EAI Endorsed Transactions
on Mobile Communications and Applications
Research Article ICST.ORG

Route-Over Forwarding Techniques in a 6LoWPAN

Andreas Weigel[1,*], Martin Ringwelski[2], Volker Turau[1], Andreas Timm-Giel[2]

[1]Institute of Telematics, Hamburg University of Technology
[2]Institute of Communication Networks, Hamburg University of Technology

Abstract

6LoWPAN plays a major role within the protocol stack for the future Internet of Things. Its fragmentation mechanism enables transport of IPv6 datagrams with the required minimum MTU of 1280 bytes over 802.15.4-based networks. With the goal of a fully standardized WSN protocol stack currently necessitating a route-over approach, i.e., routing at the IP-layer, there are two main choices for any 6LoWPAN implementation with regard to datagram fragmentation: Hop-by-hop assembly or a cross-layered direct mode, which forwards individual 6LoWPAN fragments before the whole datagram has arrived. In addition to these two straightforward approaches, we propose enhancements based on adaptive rate-restriction for the direct forwarding and a retry control for both modes to reduce the number of losses of larger datagrams. An evaluation of these modes in a simulation environment and a hardware testbed indicate that the proposed enhancements can considerably improve PRR and latency within 6LoWPAN networks.

Received on 28 February 2014; accepted on 16 September 2014; published on 28 December 2014

Keywords: 6LoWPAN, fragmentation, 802.15.4, CometOS, forwarding, route-over, wireless sensor networks

doi:10.4108/mca.2.5.e5

1. Introduction

Wireless sensor networks (WSNs) have a broad field of possible applications, starting from smart homes via monitoring of industrial plants, agricultural fields and personal health through to smart metering. WSNs are typically characterized by nodes with only constrained resources in terms of memory, computation power and available energy and by wireless links which often exhibit lossy and transient behavior. Until recently, these networks usually employed proprietary protocols and therefore off-the-shelf solutions were either not available or not interoperable.

The vision of the "Internet of Things" aims at providing each and every sensor with its own IPv6 address to make it accessible via proven and established standard protocols. This idea has given rise to the development of a standardized protocol, Transmission of IPv6 Packets over IEEE 802.15.4 Networks (RFC 4944 [1]; 6LoWPAN), which enables the use of IPv6 with the link layer protocol IEEE 802.15.4 [2]. The routing protocol for low power and lossy networks (RPL [3]) and its recent acceptance as a proposed standard as well as the constrained application protocol (CoAP [4]) complement the development towards a completely standardized IPv6 protocol stack for WSNs.

The 802.15.4 standard offers physical- and MAC-layers for low power wireless personal area networks (LoWPAN). While the maximum PHY frame payload of those networks is only 127 bytes, IPv6 demands a maximum transmission unit (MTU) of at least 1280 bytes. Additionally, the MAC header can be up to 25 bytes large and a possible AES-CCM-128 encryption might use another 21 bytes, leaving 81 bytes for data payload. By using IPv6 with UDP, another 40 bytes are occupied by the IP header and 8 bytes by the UDP header, decreasing the maximum payload in a frame to 33 bytes. 6LoWPAN offers an intermediate layer between the IP- and the data link layer to overcome these issues. It defines compression algorithms for IPv6 headers and a fragmentation mechanism for larger IPv6 datagrams to be transportable within 802.15.4 MAC frames.

With regard to routing in a multi-hop wireless network, 6LoWPAN specifies two possibilities: mesh-under and route-over. With mesh-under, routing decisions are made at the adaption layer by some not specified routing protocol; the entire 6LoWPAN network appears to the IP layer as a single hop network. Following an approach with completely standardized protocols, we are only concerned with route-over, where routing decisions are made by a routing protocol at the IP layer, e.g., RPL.

Applying strict separation of layers with route-over, each node along a path then needs to buffer incoming fragments in order to reassemble the complete datagram. If the arriving packet is in transit to another node it has to be reassembled, handed to

*Corresponding author. andreas.weigel@tuhh.de

the IP layer for routing decisions and again has to be fragmented and sent to the next node. During the whole process, buffer space has to be reserved for the whole datagram. Considering the resource limitation of WSN hardware, where a buffer is likely to be not much larger than the MTU, this may necessitate dropping additional incoming datagrams for which no buffer space is left. We call this mode **Assembly Mode** in the remainder of the paper.

This is a known issue and the informational implementation guidelines [5] for 6LoWPAN recommend the use of a virtual fragmentation buffer, which immediately forwards fragments that are just in transit to the next hop and only stores information necessary to identify and dispatch the following fragments. While such a direct forwarding scheme may overcome the buffering issue and even decrease the latency on longer paths by enabling pipelining of fragments, it is also likely to cause more collisions on the channel due to the hidden terminal problem. For this mode of operation we use the term **Direct Mode**. Fig. 1 shows the message flows for the Assembly Mode and the Direct Mode and illustrates the potential for pipelining as well as the risk of collisions.

Considering that lost fragments will inevitably lead to lost datagrams, the forwarding strategy has a tremendous impact on the performance within a 6LoWPAN-based WSN. Therefore, we evaluated the basic schemes and additionally propose rate-restriction mechanisms to prevent performance degradation using the Direct Mode and a retry-control mechanism to prevent the loss of nearly completely transmitted datagrams. These different modes are described in more detail in Section 3. An overview about existing research concerning 6LoWPAN fragmentation strategies is given in Section 2. Sections 4 and 5 provide information about the used simulation and testbed scenarios and the results of the experiments, respectively. Section 6 concludes this work.

