On two Routing Mechanisms for Wireless Sensor Networks

Adrian Fr. Kacsó
Computer Science Department
University of Siegen
57068 Siegen, Germany
Email: adrian.kacso@uni-siegen.de

***Abstract*—In this paper we extend our previous implementation of the T-MAC protocol inside the sensor network simulator with a receiver-based routing (RBR) service and we propose and implement several performance optimizations. We investigate the impact of several MAC protocol parameters (listen time, receiver contention window, radio switch time, etc.) on the performance of routing protocols used in resource constrained wireless sensor networks. The main performance criteria we are interested in are the energy consumption (reflected by the active time the node is operational), the throughput and latency of the network in delivering replies to users' requests.**

Simulation results have shown that using the proposed optimizations improve significantly the performance of the RBR. Moreover, we compare the performance of receiver-based routing against the unicast within our implementation of the T-MAC protocol. Although in direct comparison the RBR approach is outperformed by unicast, we show that RBR can be efficiently employed for opportunistic aggregation inside monitoring areas with many sources or in dynamic network scenarios.

***Index Terms*—wireless sensor network, simulation framework, MAC and routing protocols, collisions.**

I. Introduction

A wireless sensor network (WSN) is a communication network consisting of a large number of sensor nodes that are randomly and densely deployed in a geographical area. The nodes operate unattended and are forced to self-organize themselves (in a multihop wireless network) as a result of frequent topology changes (due to node transient failures, addition or depletion) and to adjust their behavior to current network conditions. Each of the distributed nodes in the WSN senses individually the environment and they collaboratively preprocess and communicate the information to a sink.

Typically, a sensor node has limited energy and memory, restricted communication range and computation capabilities. The communication cost is often higher (several orders of magnitude) than the computation cost. For optimizing the communication cost in order to conserve energy, different data-centric routing protocols and in-network processing techniques have been proposed.

In query-driven WSNs, routing protocols determine on which routes messages (query and data) are forwarded between the sink and sources (nodes able to deliver the requested data) using data-centric approaches. In such *data-centric* routing schemes, the destination node of messages is specified by tuples of attribute-value pairs of the data carried inside the packets and not using globally unique identifiers (node addr).

When the distance between source(s) and sink is large, intermediate nodes forward the messages from hop to hop until they reach the intended destination. Selecting the next hop in order to establish a path (to a source or sink) can be either initiated by the sender or delegated to receiver nodes. In the first approach, the sender decides itself by analysing its internal tables where to send the message, whereas in the second approach the sender delegates the decision to all its neighbors, which distributively elect the best receiver. The strategy to select the next hop employs various metrics which allow to find different paths, e.g., energy-efficient, shorter, rapid, reliable paths, depending on the application goals.

Typically, the information collected in a sensor network is highly correlated, yielding a spatial and temporal correlation between successive measurements. Exploiting the data-centricity and the spatial-temporal correlation characteristics allows to apply effective in-network data aggregation techniques, which further improve the energy-efficiency of the communication in WSN. Aggregation can eliminate the inherent redundancy of the raw data collected and, additionally, it reduces the traffic in the network, avoiding in this way congestions and induced collisions.

The paper is an extension of [1] and is structured as follows. Section II presents the state-of-art and the motivation behind designing energy aware protocols for WSNs using cross-layer design. Section III describes the basic approach of receiver-based routing (RBR). Section IV presents the design (using cross-layering) of the RBR service (RBRS) inside our Timeout-MAC (T-MAC) protocol implementation. Section V discusses several optimizations made to RBRS. Section VI illustrates the performance of the RBR service by giving various simulation results and comparing it with unicast. Section gives more comparison results and Section VIII concludes the paper.

II. Related work and objectives

The main impact on the energy consumption of the nodes is given by the MAC protocol and only secondly by the routing strategy. A real energy benefit is achieved when using MAC protocols with an active-sleep regime and/or low duty cycles (such as S-MAC [2], B-MAC [3], T-MAC [4]). Considering

the scarce energy, communication and processing resources of WSNs, a joint optimization of the networking layers by employing a cross-layer design is a promising alternative to maximize the network performance, while reducing the global energy consumption.

Many of the current routing protocols are sender initiated [5][6][7], that is, the decision to which neighbor to route the just received message is taken by the sender. The sender maintains some internal neighborhood table (e.g., gradient table or routing table), which is inspected when messages need to be forwarded. Other protocols [8][9][10][11] use the receiver-based approach; in [9][10][11], the receiver contention scheme is used to develop a unified cross-layer protocol and in [8] to build mechanisms that lead to efficient data aggregation without maintaining a structure, namely the Data-Aware Anycast (DAA) and the Randomized Waiting (RW).

The spare energy and processing resources of battery powered sensor nodes require energy efficient communication protocols in order to fulfill the application objectives of WSNs. The use of both cross-layer design techniques [9][11][12] and aggregation [7][8][13] improves the overall network performance in terms of energy conservation.

The use of **cross-layer design** aims optimizing jointly several layers of the communication stack. Since for a resource constrained node strict layering is inappropriate [14] [15], we employ [12] a cross-layer design by allowing exchange of information (mainly) across application, routing, MAC and physical layers in order to optimize them.

Based on the application's requirements, the network topology, source placement and the aggregation function, a near to optimal **aggregation structure** (tree) can be constructed [16]. Various structured aggregation mechanisms (centralized [13][17] or distributed [7]) have been proposed. For query-driven sensor network applications, where several source nodes periodically report data to the sink, structured aggregation mechanisms are well suited, since the traffic pattern lasts for a long time and the overhead of construction and maintenance of the structure is low, compared with the energy benefits achieved through aggregation. For sensor network applications, where the sources are spread or the network topology is dynamic, the high construction and maintainance overhead for the aggregation structure can outweight the benefits of data aggregation. In such dynamic scenarios, mechanisms are required that achieve data aggregation without the construction and maintenance of a structure.

Concerning the simulator, we proposed in [18] and extended in [12] a modular, energy-aware network architecture of a sensor node as a flexible approach to design and plug-and-play various protocols at network and MAC layers, and to combine and analyse the impact of different parameters on the performance and lifetime of the WSN. We implemented our simulation framework SNF (Sensor Network Framework) using the OMNeT++ 3.4b2 discrete event simulation package and its Mobility Framework [19].

In the present paper we focus on the implementation of an additional RBR service to T-MAC for enabling applications to use both the unicast and the RBR service. An example of such an application is the opportunistic aggregation, where data packets are aggregated, if they meet each other on some node. Inside the source area the data packets are aggregated using the RBR service, while outside it the aggregated data packets are sent using *RTS/CTS* unicast ([20]). Source nodes having matching data (same type and required timestamp) are potential aggregators. If there is a potential additional aggregator closer to sink, it gets a higher priority in the RBR-associated transmission than an aggregator that is farther away.

III. Receiver-based routing (RBR)

The RBR service employs the use of *BRTS* (Broadcast Request-To-Send) control packet to get *BCTS* (Broadcast Clear-To-Send) responses from neighbors, which take initiative to participate in the transfer of the relevant information to sink. The *BRTS* control packet serves as a negotiation between the sender and all its potential receivers. After receiving the *BRTS*, each node determines (according to the information carried in the packet), wherever it participates in the transfer. In order to route a packet to destination the next hop should be *more appropriate* than the sender. Since there are several potential receivers, one needs to separate these receivers in priority groups, according to the available and propagated routing information. Nodes that achieve an *increasing progress* (i.e., are better placed or have more energy or data packets to aggregate, etc.) are placed in a higher priority group than others. The priority of a receiver node (i.e., its priority group) is established by the routing component (and communicated through the cross-layer to the MAC) and is based on the progress a packet would made if the node forwarded the packet. This prioritization is introduced to avoid *BCTS*-collisions (as more receivers may try to respond simultaneously). It is performed by a receiver contention mechanism to access the channel and is actually a computation of a random delay for the *BCTS*.

