RD2: Resilient Dynamic Desynchronization for TDMA over Lossy Networks

Hinterhofer, Thomas ; Schwefel, Hans-Peter; Tomic, Slobodanka

Published in:
31st IEEE International Symposium on Reliable Distributed Systems (SRDS), 2012

DOI (link to publication from Publisher):
10.1109/SRDS.2012.57

Publication date:
2012

Document Version
Accepted author manuscript, peer reviewed version

RD^2: Resilient Dynamic Desynchronization for TDMA over Lossy Networks

Thomas Hinterhofer*, Hans-Peter Schwefel*† and Slobodanka Tomic*
*FTW Telecommunications Research Center Vienna, Austria
†Department of Electronic Systems, Aalborg University, Denmark
Email: [hinterhofer,schwefel,tomic]@ftw.at

***Abstract*—We present a distributed TDMA negotiation approach for single-hop ad-hoc network communication. It is distributed, resilient to arbitrary transient packet loss and defines a non-overlapping TDMA schedule without the need of global time synchronization. A participating node can dynamically request a fraction of the static TDMA period T. It will receive its fraction if enough time resources are available. In any case, every node can request and will receive at least a fair fraction of size $\frac{1}{N}$. Due to its resilience to arbitrary transient packet loss, the algorithm is well suited for lossy networks like found in wireless communications. Our approach is designed to work in highly dynamic scenarios efficiently. We will show, that it defines a dynamic non-overlapping TDMA schedule even at high packet loss rates. The performance of the TDMA negotiation is analyzed by simulation and compared to results of related work.**

I. Introduction

Wireless communication has already replaced wired communication systems in various fields of applications. Low price, easy installation and good performance are reasons for the enormous success of 802.11 in the private domain. 802.15.4/ZigBee is increasingly used in wireless sensing and home automation environments [1] where fast and easy installation and low energy consumption is more important than communication performance. Nevertheless, 802.11 and 802.15.4 are rarely used in the industrial communication and automation domain because of tight requirements regarding reliability and timely delivery of data. The unpredictability of the wireless channel and the CSMA-CA MAC technique used by 802.11 and 802.15.4 [2][3] causes additional effort to satisfy these communication requirements. CSMA-CA is based on a random selection of backoff times, which can result in long transmission times especially at high traffic loads. If the traffic can be classified in different classes of priority, 802.11e [4] can be used. By contrast, the point coordination function (PCF) [5] of 802.11 implements a centralized polling mechanism to coordinate transmission times. In order to avoid this single point of failure, a cooperative usage of the wireless channel at a higher layer is needed if the usage of off-the-shelf hardware is required. Our *Resilient Distributed Desynchronization* (RD^2) algorithm reliably prevents two nodes from transmitting at the same time. As a consequence, MAC collisions and the selection of random backoff times are prevented, which normally can cause long transmission times.

This paper is organized as follows. In Section II, we will present some related work regarding wireless TDMA negotiation and desynchronization. Section III introduces our RD^2 approach in detail. The MATLAB simulation of RD^2 can be found in Section IV. The desynchronization and TDMA negotiation performances are investigated as well as the fairness properties regarding wireless channel access. Finally, we conclude the paper in Section V.

II. Related Work

Most TDMA negotiation approaches are based on underlying clock synchronization. An example can be found in [6], where a global notion of time is used to establish a static, a priori defined TDMA scheduling. By contrast, STDMA [7] provides dynamic TDMA slot allocation. After listening during an initialization phase, every node selects an available time slot. If no free time slot is available, the node selects the time slot of the farthest away neighbor. By this, access to the communication medium is guaranteed. However, global time synchronization is required for STDMA. Our work is based on the DESYNC algorithm first presented by Degesys et al. in [8]. They present a distributed algorithm for fully connected single-hop network topologies to organize a non-overlapping TDMA schedule without the need of global time synchronization. In [9] and [10], the DESYNC approach is extended to the multi-hop network domain. Nevertheless, none of the algorithms handle packet loss by design, which is a fundamental problem for the underlying desynchronization algorithm. As a consequence, they cannot guarantee a non-overlapping TDMA schedule over lossy networks.

