Furthermore, if the multicast group does not include the whole network (which is an extreme case), the energy performance of RMD is even better (see Figure 5, RMD-50% members). Note that having 50% group members does not mean that the other 50% nodes are passive, as some of them may become forwarders at tree creation time. Figure 6 shows the average delay as a function of the packet loss rate. Since PSFQ has a fixed rate pump operation, the delay is affected only by the number of hops the packets have to travel, and not by the loss rate. Conversely, RMD is influenced both by the tree height (between 5 and 10, with an average of 7.7, for the various simulated topologies) and the loss rate.

The constant delay exhibited by PSFQ comes at the cost of having nodes that do not receive the entire message. This is shown in Figure 7. To cope with this problem, PSFQ introduces a periodic report operation, where the nodes send aggregate feedbacks to the source. In our opinion, this is only a partial solution, since (1) the report messages do not benefit from the hop-by-hop error recovery and, therefore, have higher chances of being lost along the path, and (2) there is no support for performing local repairs by examining the report messages.

## 7. Implementation

In this section, we present some early experiences implementing RMD in an experimental sensor network testbed. The practical goal is to disseminate data from the backend application, through a gateway node, to a multicast group specified by a group ID. In our experiments, we are using Ambient uNode 2.0 platform [1]. The onboard micro-controller is the Texas Instruments MSP430, which offers 48kB of Flash memory and 10kB of RAM. The radio transceiver has a maximum data rate of 100kbps.

Table 1 lists the code and memory footprints of the implementation. According to our cross-layer design, the reliable data dissemination interacts with LMAC [5], a TDMA-based, lightweight medium access control protocol. Figure 8 shows the basic architecture. RMD consists of two main tasks: *Send* and *Receive*. The *Receive* task is triggered by a radio packet received from LMAC. Within the *Send* task, regular packets are sent at $\tau_1$ rate, whereas window retransmissions occur after a $\tau_2$ timeout. There are two explicit interaction points with LMAC, where RMD defines callbacks and takes control over the normal operation: *MAC acknowledgment* and *prepare to listen phase*. The first is needed for advancing to the next packet in window. The latter implements the local multicast primitive.

We performed several experiments for a preliminary evaluation of RMD. We used the following simple setting: one gateway node attached to the serial port of a PC acting as the source, one intermediate node, and a variable number of leaves. A typical experimental run consisted in transmitting a message of 1 kB from the PC to all the leaf nodes. The packet size was 32 bytes. The communication on the serial link was performed reliably. At the end of the message, all the nodes reported back the number of packets sent and received, and the number of errors. By executing many experimental runs, we could study the effect of error rates on the energy consumption and compare with the numerical results from Section 6.1. However, due to the limited scale of the experiment, the maximum observed rate of MAC errors was relatively low ($p_1 \leq 20\%$) and no faults occured ($p_2 \approx 0\%$). Figure 9(a) shows the average energy cost on

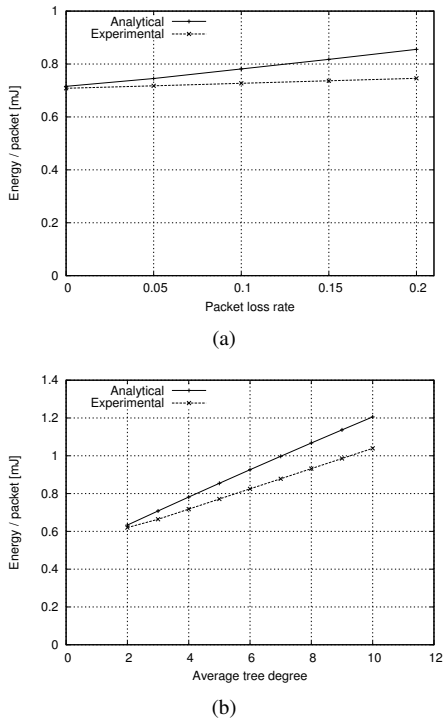



**Figure 9. Experiment - energy consumption.**

the intermediate node, for a setting with $B = 4$ leaves. The results for various values of $B$ and a constant loss rate $p_1 = 5\%$ are plotted in Figure 9(b). Overall, the experimental results confirm the linear dependencies indicated by calculus and simulations.