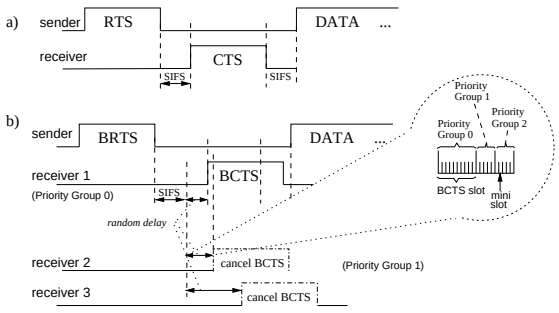


Fig. 1. a) Unicast (using RTS/CTS handshake) b) RBR contention.

Figure 1 shows the difference between the sender initiated next-hop selection (using *RTS/CTS*) and the randomized *BCTS* generation. According to which priority group j the node belongs, it waits for $\sum_{i=0}^{j-1} CW_{pG_i} + cw_j$, where CW_{pG_i} is the contention window corresponding to priority group i ($j \leq n-1$, assuming n priority groups) and $cw_j \in [0, CW_{pG_j}]$ is the delay time corresponding to j. This waiting scheme differentiates nodes of different progress into different priority groups, and attempts to assign different delays to nodes inside

the same priority group. The node getting the smallest delay wins the contention and sends a *BCTS* packet to the sender of the *BRTS*. If during the receiver contention, potential receivers hear a *BCTS*, they conclude that a node (with a shorter receiver contention) has accepted to forward the packet. Nodes that overhear a *BCTS* can switch to sleep state. However, in the case of *BCTS* collision (of nodes inside the same priority group) special attention should be paid (see §IV). When the sender receives the *BCTS* packet from the receiver that won the contention, it concludes that the receiver contention ended and sends a *DATA* packet to the intended receiver. Both *BCTS* and *DATA* packets indicate the other contending receivers the sender-receiver pair and the duration of the transmission (the latter only in the *DATA* packet). If the sender node does not receive a *BCTS* packet after $\sum_{i=0}^{n-1} CW_{pG_i}$, it resends the *BRTS* in order to restart the transmission. More details will be given in §IV. Finally, the receiver acknowledges the transmission with an *ACK* packet.

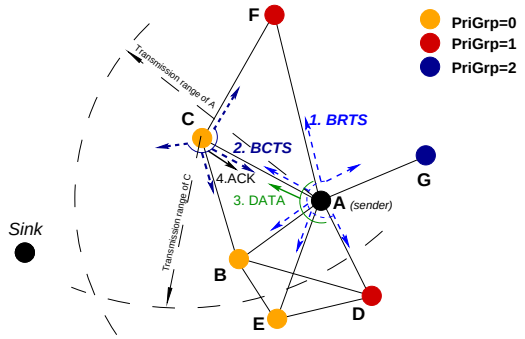


Fig. 2. The exchange of messages for a transfer between node A and C.

Assuming the neighborhood given in Fig. 2 the RBR algorithm is briefly described below and is illustrated in Figure 3.

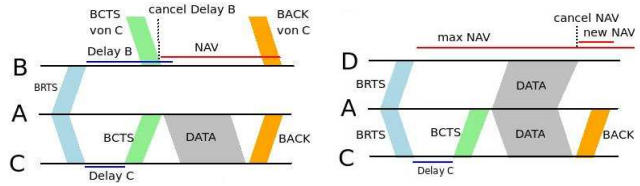


Fig. 3. Node A is the sender: **(a)** Node B and C compete for the reception **(b)** Node D remains quiet and adjusts its NAV-timer upon *DATA* reception.

1) Node A sends a *BRTS* with routing information.
2) Nodes B,C and D receive *BRTS* and compute the priority group (at network layer). More appropriate receivers calculate a lower priority (B and C), unsuitable receivers (only D) are passive (NAV) (see IV-A).
3) Receivers compute a *random time delay* according to the *RCW* of their priority group. Receivers (B and C) of the same priority group compete for the reception (see Figure 3(a)).
4) After expiration of the delay the receiver C (assume that C computed a lower time delay than B) sends a *BCTS*.
5) Potential receivers (B) who receive *BCTS* cancel their receiver contention and go passive (see Figure 3(a)).
6) A sends *DATA* to the *intended* receiver (C). Nodes still in receiver contention, which didn't overhear the *BCTS* (neighbors of A, but not of C, e.g., node E in Fig. 2) but are hearing the *DATA* will go passive. Passive nodes (including D) adjust their NAV timer (see Fig. 3(b)).

In which way receiver nodes are elected in different priority groups is a routing decision, which a node takes according to its local routing information. For example, when the routing uses geographic coordinates, the sender sends in the *BRTS* the sink and its local coordinates. Having this information, a potential receiver determines if it is closer to sink and correspondingly the node becomes a member of one of the predefined priority groups. The same principle is used when routing metrics as hop count or combinations of hop count and residual energy of nodes are used. Moreover, we may include in the priority groups some criteria to promote aggregation (see VII-3).

IV. T-MAC WITH RECEIVER-BASED ROUTING SERVICE

The T-MAC protocol uses a synchronized schedule in which nodes follow a listen-sleep regime. The main states of the protocol are illustrated in Figure 4. All nodes start in the `Startup` state by setting randomly a local timer and listening the channel. Each node switches to `Active Startup` state as soon as its own timer has expired or a foreign SYNC message has been received. At the end of the `Active Startup` state the node is synchronized and switches into *Active-Sleep* regime. In `Active Own` state the node has its own schedule during which it can receive and transmit. The protocol states for a unicast communication are illustrated in Figure 5.

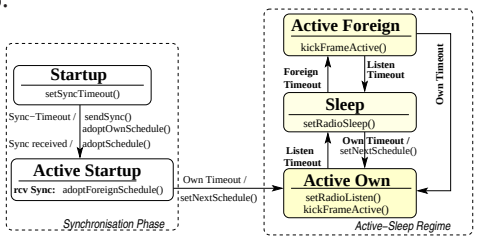


Fig. 4. Main states of T-MAC protocol.

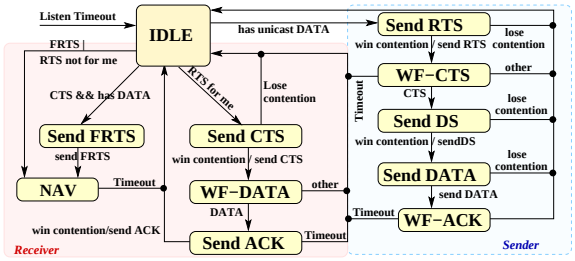


Fig. 5. T-MAC protocol state diagram for unicast (*Active Own* state).

The above states are almost self-explanatory and are common to RTS/CTS handshake mechanism ([20]). The reason to send data packets using the Request-To-Send (RTS) and Clear-To-Send (CTS) control packets is to reduce collisions, when two or more nodes transmit near the same time (hidden-station problem). This handshake mechanism is useful when the data packets are long, since if the packets collide, they are discarded, the energy is wasted and a later retransmission requires additional energy consumption both at sender and

receiver. Broadcast packets are never sent using the RTS/CTS handshake and are not acknowledged (using ACK packets).

In case of `Active Foreign`, the node is in active state of a foreign schedule, where it can only receive. Therefore, the state transitions are the same as in the left side of Figure 5. The Future-Request-To-Send (FRTS) packet is an extension meant to avoid the early sleeping problem ([4]).

We extend our previous implementation of T-MAC with the receiver-based routing service (RBRS). The protocol states for the receiver-based routing (RBR) are illustrated separately in Figure 6 for clarity reasons. The entry point for both state diagrams is the `IDLE` state, from where either the RTS/CTS handshake or the RBR service can be used.