III. Resilient Dynamic Desynchronization

With help of desynchronization, periodic events can be coordinated to not happen at the same point in time. An important example is wireless communication of nodes operating in the same interference domain. For desynchronization, time is divided into rounds of length T. If S is the set of participating nodes and $N = |S|$, every node $i \in S$ fires a beacon B_i^r at time $t_{B_i^r}$ every round r. Hence, the last node of round r and the first node of round $r+1$ are neighbors in time. Throughout the rest of this work, $i-1$ represents node i's previous neighbor in time and $i+1$ its next neighbor, respectively. The nodes are perfectly desynchronized if

$$\forall i \in S : t_{B_i} - t_{B_{i-1}} = \frac{T}{N}. \quad (1)$$

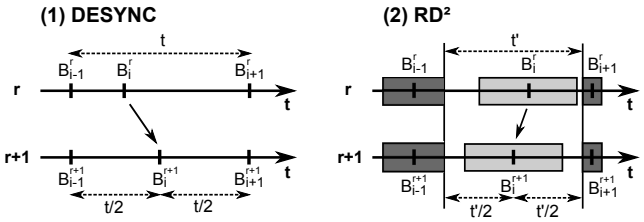

Fig. 1: DESYNC and RD^2 beacon time calculation

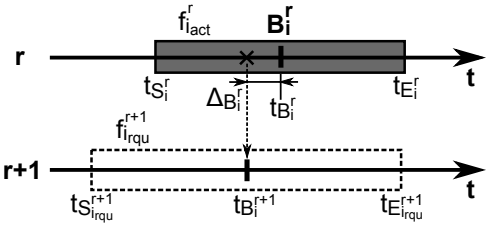

Fig. 2: virtual beaconing

Based on the desynchronized beacons, it is easy to define a non-overlapping TDMA schedule with equal slot sizes in every desynchronization round r. Our *Resilient Dynamic Desynchronization* (RD^2) approach allows every node to dynamically request a fraction of the TDMA round time T. This request can be controlled by upper layers or it is dynamically adjusted based on the outgoing buffer status. Nevertheless, RD^2 prevents nodes from getting slot sizes, which would result in unfair communication channel usage. In the following sections, we will provide the definition of fairness as well as algorithmic details.

A. RD^2 Basics and Assumptions

The basic principles of the DESYNC algorithm presented in [8] and our RD^2 approach are depicted in Figure 1.

A node i implementing DESYNC listens to round r beacons of the previous and the next neighbor in time. In the following round $r+1$, i is sending its beacon B_i^{r+1} in the middle of B_{i-1}^{r+1} and B_{i+1}^{r+1}. By contrast, RD^2 sets its round $r+1$ beacon to be in the middle of the slot end of the previous node $i-1$ and the slot start of the next node $i+1$. This enables variable TDMA slot sizes. Furthermore, the intra slot times to the neighboring slots are equalized and maximized.

Before continuing with the introduction of RD^2, we will describe the assumptions the approach is based on:

1) We assume an **ordered set S of N nodes** with **unique IDs** and each node $i \in S$ knows the ID of its previous $(i-1)$ and its next neighbor $(i+1)$.
2) A fixed set of participating nodes is assumed. **No node enters or exits** the network.
3) Every participating node has a **non-malicious behavior**, i.e. every node correctly executes the RD^2 algorithm.
4) The **communication network connects** at least every node $i \in S$ with its **neighbors** $i-1$ and $i+1$ with a direct link (single-hop). All nodes can share the same interference domain. Examples are distributed control applications like found in industrial communication or in autonomous robotic fleets.
5) We only care about arbitrary transient packet loss. Hence, we assume that there are **no permanent node failures or permanent packet loss**.
6) The maximum **transmission delay** of a packet is assumed to be **bounded**, known and small compared to $\frac{T}{N}$. Without this assumption it is not possible to guarantee a non-overlapping TDMA negotiation. d_{max} represents the maximum transmission delay and d_{min} the minimum, respectively.
7) The maximum **clock drift** between time sources of individual nodes is **bounded** and known. For simplicity, we assume the clock drift to be zero throughout the rest of this work. This is acceptable, as clock drift adds no accumulated error to the proposed TDMA negotiation.

B. RD^2 Virtual Beaconing

RD^2 introduces *virtual beaconing* (c.f. Figure 2). Every physical beacon B_i^r contains a fire offset $\Delta_{B_i^r}$, which defines the exact beacon time $t_{B_i^{r+1}}$ of the following round (2).

$$t_{B_i^{r+1}} = t_{B_i^r} + \Delta_{B_i^r} + T \tag{2}$$

Because of this, every node i knows the round $r+1$ beacon times of its neighbors $i-1$ and $i+1$ in advance. This enables an energy efficient implementation of the algorithm, as a node can go into sleep mode and wakes up shortly before the expected transmission times of B_{i-1}^{r+1} and B_{i+1}^{r+1}.