2. Related Work

Different forwarding techniques for 6LoWPAN for IPv6 datagrams without and with fragmentation were evaluated by Ludovici et al. [6]. They analyzed end-to-end delay and loss-rate of a single sender for two route-over [1] and two mesh-under schemes for a line topology of up to five TelosB nodes, yielding a maximum network diameter of 4. One main result of their studies was the dramatically higher reliability of the route-over scheme compared to mesh-under and enhanced route-over, up to a datagram size at which maximum buffer capacity is reached and datagrams have to be dropped due to the lack of buffer space. On the other hand, end-to-end delay has been observed to be better for the three non-reassembling schemes.

A similar approach was adopted by Bhunia et al. [7]. For a similar setup, they analyzed the end-to-end delay and loss rate for a single sender node in a small testbed with a line topology. Their observations are in line with those of [6].

In a draft of the IETF working group "Routing over low power and Lossy networks"[2], Thubert and Hui [8] described an extension to RFC4944 which adds negative acknowledgements and fragment recovery mechanism to 6LoWPAN. By means of recovery from individual fragment losses, the loss (and potentially congestion-causing upper layer retransmission) of whole datagrams is meant to be prevented. This approach was analytically evaluated by Ayadi et al. in [9] with the conclusion that it can reduce the overall number of bits sent by reducing the necessary number of retransmissions of whole datagrams. While certainly worth investigating, this mechanism can be seen as an orthogonal approach to the ones proposed by us and will not be further evaluated within this paper.

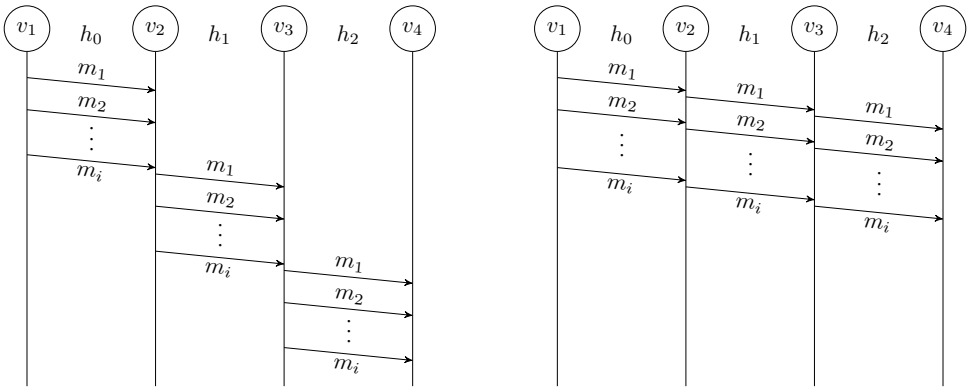
Ayadi et al. also developed an analytical model for 6LoWPAN transmissions with the aim to determine the optimum TCP segment size ([10]). Their model does not consider senders which give up on a datagram after not receiving an acknowledgement from their next link-layer hop and also does not take into account any direct forwarding modes.

Wang et al. [11] proposed a technique for mesh-under routing in 6LoWPANs, which reassembles packets at some intermediate nodes. Evaluating route-over, mesh-under and their chained mesh-under routing (C-MUR) in a testbed consisting of 6 nodes arranged in a line topology, they observed that C-MUR achieves a latency between mesh-under and route-over and a better packet reception rate than both for an increasing number of fragments.

An important issue when applying direct forwarding within a 6LoWPAN is the possible self-interference of fragments of the same datagram along a multi-hop path. Gnawali et al. acknowledged this problem of collisions with formerly forwarded frames, though in the slightly different context of their routing protocol for WSNs (CTP: Collection Tree Protocol [12]). To minimize the possibility for such self-interference, a restriction is introduced to the rate with which frames are forwarded by CTP. This approach is adopted by our rate-restricted modes which are introduced in Section 3.

[1] route-over: the "classic" re-assembling mode; enhanced route-over: a virtual re-assembling mode, which actually directly forwards individual fragments and thereby corresponds to our Direct Mode

[2] http://tools.ietf.org/wg/roll/



(a) Assembly Mode **(b)** Direct Mode

Figure 1. Illustration of message flow in Assembly Mode and Direct Mode

3. Forwarding Techniques

With the Assembly Mode (refer to Section 1) each datagram is completely reassembled at each intermediate IPv6 hop. In contrast, the Direct Mode describes the mechanism which works according to the implementation guidelines for 6LoWPAN [5]. Fragments of datagrams which are not destined to the receiving node are directly forwarded by determining the next hop from the IPv6 routing table. At arrival of the first fragment, a node creates an entry in a so-called virtual reassembly buffer, which is used to identify the following fragments and keep track of the status of in-transit datagrams.

3.1. Enhanced Direct Modes

When a node forwards an arrived fragment immediately to the next node it will compete for the channel with the previous node trying to send the next fragment. While this problem is solved by the CSMA/CA of the MAC Protocol, adding another hop will in many cases cause a hidden terminal problem and drastically increase the probability for collisions at the intermediate node. CTP (see Section 2) uses a rate restriction to decrease the impact of the hidden terminal problem in high traffic scenarios, i.e., in case nodes have several frames stored in their send queue: Every node, after having forwarded a frame, will delay the transmission of consecutive frames.