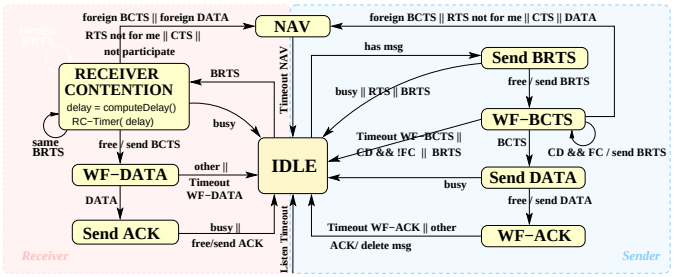


Fig. 6. T-MAC protocol state diagram for RBR (*Active Own* state).

In the following, we explain the protocol by describing the conditions and actions of the state diagram. Not mentioned terms/conditions and actions are self-explanatory. Later, we focus on particularities and optimizations.

We used the following abbreviations for conditions:

CD: collision detected between *BCTS* packets

FC: first collision between *BCTS* packets

same BRTS: reception of a previous *BRTS* packet (resent)

foreign X: reception of a packet *X* with different destination

other: reception of another message as expected

busy or *free*: channel is used or not during carrier sense

not participate: not a potential receiver (see *computeDelay*).

Moreover, we employ the action *computeDelay*, which calculates the receiver contention (*RC*) time to set the node's corresponding timer (`RC-Timer`) until it sends the *BCTS*.

Concerning the behavior of the T-MAC protocol, there are some aspects that need special attention. The first one is to customize the T-MAC's active period. The active period of a node ends, if during the timeout activation (TA) it does not detect any activity (e.g., an incoming packet, collision); then the node goes to sleep. Otherwise, if the node overhears or starts a communication, it schedules again a timeout after this communication finishes. The timeout value is set to stretch a small contention window and an *RTS-CTS* packets exchange. Hence, collisions and the resulted retransmissions extend the node's active time and increase, therefore, the energy consumption. The second important aspect is related to retransmission. Hereby, as the nodes are synchronized and we want to avoid them sending simultaneously after waking up, we need a large enough contention time before retransmitting. Note that a relatively large contention time helps avoiding collisions, but it also extends the active time of the node.

A. Cross-layer implementation of RBR

Receiver-based routing describes a communication method, in which each node has the choice to participate (or not) in the communication. In classic layer based protocols the communication is initiated by the application layer and the message passes the complete protocol stack, starting with the application layer and forwarded by each subsequent layers, network, data link and physical layers at sender, while on the receiving end it is forwarded by the physical layer in the opposite direction to upper layers until it reaches the receiver application layer.

According to the local information at sender, the routing protocols of the network layer decide to which neighbor node to forward the message. However, in case of receiver-based routing, the routing decision is not taken at the sender. Instead, the sender initializes the communication and potential receivers compete for the reception. The winner becomes the intended receiver for the sender. By shifting the routing decision to the receiver node, it is possible to use information for the routing decision, which is not local to the sender.

Due to the relocation of the decision, strictly speaking, the sequential forwarding of the message through the layers cannot be met. The sender initializes the communication with a message *BRTS*, which contains all the relevant routing information decision. Each node that receives the message will analyze this information and decide if it is a potential receiver. This decision is made at the network layer of the receiver. Thus, the network layer of the receiver is already involved in the communication before the actual data flow.

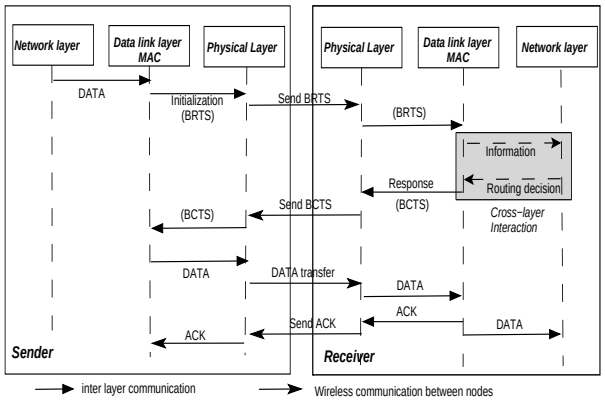


Fig. 7. Packet flow using the receiver based routing service.

When sending the reply all potential recipients are competing. The response time of the receiver depends on the decision of the network layer. The receiver which responds first (with a *BCTS*), becomes the communication partner for the sender in the current transfer.

As shown in Figure 7, in RBR mainly cooperate the data link layer and network layer in the potential receiver with each other. Therefore, the receiver-based routing application can be divided into different components, which can be assigned to the individual layers:

- *T-MAC RBR service*: service on the link layer for the

actual communication. Realizes the communication between nodes, but does not make any decisions.

- *Routing Unit*: service at the network layer. Handles communication with the MAC layer in the context of decision making.
- *Strategy Unit:* implements a routing strategy on the network layer. Take the actual decision on the basis of the transmitted routing information.

Figure 8 illustrates the message flow until the first *BRTS* packet is sent by the sender.

1) The application layer of a source node generates a DATA packet and sends it to the network layer. In the control information of the DATA packet there is also the interest identifier (*iID*) necessary to map the corresponding routing information stored at nodes. This step is omitted at relay nodes.
2) The routing unit calls the strategy unit assigned, which writes the required routing information (related to the received iID) in the dynamic part of the network header.
3) The network layer use a special target address (*L2RBR*) to signal the link layer to use the RBR service.
4) The RBR service from T-MAC copies the contents of the dynamic part of the network header in the dynamic header part of *BRTS* packet.
5) The link layer sends the *BRTS* packet as broadcast.

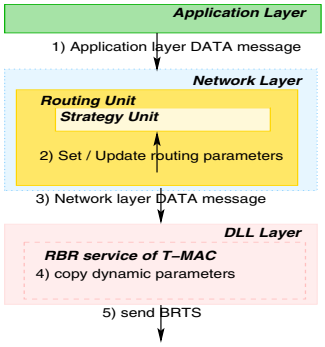


Fig. 8. Flow of messages (at sender) until the first *BRTS* is sent

Figure 9 shows the sequence flow of operations at the receiver after receiving a first *BRTS* message. Since the RBR service inside T-MAC must be flexible, in order to be able to process various RBR-strategies, at the first BRTS reception the communication between the RBR service and the Routing Unit and its associated Strategy Unit must be initialized using a cross-layer component. This is mandatory since the type and number of routing information parameters depends on the used strategy/strategies. Accordingly, it changes the number of parameters for the decision function call.

Figure 9 shows the sequence of steps until a BCTS response packet is sent.

1) The T-MAC RBR service receives the first *BRTS* packet.
2) The RBR service reads the dynamic part of the *BRTS* header and registers the individual parameters in the cross-layer. Only for the last parameter the notification service of the cross-layer is activated. This parameter is used in future receptions of a *BRTS* packet to trigger the call of strategy function (the actual routing decision) at the network level.
3) The RBR service stores the values of all routing parameters inside the cross-layer.
4) The number of parameters is registered in the cross-layer. This information type was registered at the initialization in the cross-layer with active notification. The Routing Unit has been registered as a subscriber. This information type serves as a trigger for the registration function of the routing unit to subscribe for actual routing information parameters.

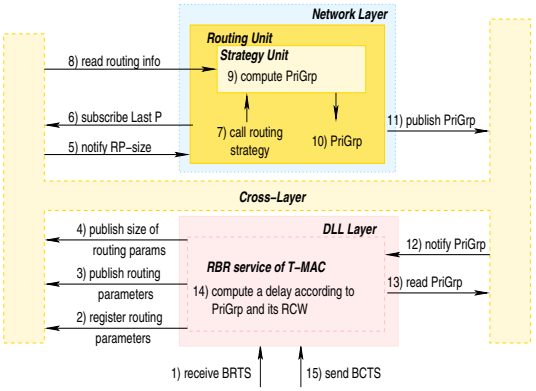


Fig. 9. Cooperation between the components at a receiver node upon reception of the first BRTS packet.