We use Figure 2, to introduce some terminology used throughout this work:

$t_{S_i^r}$	slot start time of node i in round r.
$t_{E_i^r}$	slot end time of node i in round r.
$f_{i_{act}}^r$	actual TDMA fraction of node i in round r.
$f_{i_{rqu}}^{r+1}$	requested fraction of node i for round $r+1$. The center of the requested fraction is $t_{B_i^{r+1}}$.
$t_{S_{i_{rqu}}^{r+1}}$	requested slot start time of node i for round $r+1$.
$t_{E_{i_{rqu}}^{r+1}}$	requested slot end time of node i for round $r+1$.

In addition to the fire offset $\Delta_{B_i^r}$, every beacon B_i^r contains $f_{i_{act}}^r$, $f_{i_{rqu}}^{r+1}$, $f_{i-1_{rqu}}^{r+1}$ and $f_{i+1_{rqu}}^r$. Of special interest are the fraction requests of the previous and the next neighbor. If a node j did not receive the last beacon of its neighbor, it will set the according fraction request ($f_{j-1_{rqu}}^{r+1}$ or $f_{j+1_{rqu}}^r$) to -1. This allows a node i to check in the received beacons B_{i-1}^r and B_{i+1}^{r-1} whether the neighbors received its previous fraction request (ACK) or not (NACK). We will use this acknowledgment mechanism later in Section III-D2.

The fire offset encapsulated in a beacon B_i^r is always limited by $t_{S_i^r}$ and $t_{E_i^r}$ of the sending node i corrected by the worst case transmission delays (c.f. Equation 3).

$$t_{S_i^r} - d_{min} < t_{B_i^r} + \Delta_{B_i^r} < t_{E_i^r} - d_{max} \tag{3}$$

The reason is to provide a non-overlapping TDMA schedule also in case of arbitrary transient beacon loss as will be

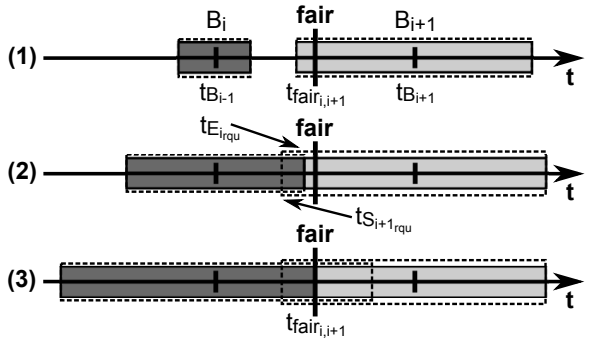

Fig. 3: neighborhood fairness

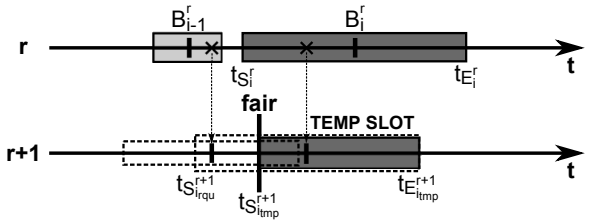

Fig. 4: temporary slot calculation

explained later in Section III-D1. A consequence of (3) is that a node i must always have an actual fraction $f^r_{i_{act}}$ of at least $2d_{max}$. Otherwise, it is not able to set its beacon offset to a value ≥ 0, which influences the performance and results in a non-static behaviour. Therefore, the requested fraction has to be limited by a configurable minimum f_{min}.

For simplicity, we assume the transmission delay to be zero throughout the rest of this work. Adding a bounded transmission delay means that for every received B^r_{i-1} the minimal transmission time d_{min} is assumed and for every received B^r_{i+1} the maximal transmission time d_{max}, respectively.

C. RD^2 Fairness

A main goal of our approach is to guarantee fair access to the communication channel. Fairness can be defined between neighboring nodes and for the global network. The fairness mechanism for neighboring nodes (F_N) is depicted in Figure 3. The *fairness boundary* time $t_{fair_{i,i+1}}$ between two neighboring nodes i and $i+1$ is defined to be in the middle of their beacons B_i and B_{i+1}.

In case (1), node i requested a slot end and node $i+1$ a slot start, which are not overlapping. As a result, both nodes get their requested slot boundaries. In case (2), $t_{E_{i_{rqu}}}$ and $t_{S_{i+1_{rqu}}}$ are overlapping. As the overlapping occurs before $t_{fair_{i,i+1}}$, both slot boundaries are set to $t_{E_{i_{rqu}}}$. Finally, in case (3) both nodes request times beyond the fairness boundary. Thus, both slot boundaries are set to $t_{fair_{i,i+1}}$.