We adopted this strategy in two different ways: First, we defined and implemented a mode which is identical to the rate restriction proposed by the collection tree protocol. We observed a mean transmission time for a 96 bytes 6LoWPAN fragment of $t_{\text{tx}} = 6\,\text{ms}$, including backoffs and transmission time. Under the assumption that a routing protocol will choose the shortest paths, the channel will be free again after waiting for the duration of two transmissions following the initial one. For example, consider $A \rightarrow B \rightarrow C$: when C has finished, the danger of a collision at B is greatly reduced. Therefore, after each transmisstion, we randomly schedule a delay t_d, with

$$1.5 \cdot t_{\text{tx}} \leq t_{\text{d}} \leq 2.5 \cdot t_{\text{tx}} \tag{1}$$

We call this mode **Direct Mode with Rate Restriction (Direct-RR)**.

This strategy, however, also has some obvious issues. First, the average transmission time can be different for different nodes, depending on their position in the network and the current traffic situation. Second, the transmission time strongly depends on the configuration of the 802.15.4 link layer, e.g., changing the minimum backoff exponent will dramatically increase the average duration of a transmission. To mitigate the impact of these issues, we propose an adaptation of the used transmission delays to the actual transmission time. We call this mode **Direct Mode with Adaptive Rate Restriction (Direct-ARR)**. Instead of setting a fixed rate restriction, the 6LoWPAN layer continuously measures the actual transmission time and calculates an exponentially weighted moving average (EWMA) to estimate the average transmission time: $t_{\text{tx}} = \alpha t_{\text{tx}} + (1 - \alpha) t_{\text{tx,curr}}$. The actual delay is again determined by Equation (1). Note that, as the number of link layer retries also influences the transmission time, Direct-ARR Mode essentially implements a local congestion avoidance.

3.2. Retry Control

Transmitting larger datagrams with several frames will increase the risk of one fragment getting lost on the way. One lost fragment results in the loss of the complete datagram. When such a loss occurs, all transmissions of fragments before have been in vain and worthlessly produced network traffic. For this reason and with IPv6 following a best-effort delivery, link layer retries are

desperately needed to prevent unacceptably high end-to-end loss rates. Therefore we set the number of link layer retries to 7 in our experiments and simulations.

Additionally, we propose a retry control mode to decrease the probability of unnecessary transmitted frames in the network. If a large part of a datagram has already been transmitted successfully to the next hop, we put more effort on transmitting the following parts. We call this method **Progress-Based Retry Control (PRC)**. The number of retries is calculated as follows, where s is the size of the fragmented datagram and s_{trans} the already transmitted size:

$$N_{Retries} = 7 + 8 \times \frac{s_{\text{trans}}}{s} \tag{2}$$

This results in a number of 7 to 15 MAC retries, with 15 being the maximum number of retries provided by the hardware-supported automatic acknowledgement mechanism of the transceiver used in our testbed.

4. Methodology

We integrated all forwarding techniques into our 6LoWPAN implementation for CometOS[3] [13]. CometOS enables the reuse of its C++ module implementations for simulation in the OMNeT++ framework and for the testbed deployment. To avoid the influence of any routing mechanism to the measurement result, we applied a static routing scheme during all experiments. CometOS' physical channel model is based on the MiXiM framework[4]. For our simulation runs, we used a channel model resembling a LogNormal Shadowing with a given fixed average signal strength and a variance. Different from a standard propagation model, we configured each link individually by means of a configuration file.

4.1. Scenarios

For simulations we considered five different network topologies as shown in Figure 2.

The chain-like network (Fig. 2a) was chosen because we expect that the benefits of pipelining fragments are most clearly visible in this setup. In contrast, the "Star" network (Fig. 2c) exhibits paths with a maximum of three hops and therefore does not yield any potential for pipelining and clearly favors the assembly modes in this regard. On the other hand it contains enough nodes routing their traffic over the central node to reveal potential bottlenecks concerning the available buffer space. The Y network (Fig. 2b) again provides tremendous potential for pipelining while at the same time it contains a potential bottleneck. Note that for Chain, Star or LongY network the shown routing topology also matches the connectivity of the network, i.e., the physical links between the nodes.

The RealSim networks (2d and 2e) were modeled after real world networks and thereby represent more typical WSN topologies. They were created by collecting link data (received signal strength indicator (RSSI) mean and variance, packet reception rate (PRR)) from the testbeds itself and installing the corresponding links into our physical channel model. Static routes for these scenarios were created by executing the Dijkstra algorithm on the collected link data, where the weight of the edges was set to the product of the ETX values for incoming and outgoing links. Although this method does not capture the transient properties of links in a real world deployment, where links may exhibit dramatic changes of the experienced PRR, or the possible interference from other networks (IEEE 802.11), it enables the comparison of results from the testbed with those from an equivalent simulated network topology. For RealSim and the actual testbed networks, we used two different (routing) topologies for the two sets of experiments (shown in Figures 2d and 2e for the first and second set, respectively), because between the two sets a longer period of time passed and the exact positioning of nodes had slightly changed.