5) The notification service of the cross-layer informs the routing unit that the information type for the number of routing information parameter has been updated (and for subsequent receptions that the routing parameters are updated).
6) The registration function subscribes itself to be notified for updates of the routing information parameters (the last parameter update triggers the notification).
7) The registration function calls explicitly the strategy function, since this is not automatically called by the first update of the routing information at the first *BRTS* reception.
8) The strategy function reads the routing information parameters from the cross-layer.
9) The strategy function calculates the priority group (*PriGrp*) according to the received routing information parameters.
10) The priority group is passed to the routing unit.
11) The routing unit publishes the computed priority group in the cross-layer. This information type was registered at initialization and the cross-layer has registered itself as a subscriber.
12) The RBRS is notified about the updating of the information type for the priority group.
13) The RBRS reads the priority group from the cross-layer.
14) Using the calculated priority group the RBR service computes the delay for its *BCTS* packet. (If the node does not participate in the communication, it skips in the NAV state.)
15) When the timer expires the node sends a *BCTS* response if during the delay no *BCTS* or a *DATA* packet was received.

At subsequent receptions of *BRTS* the handling is analog, excepting the steps: the routing information parameters are already registered and the routing unit has subscribed to be notified when the routing information is updated, i.e., the call of the strategy function is triggered automatically.

V. RBR OPTIMIZATIONS

To design an effective receiver-based service implies to avoid collisions whenever possible and, if they still occur, to handle them efficiently. To that end, we propose in the following several optimizations.

A. First Group Weight optimization

Potential receivers compete for reception only within the same priority group. Each priority group has an own *receiver contention window (RCW)*. The smaller the *RCW* is, the higher is the probability that collisions occur. Collisions within the highest priority group have the largest negative impact on the performance of the RBR-service. In order to extend the receiver's contention window for the highest priority group (to reduce the likelihood of collisions) we provide an optimization referred as *First Group Weight*. The weight of the first *RCW* is set through a configuration file. For a larger *RCW* there will be fewer collisions, but the average duration of a data transmission extends also. The weighting should reflect the density of the network, i.e., it must scale with the number of neighboring nodes. For example, for a network with 5-7 neighbors we set the weight for the highest priority group to 40% and the rest of 60% is equally divided between the remaining groups (see Figure 10).

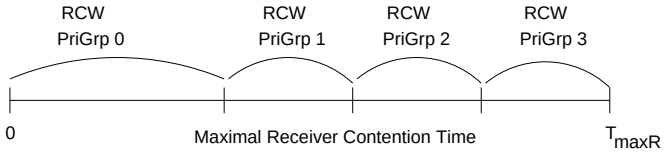


Fig. 10. Division of the T_{maxRC} for four priority groups.

Knowing the maximal neighbors density of a node one can analytically determine the minimal *RCW* size, such that the probability of no collisions has a given value p_{no_coll}. The *RCW* is given by

$$RCW = t_{sw}\sqrt[k]{\frac{k\sum_{i=1}^{n-1} i^{k-1}}{p_{no\_coll}}},$$

where t_{sw} is the switching time of the transceiver. The number of slots is given by $\frac{RCW}{t_{sw}}$.

B. Early Resend optimization

Potential receivers check after their own *receiver contention* expiration whether the medium is free. Note that the nodes have a single-channel radio, i.e., they are not full-duplex and require a switching time between the transmit and receive mode. Even though the medium is checked before each transmission, the switching time and the finite speed of radio waves propagation may lead to collisions.

The denser a network is, the more potential receivers compete for reception, which increases the probability of *BCTS*-collisions. Collisions of *BCTS* packets have a negative impact on the performance of the RBR-service, since the data transmission needs to be re-initialized. The retransmission includes the initial contention of the *BRTS* packet. In addition, nodes that heard the *BRTS* packet or one of the two collided *BCTS* packets go in *NAV* state. Since during this time the medium is not used, the throughput decreases while the latency and the power consumption increase. The optimization *Early Resend* ensures a faster recovering after *BCTS*-collisions by repeating the receiver contention process as soon as possible.

To that aims, after a collision of *BCTS* packets, a new *BRTS* packet is sent without an initial contention period. Nodes that have caused the collision don't notice instantly, since the single channel radio has a relatively long switching time from send to receive. These nodes require a short *WF-DATA* timeout until they reach the `IDLE` state (the *WF-DATA* timeout and the delay to resend *BRTS* are small compared to the average contention time). Nodes that observe the collision break their receiver contention and go to `IDLE` state. Since the nodes that received only one of the two collided *BCTS* packets are in `NAV` state, all neighboring nodes of the sender are either in `IDLE` or `NAV` state when the *BRTS* packet is resent; thus, the risk of a collision does not increase significantly. Usually, nodes in the `NAV` state are passive and do not respond to a *BRTS* message, excepting a retransmission of the *BRTS*. The nodes detect that they received a copy of the *BRTS* and start a priority group calculation. After the retransmission of the *BRTS*, all neighbors of the sender start a new priority group calculation and possibly a new receiver contention. The sender remains in state `WF-BCTS` (see sender state diagram in Figure 6). When a second collision occurs, the sender transits in `IDLE` state and restarts the communication completely from the scratch.

The scope of the *Early Resend* optimization is to recover faster after *BCTS*-collisions by skipping the initial contention at sender. By omitting the initial contention time of the *BRTS* the risk of a collision does not increase significantly since adjacent sender's nodes are either in the `NAV` or just switched into the `IDLE` state.

C. Change Priority-Group optimization

Depending on the topology of the network and the strategies applied (since the receiver priority group is computed by a routing strategy), it may happen that a sender cannot find an optimal receiver. Getting a non-optimal priority group at all potential receivers means a long *RC* time, which leads to a higher latency of the transmission and a higher energy consumption, as the active phase of T-MAC is extended. The optimization aims to prevent this by raising the group priority of all potential receivers, until at least one belongs to the highest priority group. This is achieved through the interest ID (*iID*) and *flag* fields inside the header of the RBR-service messages. The *iID* is necessary to map the data to the given interest (request). The one byte flag field (see Figure 11) is divided into a 1-bit field used in the *BCTS* response to notify the sender that the receiver has raised its priority group, and a 7-bit field for the value of the decrease in the priority group (in the *BRTS*) or the current computed priority group (in the *BCTS*). A potential receiver sends in the *BCTS* its current adjusted priority group (*PriGrp*).

Upon receiving the *BCTS* response, the sender verifies the priority group. If this does not correspond to the optimal *PriGrp*, it increments a counter for the specified *iID*. If the

counter reaches a threshold (specified in a configuration file), at the next transmission of a data message for the same interest, the sender sets in the flag the required decrease (a multiple of the threshold) in order to raise the *PriGrp* of all potential receivers. The potential receivers read the flag from the sender's *BRTS* message and, if the value is greater than zero, they raise their own *PriGrp* with the given value. Receivers send in their *BCTS* response the new *PriGrp* and the flag that indicates that they have raised their priority group.

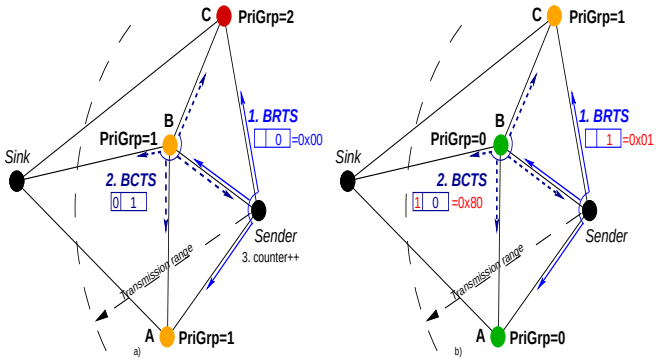


Fig. 11. Change Priority-Group: **(a) 1:** Sender sends *BRTS* **2:** Node B computes *PriGrp* 1 and since no node has the smallest priority group, it wins the *RC* and sends *BCTS* with its *PriGrp*. **3:** Sender increments a local counter for not optimal *BCTS* response. **(b) 1:** Counter reaches threshold: sender sends *BRTS* with request to adjust the *PriGrp*. **2:** Receivers increase their *PriGrp*. B wins the *RC* and sends *BCTS* by setting the first bit, i.e., it has raised the priority, and its current computed *PriGrp* (flag = 0x80).