D. RD^2 Two-Phase Slot Calculation

We will now describe the principle how every node defines its upcoming TDMA slot. In order to be resilient to arbitrary transient beacon loss, a node i can only extend its TDMA slot boundaries if the request was acknowledged by the according neighbor or if it is known that there are sufficient time resources. Hence, every node does the calculation of its round $r+1$ slot boundaries in two phases:

1) Dissemination Phase: A temporary slot is calculated by a node i, when it sends its beacon B^r_i. This has to be done as the upcoming beacons B^r_{i+1} and B^{r+1}_{i-1} of the neighbors may not be received by node i, which would result in no definition of the slot boundaries. Regardless of the successful dissemination of node i's beacon B^r_i, the temporary slot will never overlap with its neighboring TDMA slots. The round $r+1$ temporary slot is always a subslot of the round r slot:

$$t_{S^r_i} + T \leq t_{S^{r+1}_{i_{tmp}}} < t_{E^{r+1}_{i_{tmp}}} \leq t_{E^r_i} + T \tag{4}$$

As can be seen in Figure 4, the temporary slot end $t_{E^{r+1}_{i_{tmp}}}$ only becomes smaller if node i requested an earlier round $r+1$ slot end by itself. The temporary slot start $t_{S^{r+1}_{i_{tmp}}}$ becomes greater if node i requested a later round $r+1$ slot start by itself or if the fairness boundary limits the slot start (c.f. Figure 4).

We already mentioned in Section III-B, that the fire offset $\Delta_{B^r_i}$ encapsulated in a beacon B^r_i is always limited by $t_{S^r_i}$ and $t_{E^r_i}$ (c.f. Equation 3). This guarantees that B^{r+1}_i can be transmitted within the round $r+1$ temporary slot.

2) Acknowledgment Phase: As already mentioned, slot boundaries can only become greater if an acknowledgment of the according neighbor (c.f. Section III-B) was received or if sufficient time resources are available. As can be seen in Figure 5, node i requests a fraction $f^{r+1}_{i_{rqu}}$ which results in a requested slot start $t_{S^{r+1}_{i_{rqu}}} < t_{S^r_i} + T$ and a requested slot end $t_{E^{r+1}_{i_{rqu}}} > t_{E^r_i} + T$. Assume, that B^r_i is received by node $i+1$, but not by node $i-1$. As a consequence, $i-1$ will not acknowledge $f^{r+1}_{i_{rqu}}$ in B^{r+1}_{i-1}. Nevertheless, i can set its slot start to $t_{E^{r+1}_{i-1_{rqu}}}$ as it is known that $i-1$ does not need the channel after this point. By contrast, node $i+1$ acknowledges $f^{r+1}_{i_{rqu}}$ in B^r_{i+1} and $t_{E^{r+1}_i}$ is set using the already introduced fair slot calculation.

Equations 5 and 6 define the acknowledged calculation of round $r+1$ slot boundaries in general.

$$t_{S^{r+1}_i} = \begin{cases} \text{keep temporary slot start,} & B^{r+1}_{i-1} \text{ lost,} \\ \text{update slot start,} & \text{NACK by } B^{r+1}_{i-1}, \\ \text{calculate fair slot start,} & \text{ACK by } B^{r+1}_{i-1}. \end{cases} \tag{5}$$

$$t_{E^{r+1}_i} = \begin{cases} \text{keep temporary slot end,} & B^r_{i+1} \text{ lost,} \\ \text{update slot end,} & \text{NACK by } B^r_{i+1}, \\ \text{calculate fair slot end,} & \text{ACK by } B^r_{i+1}. \end{cases} \tag{6}$$

E. RD^2 Beacon Pushing

As mentioned in Section III-C, guaranteeing fair access to the communication channel is a main goal of our approach. We already defined the fairness property between neighboring