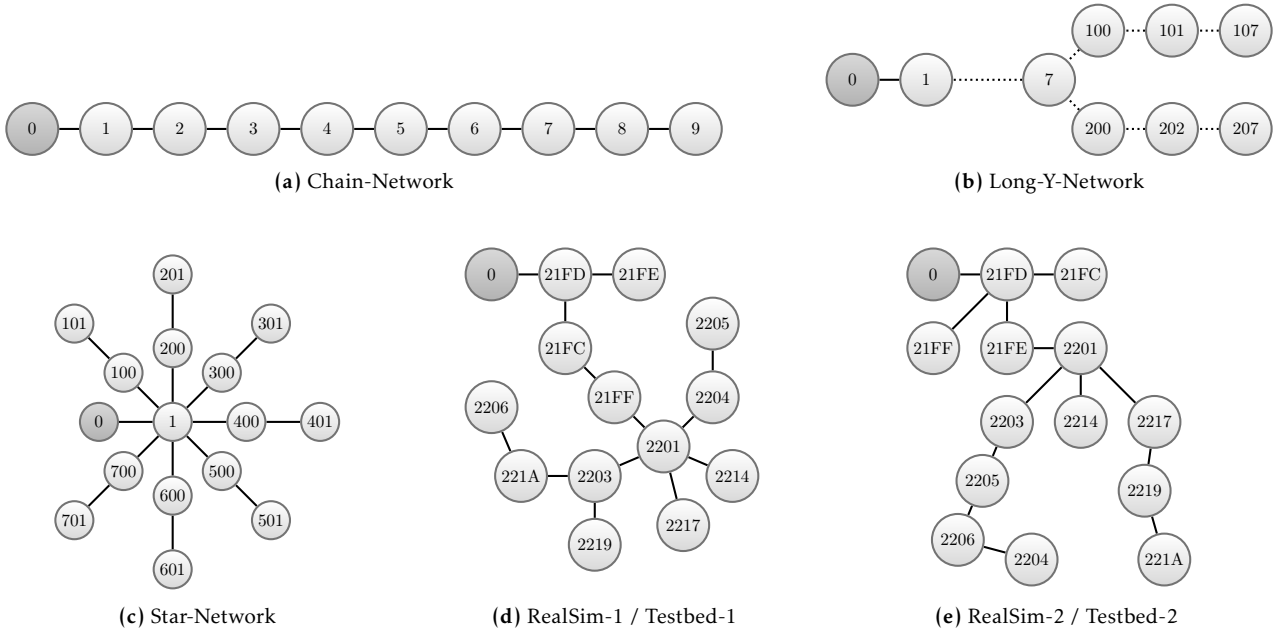
In the Chain and Star networks, the links were set to artificially achieve a PRR of virtually 100% at the link layer. Frame collisions on the other hand inevitably lead to datagram losses for those setups. In the Long-Y setup, each link exhibits a frame error rate of 8.3 % (before applying 802.15.4 retransmissions) for a 96 bytes 6LoWPAN fragment.

One dominant traffic pattern in WSNs is to collect data from the sensors to a sink. We restricted our experiments to this traffic pattern and let every node send UDP data packets towards the sink with different rates λ and payloads. The interval i between two consecutive UDP packets has a fixed component and a random component according to $i = I + \frac{1}{2\lambda}$, with I being uniformly distributed within $\left[0, \frac{1}{\lambda}\right]$.

4.2. Testbed

For the testbed, we deployed 13 ATmega128RFA1 radio modules in an office environment. The ATmega128RFA1 is a single chip transceiver/mcu using an 802.15.4 physical layer and provides 16 kB of RAM and 128 kB of program memory. The static routing tables for the testbed are based on the same link data which were used to create the RealSim. To overcome the problem of transient link behavior (confer [14]) we used a lower transmission power for determining the routes than for the actual experiment. While this measure in many cases caused routes to be chosen too

[3] http://www.ti5.tuhh.de/research/projects/cometos/
[4] http://mixim.sourceforge.net/index.html



(a) Chain-Network **(b)** Long-Y-Network

(c) Star-Network **(d)** RealSim-1 / Testbed-1 **(e)** RealSim-2 / Testbed-2

Figure 2. Network routing topologies. Edges represent static routes, the dark gray node is the sink.

pessimistically and thereby artificially increased the diameter of the resulting routing topology, it turned out to be necessary to guarantee that the network was connected most of the time.

To be able to determine the latencies of UDP packets in the testbed we introduced a time synchronization mechanism which makes use of timestamps provided by the transceiver driver. In order to keep the traffic overhead introduced by this mechanism low, we reduced the rate at which new synchronization beacons are sent to an average of once every 75 s.

For our first set of experiments, the 6LoWPAN layer of our implementation has an assembly buffer of 2000 bytes, which is also used for buffering enqueued fragments and as data storage for datagrams to be sent. For a second set of experiments we reduced the buffer size to the IPv6 minimum MTU of 1280 bytes.

In the Assembly Mode, we set assembly_entries to 10, i.e., up to 10 datagrams can be reassembled at the same time (given that their combined size fits into the buffer). As with the direct modes only datagrams which are destined to the node itself have to be reassembled, we set this value to 4. For the direct modes, a tiny fragment buffer keeps track of the state of up to 15 in-transit datagrams, by setting tfb_size to 15. With this configuration both modes use about the same amount of RAM yielding a basis for a fair comparison. Concerning program memory, our implementation of the Direct Mode uses 3910 bytes more than the corresponding implementation of the Assembly Mode. As the Direct Mode basically has to provide the same assembly service for packets destined to the node itself, plus offering the additional services for direct forwarding, this was to be expected. The rate-restricted modes mainly use a single variable of 2 bytes to store the current estimated transmission time and a CometOS timer object which is scheduled after each transmission corresponding to 1.

While the nodes are constrained with regard to the size of the assembly buffer and assembly or forwarding information data structures, for all experiments we provided the base station with buffers and structures large enough to not cause any drops of datagrams. We deem this approach reasonable as we expect 6LoWPAN border routers to be slightly more powerful devices, equipped with larger memories and a more resourceful power supply.