If during the priority increase optimization a new potential receiver is added to the neighborhood of the sender, the new node computes a better priority group that the optimum. This receiver will not set the 1-bit field in the *BCTS* response, notifying the sender that it has computed the highest priority group, without using the priority group increase request. In addition, the receiver contention is reduced by half time, so it is likely that this node wins the contention and this receiver can send its *BCTS* response. If the sender receives a *BCTS* with no flag set, it resets the counter for the corresponding interest. For the next transmission the sender cancels its request for raising the priority group of all its potential receivers.

VI. Performance Evaluation

For the evaluation of the RBR service, we analyze the following performance parameters: active time (which highly impacts on the energy consumption), throughput and latency.

Simulator settings: in the simulation we used the Chipcon CC1000 (used by MiCA2) and CC2420 (used by Telos) single channel radio transceivers with the following parameters:

	current [mA]			**power** [mW]			
	SL	RX	TX	SL	RX	TX	Switch
CC1000	0.11	10	8.3	0.33	30	33	25
CC2420	0.02	24	14	0.04	48	28	30

	switching time [μs]				
Transceiver	*SL→RX*	*SL→TX*	*RX,TX→SL*	*RX→TX*	*TX→RX*
CC1000	850	850	10	850	850
CC2420	580	580	10	580	580

For T-MAC we set the listen time to 30ms and the frame time to 600ms. The overhearing avoidance flag is disabled.

A. Active time

The active time of a node significantly influences its energy consumption. Activities in the node's neighborhood extend the node's active time, since they reset the active timeout. For the measurements we used a multihop sensor network with m hops between source and destination, and a variable neighborhood density of n neighbors (see Figure 12). The source generates data packets at each 200ms, and the simulation time is 1 minute. To minimize the effects of subsequent transfers in the first measurement we chose $m = 1$.

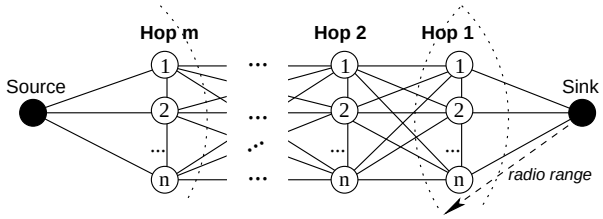


Fig. 12. Simulation scenario.

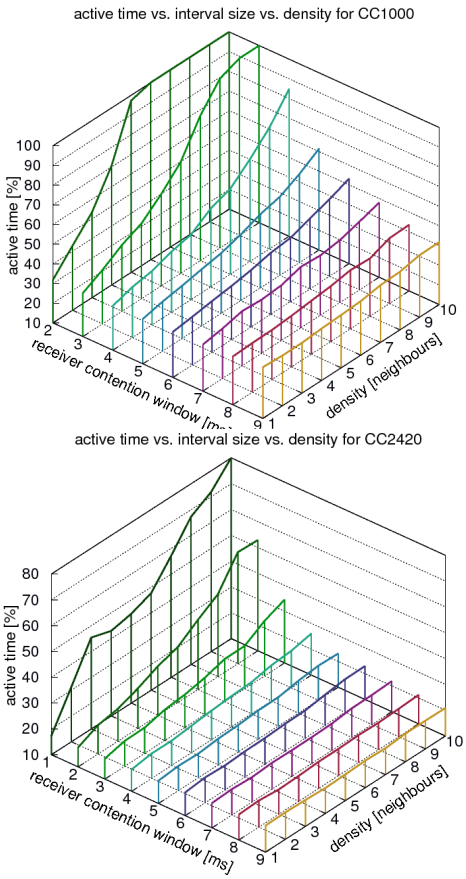


Fig. 13. Impact of the network's density and *RCW* size on active time.

Figure 13 shows the effects of the density of the neighbors and the size of the *receiver contention window* (*RCW*) on the active period of both transmitters.

The CC1000 transceiver has a much higher active time than the CC2420 transceiver. The difference cannot be explained only by the higher transmission rate (250 kbps compared to 76.8 kbps), since the proportion of time in which the transceiver is in transmitting mode is approximately 2%. Rather, it seems likely that additional causes generate this

behavior. To investigate this closer one needs to analyse the number of events that occur during a transfer. Events that negatively affect the behavior and extend the active period are collisions of messages and their consequences.

For small *RCW*, the active time increases quickly with increasing of the network's density. This leads to frequent collisions of *BCTS* responses, whereby the receiver contention needs to be repeated. Therefore, the active timeout is set again. For large *RCW*, the active time remains relatively constant. The negative impact on the active time by a long-lasting transmission (due to the large *RC*) will be compensated by rare occurrence of *BCTS*-collisions.

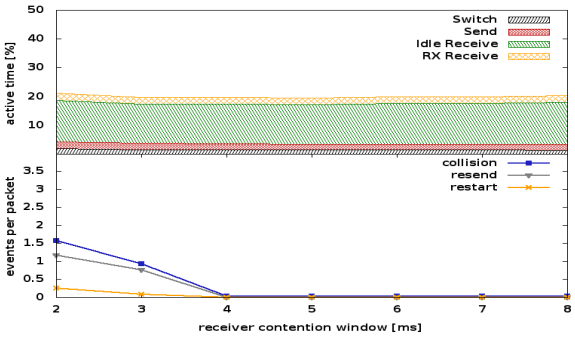
Next we investigate the possible causes for an extended active time. The measurements in Figure 14 show the influence of the *BCTS*-collisions on the active period of the CC2420 transmitter. Each graph shows the number of events (per packet) occurring (lower part) and the active time of the transceiver (upper part) according to various *RCW* sizes and different neighbors densities.

A *BCTS*-collision occurs mainly due to the fact that the difference of two or more calculated receiver contention times is smaller than the transmitter's switch time. Using a single channel transmitter, a potential receiver node is able to check the channel for activity until its transmitter switches from receive to send. It is possible, that during switching another node starts to transmit its *BCTS*. The first node cannot notice that and, therefore, the length of the *RCW*, especially the difference between two receiver contentions is important. The shorter the switching time of the transmitter is, the smaller the difference between two receiver contentions can be. That means, the smaller the switching time of the transmitter is, the more opportunities have other nodes to compute a receiver contention that does not lead to a collision.

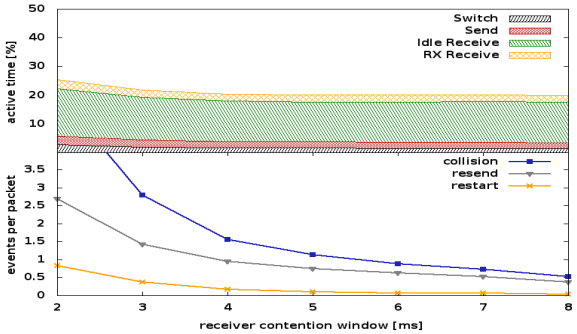
During a transfer, the *resend event* occurs at the first collision of the *BCTS* responses. This event is triggered by the *Early Resend* optimization, when after the first *BCTS* collision, a new *BRTS* is sent without contention (see V-B) . Since the initial contention time is omitted, this event has a relatively small influence on the extension of the active time compared to restart the transmission. If a second *BCTS*-collision happens, the whole transfer must be restarted, including the initial contention. In graphs this corresponds to the *restart event*, which has a much higher impact on the active period, since it increases the fraction of time that the radio spent in *Idle Receive* state as part of the whole active time.

Hence, for increasing neighbor density and large *RCW*, the active time remains relatively constant, despite the increase of the number of negative events, since if a transfer was successful, it is likely that a low delay time has won the contention. The number of retransmissions will be higher, but the transfers are in average completed faster and this compensates partially the negative effect.