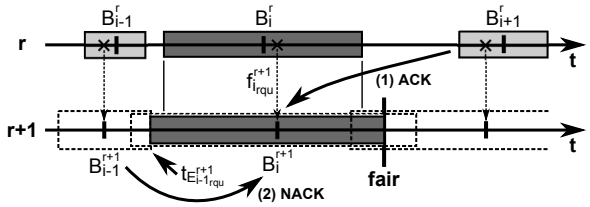

Fig. 5: acknowledged slot calculation

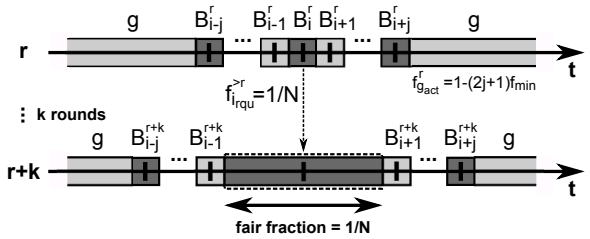

Fig. 6: static unfair schedule

nodes. In addition, fairness has to be reached eventually also for the global TDMA schedule. For the definition of global fairness we need the notation f^r_{SUM} representing the sum of all round r fraction requests in the network (c.f. Equation 7).

$$f^r_{SUM} = \sum_{i=1}^{N} f^r_{i_{rqu}} \tag{7}$$

The property of fulfilling global fairness (F_G) in round r is defined in Equation 8. In the third case, it is sufficient, that a node gets at least a fair fraction of $\frac{1}{N}$.

$$\forall i \in S : f^r_{i_{act}} = \begin{cases} f^r_{i_{rqu}}, & f^r_{i_{rqu}} \leq \frac{1}{N}, \\ f^r_{i_{rqu}}, & f^r_{i_{rqu}} > \frac{1}{N} \wedge f^r_{SUM} \leq 1, \\ \geq \frac{1}{N}, & f^r_{i_{rqu}} > \frac{1}{N} \wedge f^r_{SUM} > 1. \end{cases} \tag{8}$$

There exist some static TDMA schedules preventing nodes from getting fair access to the communication channel. A TDMA schedule is static at round r if

$$\forall i \in S, \forall r' \geq r : \Delta_{B^{r'}_i} = 0. \tag{9}$$

Assume the TDMA schedule depicted in Figure 6.

At round r, node i has a minimal actual fraction $f^r_{i_{act}} = f^r_{i_{rqu}} = f_{min}$ and is enclosed by $2j$ nodes with similar fraction requests. In addition, there is a node g requesting a slot bigger than the remaining time resources of the TDMA round. Node i requests for round $r+1$ the fair fraction $\frac{1}{N}$. Nevertheless, the neighboring nodes $i-1$ and $i+1$ will not move, because their fire offset is calculated based on the previous slot end and the next slot start. These do not change because of the higher fraction request of node i.

As a consequence, we introduce a *beacon pushing* mechanism to resolve static TDMA schedules, which do not fulfill the global fairness property F_G. Every node i checks its actual fire offset $\Delta_{B^r_i}$. If it is below a configurable threshold, it compares $f^r_{i-1_{rqu}}$ and $f^{r-1}_{i+1_{rqu}}$ to $f^r_{i-1_{act}}$ and $f^{r-1}_{i+1_{act}}$ to check whether the neighbors have fair access to the communication channel. If not, the according neighboring node will impair the fire offset of i. This is done in a way, that i's round $r+1$ beacon is pushed away to provide more time resources for the unfair treated node. If both neighboring nodes would push node i, node $i+1$ wins by definition.

The new fire offset is limited like usual by Equation 3. Beacon pushing will be done until the neighboring node has a fair actual fraction. This leads finally to a static TDMA schedule, which fulfills our fairness properties F_N and F_G.

F. RD^2 Pseudocode

The pseudocode of RD^2 can be found in Algorithm 1.

Algorithm 1 RD^2

```
1:  procedure onFiringTimerExpire
2:      reset ack_p, ack_n
3:      calculate temporary slot for round r+1          ▷ Eq. 4
4:      broadcast beacon(myId,Δ, rcv_p, rcv_n)
5:      reset rcv_p, rcv_n
6:      set fire time for round r+1
7:  end procedure
8:  procedure onRcvBeacon(t_rx, id_rx, Δ_rx, cnf_p, cnf_n)
9:      if id_rx = next then
10:         set rcv_n
11:         if cnf_p then          ▷ confirmation from next node?
12:             set ack_n
13:         end if
14:         calculate/update slot end for round r+1     ▷ Eq. 6
15:     end if
16:     if id_rx = previous then
17:         set rcv_p
18:         if cnf_n then ▷ confirmation from previous node?
19:             set ack_p
20:         end if
21:         calculate/update slot start for round r     ▷ Eq. 5
22:         if ack_p ∧ ack_n then
23:             calculate fire offset Δ
24:             check beacon pushing
25:             limit Δ
26:         end if
27:     end if
28: end procedure
```

IV. MATLAB Simulations

We implemented the presented algorithm in MATLAB to study its behavior and performance. In the simulation model, the *Beacon Loss Rate* (BLR) and the dynamics of the individual fraction requests can be configured. Specific scenarios were simulated with variable beacon loss rates and a TDMA round time of $T = 100ms$. We will show, that RD^2 is capable to work reliably even at high loss rates.