5. Evaluation

In this section, we compare the different forwarding techniques in terms of packet reception rate and latency. Our RealSim network is used to verify the comparability of the simulation results with the testbed network. In the simulations, we used five runs with each node sending a fixed amount of (application payload) data of 240 000 bytes each run, in the testbed we sent 48 000 bytes. This results in 40 packets of 1200 bytes to 480 packets of 100 bytes per run in the testbed. Four runs per configuration were executed in the testbed.

For depicting the latencies we use boxplots, depicting the minimum and maximum measurements by whiskers, the 10th and 90th percentile by the box and the median by the line in the middle. The confidence

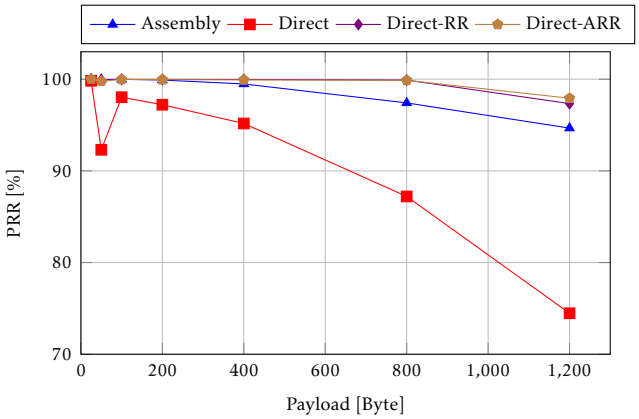


Figure 3. PRR of the Chain Network 37.5 B/s; first set of experiments

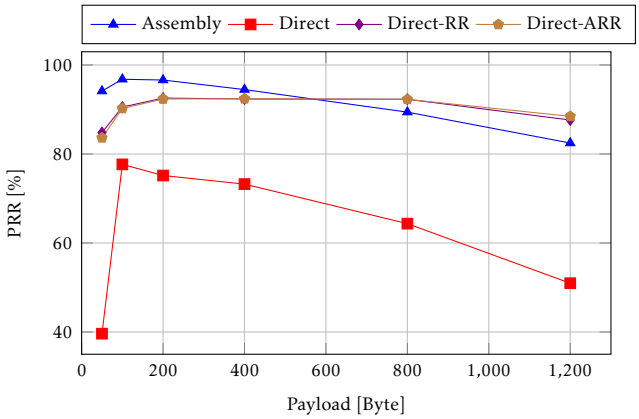


Figure 6. PRR of the LongY Network 37.5 B/s; first set of experiments

intervals shown are 95 % intervals. We show the results for a rate of 37.5 B/s originating at each node. Results for the other (lower) rates do not reveal significantly different trends in the results.

We conducted two sets of experiments, varying the configuration for the 802.15.4 MAC layer. For the first set, we configured the MAC as shown in Table 1, with a minimum backoff exponent of 3, and used the topology shown in Figure 2d for the RealSim and testbed networks. For the testbed, preliminary tests showed very low reception rates for the default setting of *aMaxFrameRetries*=3. Because of this observation and the arguments given in Section 3.2, we set the maximum number of retries to 7 for all experiments, which is also the maximum value for this parameter specified by the standard.

First Set of Experiments.

Chain Network. In the Chain Network, the Direct-RR Mode achieves a better PRR and latency than the Assembly Mode (Figure 3), while the Direct Mode suffers from heavy packet losses due to collisions caused by self-interference. Up to the packet size of 800 bytes Direct-ARR has a PRR of almost 100%, which drops to 96.9% at the packet size of 1200 bytes. As expected, the direct modes exhibited significantly better latency for large fragmented datagrams (shown for 1200 bytes in Figure 4), although for a small percentage of datagrams the maximum values exceed those of the Assembly Mode. For nodes farther away from the sink, the advantage of pipelining datagrams by reusing the channel becomes obvious.

As the PRR is near optimum for the Chain Network, PRC has a limited impact on the results and is omitted here. Only the Direct Mode without any Rate Restriction can profit from PRC with an increased PRR by 3%, but is 17% worse than the Assembly Mode.