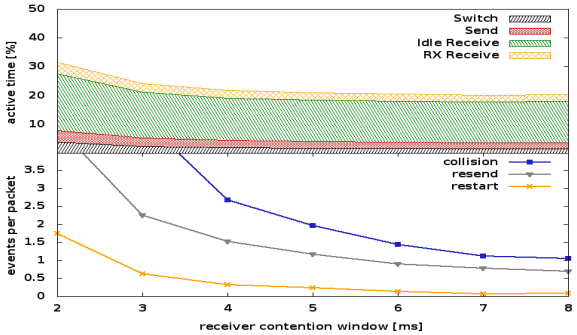
In case of the CC1000 transmitter the same measurements lead to a larger number than for CC2420 (figures are omitted here). This is due to the fact that the CC1000 has larger switching time, which increases the frequency of *BCTS*-collisions. A



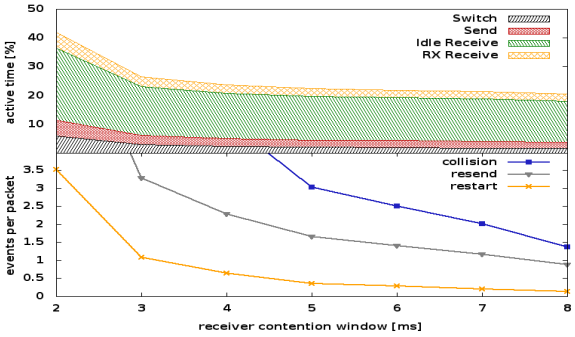
(a) 2 potential receivers



(b) 4 potential receivers



(c) 6 potential receivers



(d) 8 potential receivers

Fig. 14. CC2420: Impact of *RCW* size on the active period of the transmitter (reflected by the procentage of each active radio state (sleep is what left until 100%) for different density of neighbors. Each graph gives also the number of relevant events (collision, resend, restart) for different size of *RCW*.

larger switching time means that during a node is checking the medium and switching to send the probability that another node (during this time) starts to transmit is higher and, thus, more *BCTS*-collisions occur.

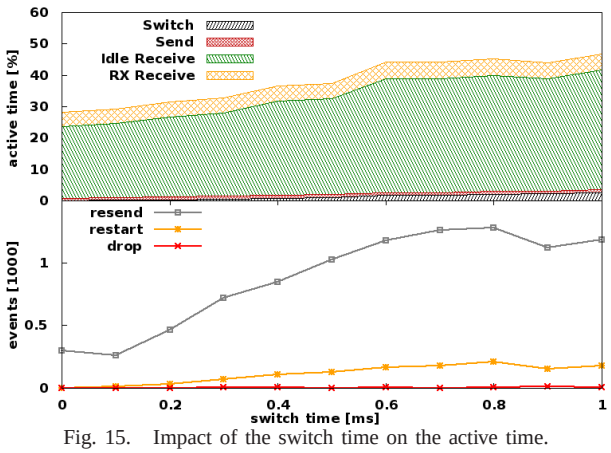


Fig. 15. Impact of the switch time on the active time.

Figure 15 shows the impact of the switch time on the active time for a fixed *RCW* (4ms) and a given network density. For smaller switching times, the active time decreases. By increasing the switching time, the active time increases also, but more than the sum of the individual switching times. The cause is the higher number of *BCTS*-collisions when the switch time increases.

B. Throughput and latency

In order to measure the maximum possible throughput and the source-sink latency in dense network, we set m=5 and n=3.

Figure 16 shows the drop rate and the latency for different *RCW* sizes. In low traffic networks the latency is independent of *RCW*. With increasing data rate increases also the latency and its variance. This occurs rather for small than for large *RCW*. An increasing latency leads also to packet loss, as can be seen in Figure 16.

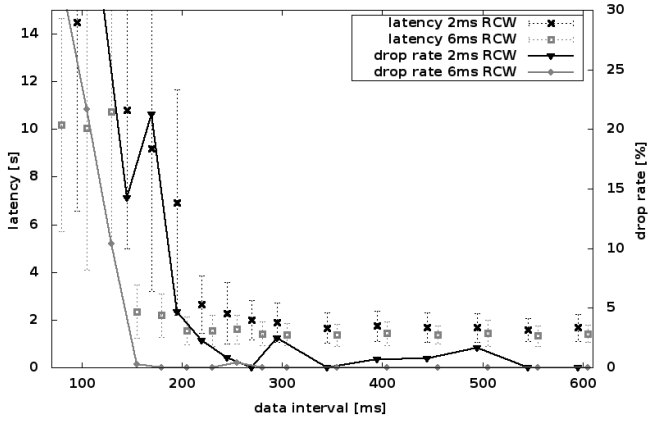


Fig. 16. Throughput and latency for *RCW* of $2ms$ and $6ms$.

The reason lies in the interaction of different transfers within a region. The RBRS uses instead of the SIFS (Short Interframe Space) between the *RTS* and *CTS* control packets a receiver contention, which defers the *BCTS* response. As a potential receiver is in the receiver's contention, a node in the neighborhood, which has not received the original *BRTS*, can start itself a transmission by sending a *BRTS* packet. If this is the case, only one of the two transmissions can be successfully completed, assuming that no collision has happened. The hidden-station problem cannot be effectively solved by the RBRS. If a collision occurs, both transfers must be re-started. These two cases occur more often when the data rate increases. Additionally, the *BCTS* collisions mentioned in the previous measurements occur very frequently in small *RCW*. If the number of retransmissions exceeds a given threshold, the packets are deleted.

VII. Comparisons and Evaluations

In order to compare the energy savings achieved through the proposed optimizations we consider here the sensor network given in Figure 17.

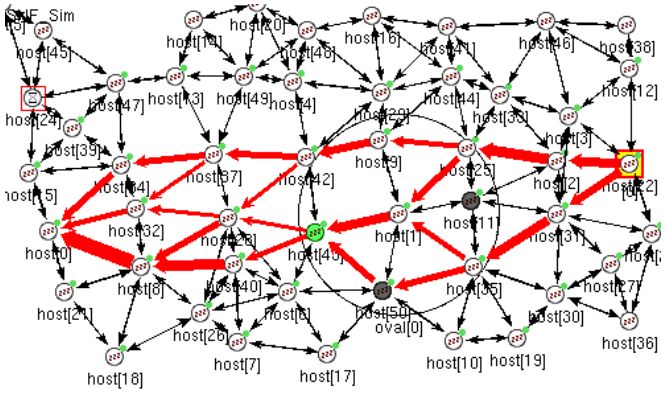


Fig. 17. A simulation network with 51 sensor nodes.

For this network the source is node 22 situated on the right side of the figure and the sink is node 0 situated on the left side of the figure, where all the red arrows end.

1) Comparison RBR with and without optimizations: In order to compare the energy savings achieved through the proposed optimizations, we have enabled and disabled the optimizations.

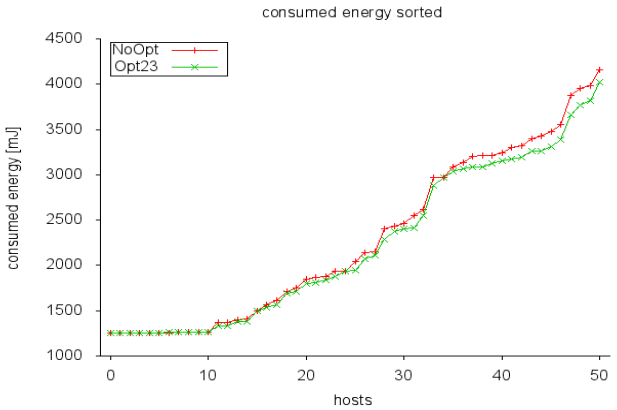


Fig. 18. Energy consumption using RBR with and without optimizations.

The comparison of the energy consumed in both cases for a simulation time of 3 minutes considering three priority

groups and a weight for the highest priority group of 60%) is illustrated in Figure 18. One can observe that with all three optimizations the energy consumed by some nodes improves up to 7%, but there are also nodes where the energy consumption increases. The overall energy consumption is reduced when using optimizations by at least 4%.