An example schedule of the RD^2 algorithm can be found in Figure 7. Every node i changes its fraction request $f_{i_{rqu}}$

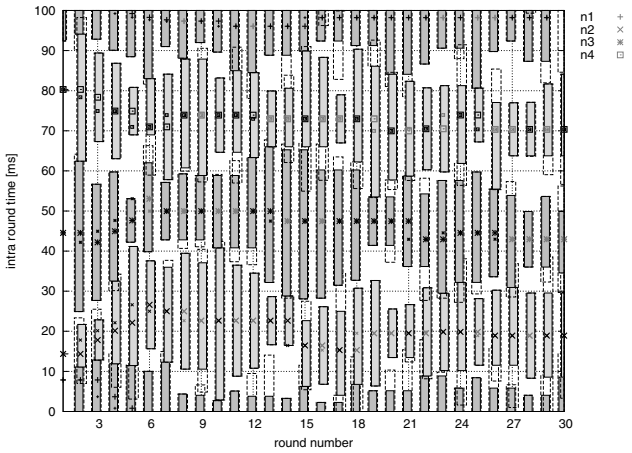

Fig. 7: RD^2 example schedule

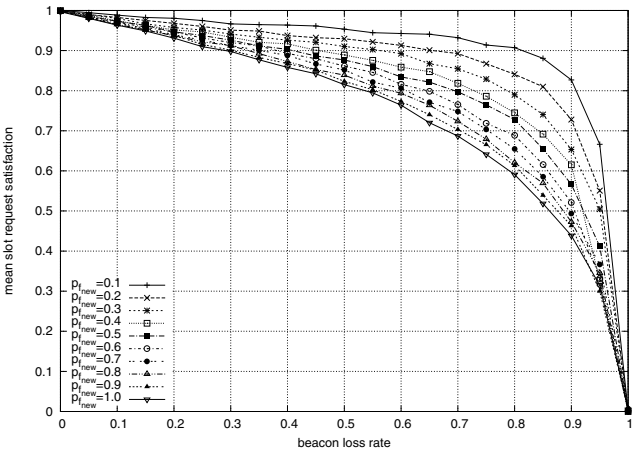

Fig. 8: mean slot request satisfaction

every round. The beacon loss rate is configured to 0.3. The dashed boxes indicate the requested slots, while the filled ones represent the actual slot. Sent beacons are drawn black and lost beacons gray. In this figure, lost beacons are lost for the previous and the next node. For the numerical simulation results, beacons can also be lost for only one neighbor.

A. RD^2 Slot Request Satisfaction

To analyse the performance of RD^2 we look at the mean slot *Request Satisfaction* $\overline{RS_i}$ of a node i, which is defined by (10). R represents the number of simulated rounds.

$$\overline{RS_i} = \frac{\sum_{r=1}^{R} \frac{f^r_{i_{act}}}{f^r_{i_{rqu}}}}{R} \tag{10}$$

In Figure 8, a simulation ($R = 10000$) of RD^2 is depicted. Four nodes are independently and randomly requesting fractions from a uniform distribution between 0 and 0.25. The probability, that a node updates its request in the actual round is defined by $p_{f_{new}}$. Using a normally distributed assumption on the estimator $\overline{RS_i}$, the 95% confidence intervals of Figure 8 results are in the worst case of width 0.013.

As can be seen in Figure 8, RD^2 has even at high beacon loss rates of > 0.6 $\overline{RS_i}$ values of 0.8 and more if $p_{f_{new}} \leq 0.7$. For a beacon loss rate of 0.3, $\overline{RS_i}$ is > 0.9 even at a fraction update probability $p_{f_{new}}$ of 1.

B. RD^2 Convergence

In order to investigate the convergence of RD^2, simulations were done on more static *Fraction Request Patterns* (FRP). FRPs are defined by a matrix of size $U \times (N + 1)$. The first column represents the round number r of a fraction request update. The remaining columns represent the fraction requests of the individual nodes. U is the number of fraction request updates and every row stands for a specific update. For the following simulations for $N = 4$ nodes, the FRP contains two rows for updates in round 1 and round 15 (c.f. Equation 11). We will concentrate on the dynamics caused by this fraction request updates.