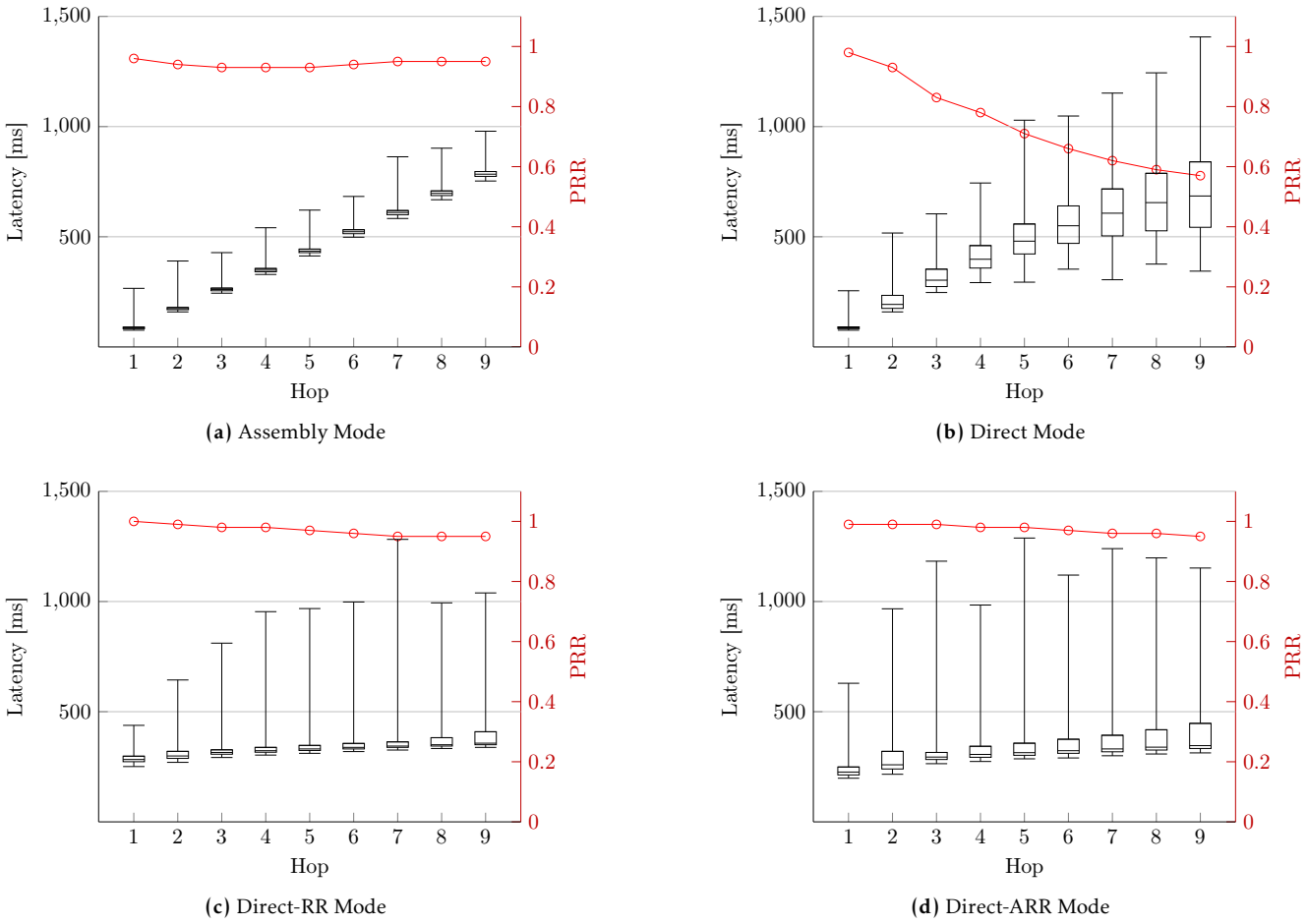
Star Network. In the Star Network, no forwarding mode achieves a PRR of 100% (Figure 5). Multiple opportunities for hidden-terminal-caused collisions exist in every branch and at the centering node, whereas the possibilities for self-interference (and pipelining) on the short way are rare. For these reasons, the Assembly Mode and the Direct Mode perform similar in terms of PRR and latency (which we omitted in this contribution). The comparatively steep drop in PRR of the Assembly Mode at 1200 bytes is due to an increased number of drops caused by lack of buffer space at the central node. For the Star Network, the usage of retry control increases the PRR by 2 % to 4 %.

Long-Y Network. In the LongY Network, the Direct-RR Mode and the Direct-ARR Mode show almost no difference and perform comparable to the Assembly Mode regarding the PRR (Figure 6). For payloads over 800 bytes these modes exhibit a even better PRR than the Assembly Mode. With PRR getting down to 60% and only up to less than 90%, the classical Direct Mode performs impractical even with the PRC enhancement.

In terms of average latency (see Fig. 7), the rate-restricted direct modes outperform the Assembly Mode significantly: For 1200 bytes and at a distance of 15 hops, the median of the direct modes (RR: 395 ms, ARR: 392 ms) is less than a third of that of the Assembly Mode (1311 ms).

All of the PRR results show that the PRR with 100 bytes is higher than with 50 bytes payload. This can be explained by the fact that 50 and 100 bytes payload both result in a datagram with two fragments (with the corresponding control overhead), but the datagrams of 100 bytes payload are sent at only half the rate.

RealSim and Testbed. Figures 9a and 9b show the PRR of the RealSim and Testbed Network with a byterate of 37.5 B/s. Note that the payloads of 50, 200 and 800 Bytes have not been used in the testbed, but only



(a) Assembly Mode

(b) Direct Mode

(c) Direct-RR Mode

(d) Direct-ARR Mode

Figure 4. Per hop latency and PRR in the Chain Network with 37.5 B/s and 1200 bytes payload; first set of experiments