2) Comparison RBR with unicast: Next we compare the RBR service with the unicast in our variant of T-MAC [1]. We set up two scenarios, one for the RBR and one for the unicast; both use the same application and network layer. At network layer, the routing information is propagated through interest refreshes without extra traffic for routing. For data routing we use in both cases a strategy based on hop count and node's residual energy. The simulation time is 3 minutes, the source generates data packets at each $200ms$ and the *RCW* for the RBR-service is $4ms$. For the unicast scenario the overhearing avoidance flag inside the T-MAC is enabled, meaning that nodes in the NAV state turn off their radio to conserve energy.

Figure 19 illustrates the energy consumption for the RBR-service and unicast respectively.

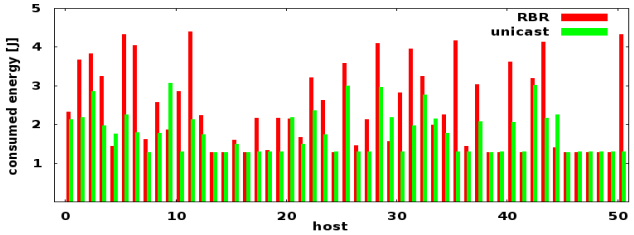
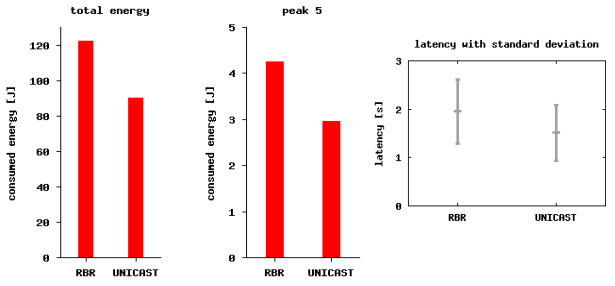


Fig. 19. Energy consumption for unicast and RBR services of T-MAC.

The comparison of energy consumption shows a higher energy consumption when using the RBR service. The reason is the additional receiver contention of the RBRS and the *BCTS*-collisions, since both increase the active time of a node. In case of unicast, since the hidden station problem is successfully solved, there are fewer adverse events and a transfer can be faster terminated.

Using the same routing strategy, the total energy consumption for unicast is $91J$ in comparison to $123J$ for RBR (see Figure 20(a)). The situation remains similar when we compare the energy consumed by the five most heavily loaded nodes (see Figure 20(b)).



(a) total (b) peak five (c) latency

Fig. 20. Energy consumption: **(a)** total energy consumption **(b)** average energy consumption of the five highest loaded nodes. **(c)** latency

Comparing the source-sink latency in the two cases we got an average of $2s$ for RBR and $1.5s$ for unicast. Here too, the lower latency of the unicast is due to the fact that in the RBR case the additional receiver contention increases the transfer time per hop.

Thus, under these settings, the unicast outperforms the RBR service. We discuss in the sequel a scenario, where the situation can be different.

3) Networks with both modes enabled: We consider a wireless sensor network with many sources placed in a closer area; here aggregation of the sensor data is necessary inside the source area in order to significantly reduce the amount of data traffic (otherwise each source establishes individual paths to the sink, leading also to increased energy consumption and, thus, to a shorter lifetime of the network).

The aggregation inside the monitoring area can be realized either by constructing an aggregation tree using unicast or by employing the RBR service. The RBR mechanism allows to route data packets in this area without maintaining information about the next hop, and to aggregate without the construction and maintenance of an aggregation structure. On the other side, when using unicast a significant overhead (in terms of communication cost to spread the information about the network topology) occurs for construction and maintenance of cache tables and aggregation structures.

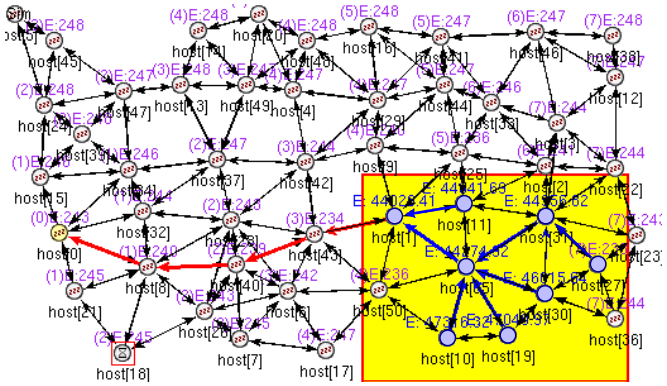


Fig. 21. Simulation network using both *RBR*-service and unicast.

For simulations we use our SNF with both unicast and RBR service enabled inside T-MAC protocol implementation and consider the 51 nodes network illustrated in Figure 21. It has 8 sources situated in the buttom-right corner and a sink (node 0) placed on the left side of the figure. Two source nodes (50 and 36) situated in the rectangle area are not equipped with the required sensors, i.e., they cannot deliver the requested data packets. We set the simulation time to 3min, the data generation interval $200ms$ and each source sends 750 packets.

We discuss two scenarios: one using unicast and the other using RBR for aggregation inside the observed area (the yellow rectangle); outside the area the aggregated data packets are always sent using unicast.

The strategy used by unicast can be, for example, a simple hop count strategy as illustrated in Figure 21 (see red line).

In this case, outside the observed area the path used to reach the sink is always the same and the energy consumption of the nodes along the path remains the same.

Of course, in order to balance the total energy consumption among the nodes between the source zone and the sink, one can use an energy-aware strategy, e.g., a routing metric based on hop count and the path's residual energy which leads to more paths to sink (see the red lines in Figure 22). This strategy redistributes the traffic load uniformly and avoids bottleneck nodes on the path to sink with less residual energy.

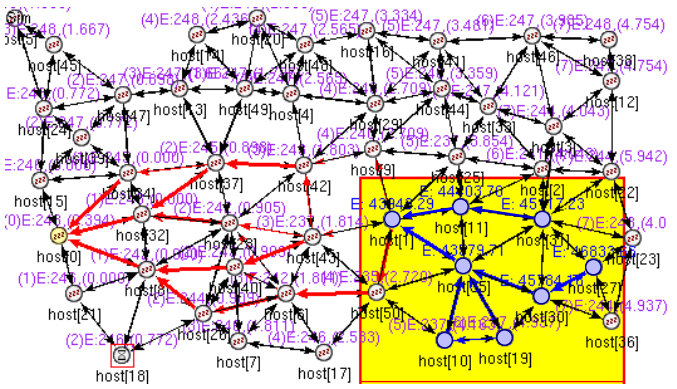


Fig. 22. Using a energy-aware routing strategy between sources and sink.

In the unicast scenario (regardless of the chosen routing metrics outside the monitoring area), in order to be able to aggregate data, an aggregation algorithm is needed to construct the aggregation structure (tree); source nodes having matching data are potential aggregators.

One possibility to construct an aggregation tree is to use the same mechanism as for flooding the interest. When an interest reaches the first source, this one initiates an aggregation interest which is flooded only inside the source region. This leads to a local greedy aggregation tree rooted at the first source.

Another alternative is to design an own aggregation protocol. When a source gets the first interest, it starts to find eligible (best) aggregation nodes in the zone. This can be achieved by sending an invitation to other sources to be aggregated, and the algorithm works as follows:

- The invitation control packet ($CAN\_I\_AGG\_YOU$) contains the sender id and information about the aggregation possibilities of the sender such as its distance to the sink, energy reserve, number of sources surrounding it (optionally their aggregator), its connectivity, etc. The invitation control packet is rebroadcasted by each source (to include farther sources).
- When a source receives the invitation it checks if it is already aggregated by another node or is an aggregator itself. If yes, the control packet is discarded. Otherwise, the receiver decides according to its local information and the received information if it is a better (closer to sink or has more sources, etc.) aggregator than the sender. When the receiver source accepts to be aggregated, it sends a confirmation control packet (YES_AGG_ME). The confirmation packet must be acknowledged (ACK_AGG) by the aggregator.
- Finally, each source knows which node is its aggregator. This information can be broadcasted to the neighbors (for more reliability) using a notify control packet ($I\_AM\_AGG\_BY$).