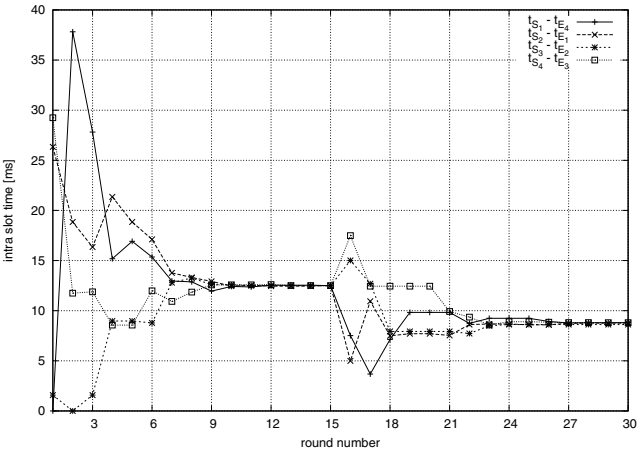

Fig. 9: intra slot time convergence, requests updated in rounds 1 and 15, beacon loss rate 0.3

$$FRP = \begin{pmatrix} 1 & 0.1 & 0.05 & 0.15 & 0.2 \\ 15 & 0.2 & 0.1 & 0.05 & 0.3 \end{pmatrix} \tag{11}$$

In Figure 9, the *intra slot time* (IST) between neighboring nodes is depicted. It can be seen that with a beacon loss rate of 0.3 the algorithm reaches equalized IST after every fraction request update ($12.5ms$ for $r > 11$ and $8.75ms$ for $r > 26$).

C. Illustration and Analysis of Beacon Pushing

Next, the beacon pushing mechanism needed for global fairness is simulated. We created a static TDMA schedule as described in Figure 6. For $N = 4$, this is the worst case for the encapsulated node 3. The FRP is defined as follows:

$$FRP = \begin{pmatrix} 1 & 1.0 & 0.1 & 0.1 & 0.1 \\ 15 & 1.0 & 0.1 & 0.25 & 0.1 \end{pmatrix} \tag{12}$$

As a result, for $r \leq 15$ nodes $2 - 4$ get their requested fraction (0.1) and node 1 is allowed to take the rest of the time resources (0.7). In round 15, node 3 is requesting its fair fraction. As can be seen in Figure 10, node 3 eventually gets its fair fraction from resources held originally by node 1.

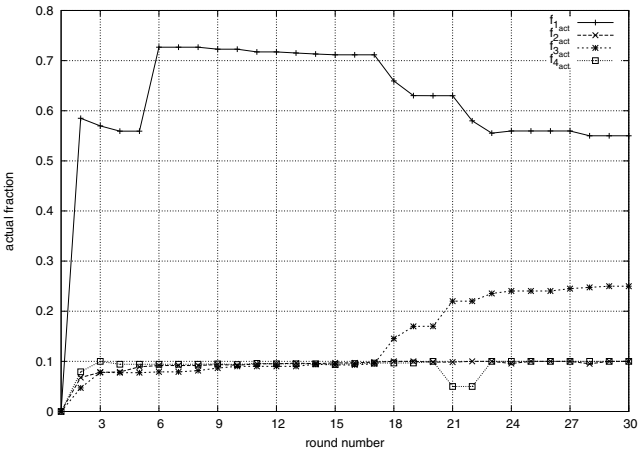

Fig. 10: beacon pushing example, requests updated in rounds 1 and 15, beacon loss rate 0.3

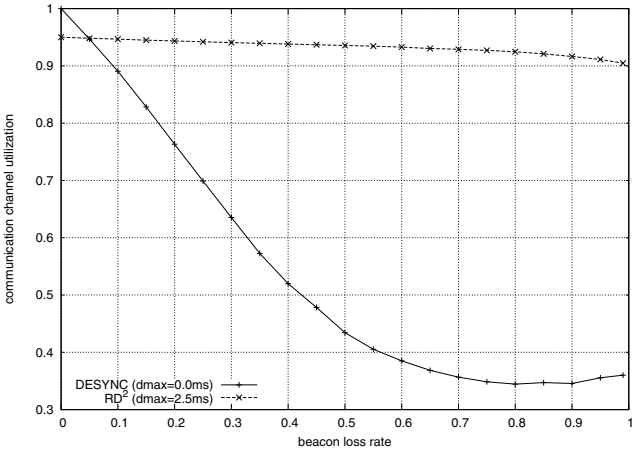

Fig. 11: communication channel utilization (N=4, T=100ms)