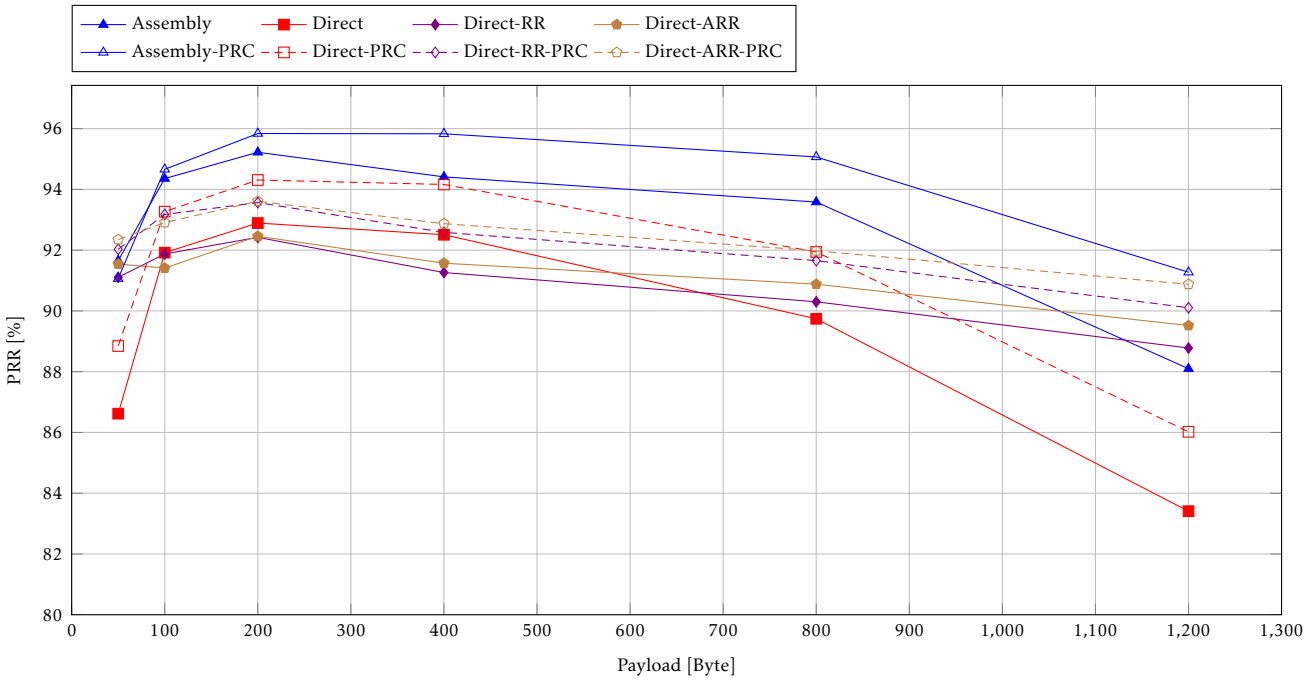
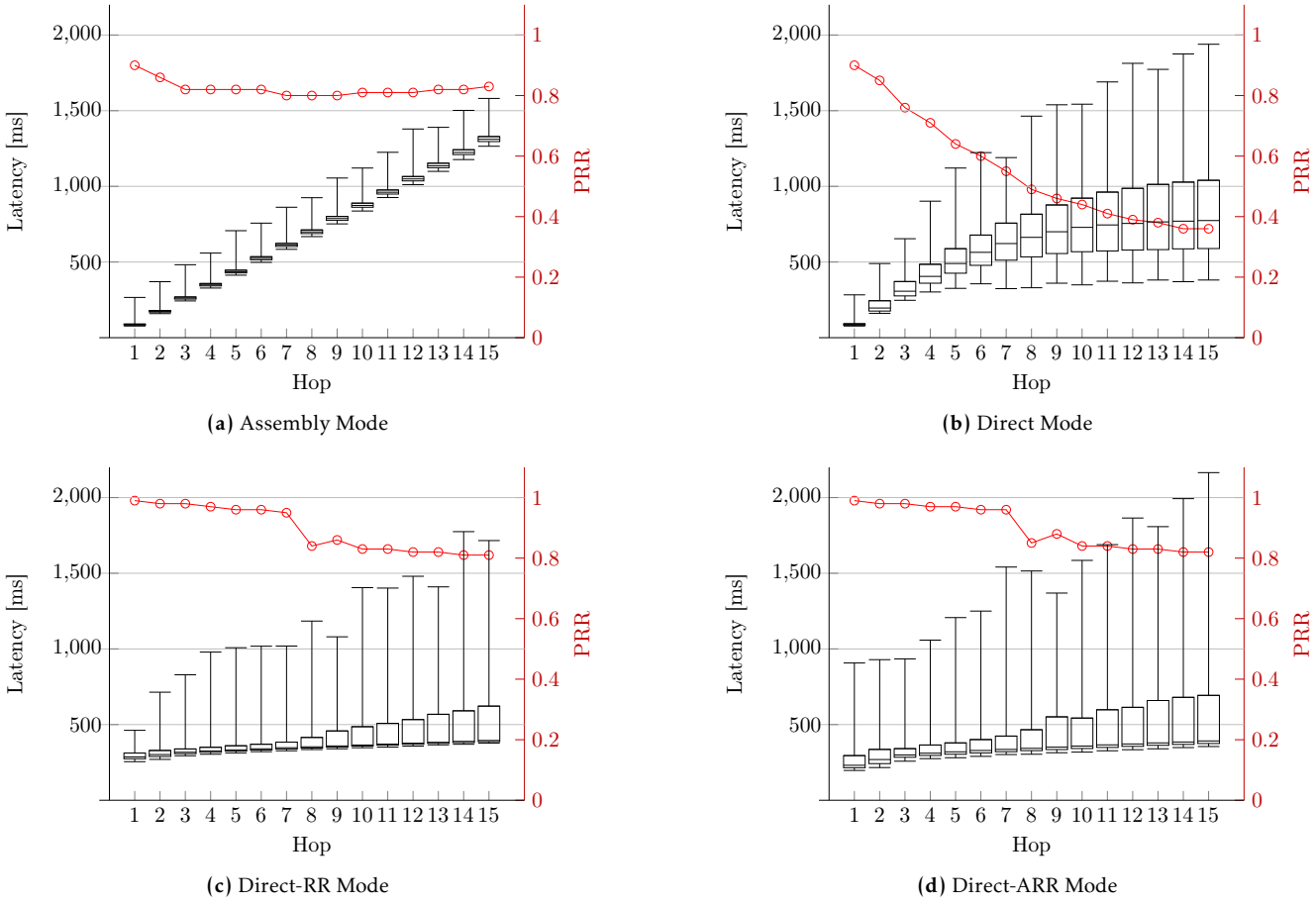