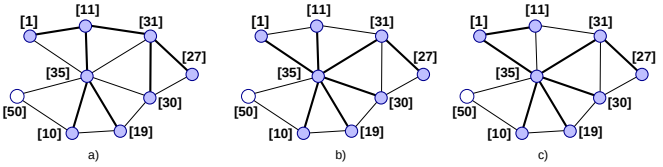


Fig. 23. (a), (b) Aggregation trees for unicast; (c) *RBR*-service

Using the latter aggregation algorithm, the resulted aggregation tree can be one of the trees illustrated in Figure 23 a) and b). In the first case we have three intermediate aggregators, nodes 11, 31 and 35 and in the second case only two, nodes 31 and 35. In the above simulation the establishment of the aggregation structure is not repeated periodically, but is realized only once.

In the RBR scenario, the aggregation is achieved without additional communication by including aggregation criteria in the definition of the priority groups, namely

- **Group 0**: Receivers having DATA packets with the same type and being closer to the sink than the sender.
- **Group 1**: Receivers having DATA packets with the same type, but farther away from the sink than the sender.
- **Group 2**: Receivers without same type of DATA, but closer to the sink.
- All receivers not belonging to one of the groups do not participate in the communication.

As a potential intermediate aggregator closer to sink gets a higher priority than an aggregator farther away, the aggregation is opportunistic. (For the considered network the aggregation flow is illustrated in Figure 23(c))

For these two scenarios, we compared the total energy consumption, the source to sink latency and the throughput. It turns out that all three performance criteria are quite similar for both scenarios.

In case of unicast, each aggregation node waits until all its children sent their data, then aggregates these and sends the result data to its parent. This increases the node's active time in the unicast scenario and reaches the same level as for the RBR (induced by its higher receiver contention).

So for both scenarios we get this time almost the same source to sink latency, throughput and total energy consumption.

Hence, for such applications the unicast does not outperform the RBR service even if the aggregation tree is constructed only once (as we considered in our simulations). Obviously, for highly dynamic networks or networks with longer activity, the aggregation structure needs to be reestablished periodically, which finally leads to weaker overall performance.

VIII. Conclusion and future work

In the present paper, we strive for more modularity at MAC layer, mainly to embed at this layer more customizable services. We supplement here the sender initiated unicast by a receiver-based contention in order to provide another perspective to the interlayer communication. The RBR service of T-MAC allows (reactive) applications, in which a sender does not know its potential destination or applications with a dynamic network topology where the construction and maintainance overhead for the cache tables and/or aggregation structures is expensive (in terms of communication cost to spread the information about the network toplogy). In such dynamic scenarios the RBR mechanism allows to route, without maintaining information about the next hop, and to aggregate, without the construction and maintenance of an aggregation structure.

The accurate energy model integrated in our simulator allows us to quantify the impact of transceiver's switch time, the *RCW* and the occurrence of collisions and their retransmissions on the energy consumption of the sensor node. The possible collisions of the *BCTS* responses and their consequences must be minimized (using transceivers with smaller switching time) for a good performance of the RBR service. Therefore, we proposed and implemented several optimizations of the RBRS. Simulation results have shown that these optimizations improve significantly the performance of the RBR. We have analyzed the performance parameters of the RBR service, namely its energy-efficiency, throughput and latency. Moreover, we compared (VII-2) the efficiency of both forwarding approaches inside T-MAC: the sender initiated one using unicast versus the receiver-based routing. For our simulation scenario it turned out that the RBR approach outperformed by unicast in terms of energy consumption, throughput and latency. Nevertheless, the RBR can be efficiently employed for opportunistic aggregation inside monitoring areas with many sources or in dynamic network scenarios (VII-3); here the routing performance of RBR and unicast are similar.

As future work we intend to build in our simulator different simulation scenarios in order to closer investigate and compare the performance of RBR versus different aggregation algorithms using unicast.

References

[1] F. Kacsó and U. Schipper, "Receiver-based routing service for t-mac protocol," in *Proc. 4th Int. Conf. on on Sensor Technologies and Applications (SENSORCOMM 2010)*. Venice/Mestre, Italy, July 2010, pp. 489–494.

[2] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated, adaptive sleeping for wireless sensor networks," *IEEE/ACM Trans. on Netw.*, vol. 12, no. 3, pp. 493–506, 2004.

[3] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. 2nd ACM Int. Conf. on Embedded Networked SenSys*. NY, USA, November 2004, pp. 95–107.

[4] T. Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proc. 1st Int. Conf. on Embedded Networked SenSys*. LA, California, USA, 2003, pp. 171–180.

[5] M. Busse, T. Hänselmann, and W. Effelsberg, "Energy-efficient forwarding schemes for wireless sensor networks," in *Proc. Int. Symp. on WoWMoM*. New York, USA, June 2006, pp. 125–133.

[6] F. Ye, G. Zhong, S. Lu, and L. Zhang, "Gradient broadcast: A robust data delivery protocol for large scale sensor networks," *Wireless Networks/Springer, The Netherlands*, vol. 11, no. 2, pp. 285–298, 2005.

[7] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, 2003.

[8] K.-W. Fan, S. Liu, and P. Sinha, "Structure-free data aggregation in sensor networks," *IEEE Trans. Mob. Comput*, vol. 6, pp. 929–942, 2007.

[9] I. Akyildiz, M. Vuran, and O. Aka, "A cross-layer protocol for wireless sensor networks," in *Proc. CISS*. Princeton, NJ, March 2006.

[10] T. Watteyne, A. Bachir, M. Dohler, D. Barthel, and I. Aug-Blum, "1-hopmac: An energy-efficient mac protocol for avoiding 1-hop neighborhood knowledge," *Sensor and Ad Hoc Communications and Networks*, vol. 2, pp. 639–644, Sept 2006.

[11] P. Skraba, H. Aghajan, and A. Bahai, "Cross-layer optimization for high density sensor networks: Distributed passive routing decisions," in *Proc. Ad-Hoc Now04*. Vancouver, July 2004, pp. 266–279.

[12] F. Kacsó and R. Wismüller, "A simulation framework for energy-aware wireless sensor network protocols," in *Proc. 18th Int. Conf. on Computer Communications and Networks (ICCCN'09), Workshop on Sensor Networks*. San Francisco, CA, USA, August 2009, pp. 1–7.

[13] J. Wong, R. Jafari, and M. Potkonjak, "Gateway placement for latency and efficient data aggregation," in *Proc. 29th Annual IEEE Int. Conf. on Local Computer Networks*, Nov 2004, pp. 490–497.

[14] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica, "A unifying link abstraction for wireless sensor networks," in *Proc. 3rd ACM Int. Conf. SenSys*, November 2005, pp. 76–89.

[15] C. Ee, R. Fonseca, S. Kim, D. Moon, A. Tavakoli, D. Culler, S. Shenker, and I. Stoica, "A modular network layer for sensornets," in *Proc. 7th Symp. OSDI*. Seattle, WA, USA, 2006, pp. 249–262.

[16] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion a scalable and robust communication paradigm for sensor networks," in *Proc. ACM MobiCom*. Boston, 2000, pp. 56–67.

[17] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. 33rd Annual Hawaii Int. Conf. on System Sciences (HICSS'00)*, 2000, pp. 3005–3014.

[18] A. Kacsó and R. Wismüller, "A framework architecture to simulate energy-aware routing protocols in wireless sensor networks," in *Proc. IASTED Int. Conf. on Sensor Networks*. Greece, 2008, pp. 77–82.

[19] M. Löbbers and D. Willkomm, *Mobility Framework for OMNeT++ (API ref.)*. http://mobility-fw.sourceforge.net: OMNeT++ Ver.3.2, 2006.

[20] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "Macaw: A media access protocol for wireless lans," in *Proc. of SIGCOMM Conf.* London, UK, September 1994, pp. 212–225.