D. DESYNC vs. RD^2

A comparison of the communication channel utilizations of DESYNC and RD^2 is depicted in Figure 11. Utilization is defined as the fraction of T, in which any node has exclusive access to the communication channel. Hence, overlapping time slots and unused time resources decrease the utilization. All simulations were started with $N = 4$, $T = 100ms$ and a beacon loss rate of 0. After a static desynchronized state was reached, the according beacon loss rate was set to analyse its impact. DESYNC has a good utilization only at very low beacon loss rates (< 0.1) because higher loss rates cause significant time slot overlappings. An ideal utilization of 1 at a loss rate of 0 is only reached because DESYNC was simulated with zero beacon transmission delay. RD^2 nodes were configured to request a static fraction of 1.00, which results in time slots of equal size. The maximum beacon transmission time d_{max} was set to $2.5ms$. To be resilient to the transmission delay, RD^2 leaves guard margins between consecutive time slots causing a decrease of the utilization. Nevertheless, RD^2 also works efficiently at high loss rates.

V. Conclusion

We presented a distributed algorithm capable of defining a non-overlapping TDMA schedule in the presence of arbitrary transient packet loss without the need for global time synchronization. The TDMA slot sizes are not centrally controlled but can be requested by each node individually. We showed by simulation that a node can get fair access to the communication medium at any time. This means, that it will eventually get a time slot of size $\frac{T}{N}$ if requested. Furthermore, we verified the performance of the resilient TDMA negotiation. Even in highly dynamic scenarios at packet loss rates of 30%, over 90% of the requested TDMA slot times are acknowledged assuming a fair behaviour of all participating nodes.

Our next step towards a real implementation is the definition of procedures for node entrance and exit. For the uncontrolled exit of nodes, a diagnosis mechanism which decides whether lost beacons are caused by transient beacon loss or by a permanent node failure is needed. After simulation of the extended algorithm, it will be implemented on 802.15.4 and 802.11 testbeds in order to compare real world results to simulation results.

Acknowledgment

This paper is supported within the European Research Project RHEA. The Telecommunications Research Center Vienna (FTW) is supported by the Austrian government and the City of Vienna within the competence center program COMET.

References

[1] M. Osipov, "Home automation with zigbee," in *Proceedings of the 8th international conference, NEW2AN and 1st Russian Conference on Smart Spaces, ruSMART on Next Generation Teletraffic and Wired / Wireless Advanced Networking*, ser. NEW2AN '08 / ruSMART '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 263–270.

[2] E. Ziouva and T. Antonakopoulos, "Csma/ca performance under high traffic conditions: throughput and delay analysis," *Computer Communications*, vol. 25, no. 3, pp. 313 – 321, 2002.

[3] T.-S. Ho and K.-C. Chen, "Performance analysis of ieee 802.11 csma/ca medium access control protocol," in *Personal, Indoor and Mobile Radio Communications, 1996. PIMRC'96., Seventh IEEE International Symposium on*, vol. 2, oct 1996, pp. 407 –411 vol.2.

[4] S. Mangold, S. Choi, G. Hiertz, O. Klein, and B. Walke, "Analysis of ieee 802.11e for qos support in wireless lans," *Wireless Communications, IEEE*, vol. 10, no. 6, pp. 40 – 50, dec. 2003.

[5] C. Coutras, S. Gupta, and N. B. Shroff, "Scheduling of real-time traffic in ieee 802.11 wireless lans," *Wirel. Netw.*, vol. 6, no. 6, pp. 457–466, Dec. 2000. [Online]. Available: http://dx.doi.org/10.1023/A:1019130823870

[6] R. Leidenfrost and W. Elmenreich, "Establishing wireless time-triggered communication using a firefly clock synchronization approach," in *Intelligent Solutions in Embedded Systems, 2008 International Workshop on*, july 2008, pp. 1 –18.

[7] K. Bilstrup, E. Uhlemann, E. Strom, and U. Bilstrup, "Evaluation of the ieee 802.11p mac method for vehicle-to-vehicle communication," in *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, sept. 2008, pp. 1 –5.

[8] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "Desync: Self-organizing desynchronization and tdma on wireless sensor networks," in *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, april 2007, pp. 11 –20.

[9] J. Degesys and R. Nagpal, "Towards desynchronization of multi-hop topologies," in *Self-Adaptive and Self-Organizing Systems, 2008. SASO '08. Second IEEE International Conference on*, oct. 2008, pp. 129 –138.

[10] C. Muehlberger and R. Kolla, "Extended desynchronization for multi-hop topologies," Institut für Informatik, Technical Report 460, Juli 2009.