## 8. Conclusions

We presented RMD, a reliable, multicast-based dissemination protocol that supports dynamic reconfiguration of groups of collaborating objects and sensor nodes. We evaluated analytically and through simulation the performance of RMD. The results show the benefits of using cross-layer interactions between dissemination and MAC. Compared to PSFQ, one of the well-known transport protocols for WSNs, RMD ensures the data delivery to all recipients even under high error rates, while consuming 2-3 times less energy, and maintaining a comparable delay. Early experiences with implementing on sensor nodes yielded promising results, but also indicated the need for a future larger scale testbed, where different errors and error rates can be better observed.

For future work, we consider to support mobility, by distinguishing between two types of messages that an application can send: with *guaranteed* and *best-effort* reliability. For example, guaranteed message represent code updates, whereas best-effort messages are sent when altering parameters, such as the sampling rate. The dissemination of guaranteed messages will be done in two steps, i.e. the mobile nodes will be updated after stabilizing and registering to the new parents. Conversely, best-effort messages may not be received eventually by all group members, but still, the backend application will receive the list of non-updated nodes, for management purposes.

## References

[1] Ambient system. http://www.ambient-systems.net.
[2] Collaborative business items. http://www.cobis-online.de.
[3] OMNeT++. http://www.omnetpp.org.
[4] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5:784–803, 1997.
[5] L. V. Hoesel, T. Nieberg, J. Wu, and P. Havinga. Prolonging the lifetime of wireless sensor networks by cross-layer interaction. *IEEE Wireless Communication Magazine*, 12 2004.
[6] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *SenSys '04*, pages 81–94.
[7] S. S. Kulkarni and L. Wang. MNP: Multihop network reprogramming service for sensor networks. Technical Report MSU-CSE-04-19, Michigan State University, 2004.
[8] B. N. Levine and J. Garcia-Luna-Aceves. A comparison of reliable multicast protocols. *Multimedia Syst.*, 6:334–348, 1998.
[9] P. Levis, D. Gay, and D. Culler. Bridging the gap: Programming sensor networks with application specific virtual machines. Technical report, UC Berkeley, 2005.
[10] M. Marin-Perianu and P. Havinga. Experiments with reliable data delivery in wireless sensor networks. In *ISSNIP 2005*, pages 109–114.
[11] M. Marin-Perianu, T. Hofmeijer, and P. J. M.Havinga. Assisting business processes through wireless sensor networks. In *International Conference on Telecommunications 2006*.
[12] R. Marin-Perianu, H. Scholten, and P. Havinga. CODE: Description language for wireless collaborating objects. In *ISSNIP 2005*, pages 169–174.
[13] K. Obraczka. Multicast transport protocols: A survey and taxonomy. *IEEE Comm. Magazine*, 36(1):94–102, 1998.
[14] S.-J. Park, R. Vedantham, R. Sivakumar, and I. F. Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In *MobiHoc '04*, pages 78–89.
[15] S. Pingali, D. Towsley, and J. F. Kurose. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. In *SIGMETRICS '94*, pages 221–230.
[16] Y. Sankarasubramaniam, Ö. B. Akan, and I. F. Akyildiz. ESRT: Event to sink reliable transport in wireless sensor networks. In *MobiHoc '03*, pages 177–188.
[17] F. Stann and J. Heidemann. RMST: Reliable data transport in sensor networks. In *SNPA 2003*, pages 102–112.
[18] T. Stathopoulos, J. Heidemann, and D. Estrin. A remote code update mechanism for wireless sensor networks. Technical report, UCLA, 2003.
[19] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A reliable transport protocol for wireless sensor networks. In *WSNA '02*, pages 1–11.
[20] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell. CODA: Congestion detection and avoidance in sensor networks. In *SenSys '03*, pages 266–279.