Figure 5. PRR of the Star Network 37.5 B/s; first set of experiments

macMinBE	macMaxBE	macMaxCSMABackoffs	macMaxFrameRetries
3	8	5	7

Table 1. Configuration of the underlying 802.15.4-based MAC layer; first set of experiments



(a) Assembly Mode

(b) Direct Mode

(c) Direct-RR Mode

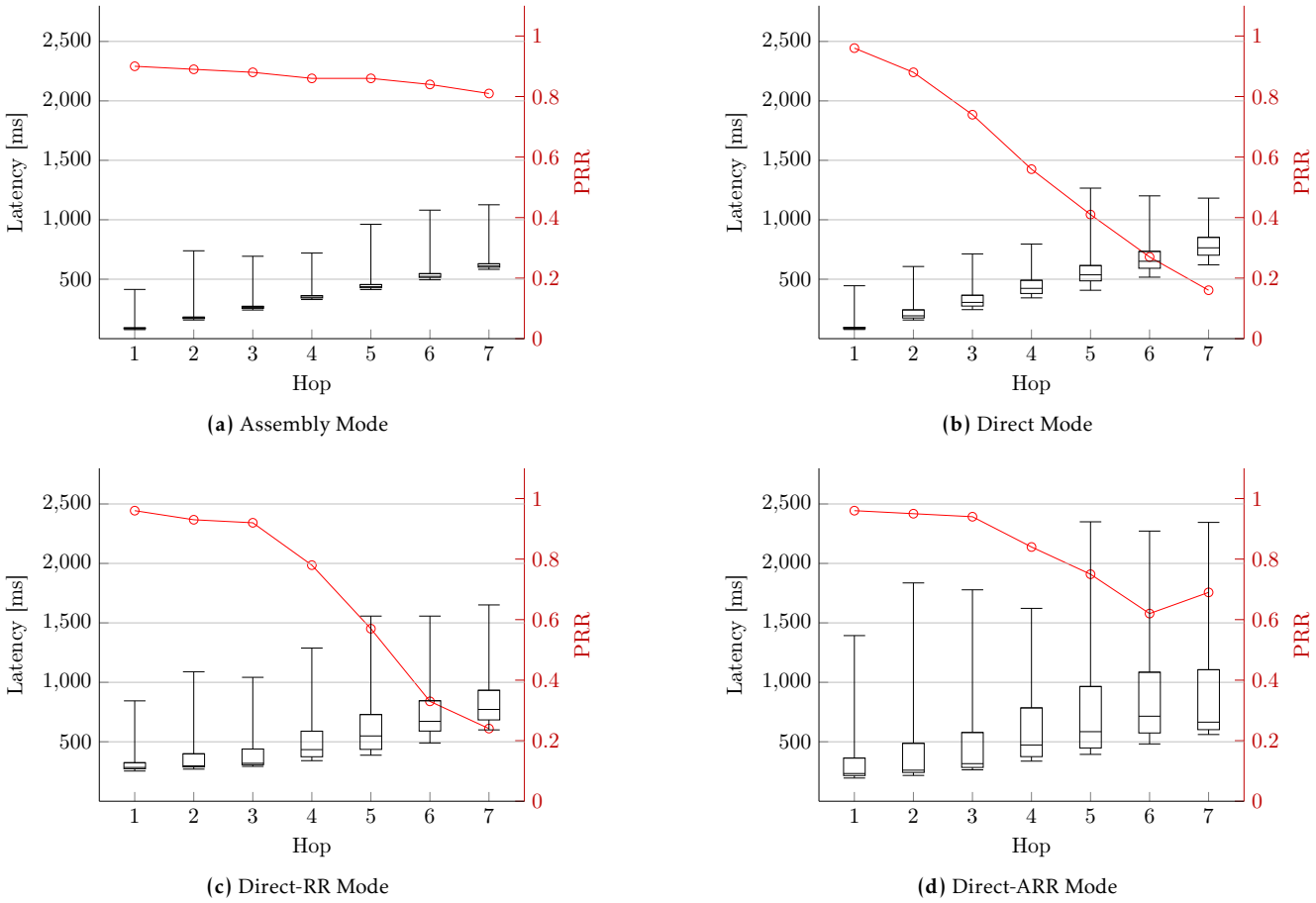
(d) Direct-ARR Mode

Figure 7. Per hop latency and PRR in the Long-Y Network with 37.5 B/s and 1200 bytes payload; first set of experiments

in the simulation. Naturally, some differences can be observed between simulation and experiments in the real network. The overall PRR for all modes are lower and the confidence intervals of the averages from the testbed are more widespread. We explain these differences with the nature of a real world environment. During the experiments there were people moving in the office building, which also contains various WiFi hotspots causing additional interference. The transient behavior of some wireless links may have caused a temporal degradation of the link quality between certain pairs of nodes. The mechanism for time synchronization additionally puts a small load on the real network. All in all, the results of the RealSim network and the testbed network show similar tendencies and seem to confirm the simulation results as an accurate-enough estimation of the real world.

As a first result, we can see that the Direct Mode has the worst PRR of all modes. Direct-ARR outperforms Direct and Direct-RR, but has still a worse PRR than the Assembly Mode. This trend can be observed in the testbed even stronger than in the RealSim.

Figures 8a, 8b, 8c and 8d show the latency results for the RealSim Network with the Assembly, Direct, Direct-RR and Direct-ARR Mode. We can see that the Direct Mode has no significant difference in latency, but the PRR drops dramatically for further hops. The rate restriction of the Direct-ARR Mode achieves similar latencies, but with a higher PRR, though the latency is more widespread. It has to be noted that the static routes chosen for the RealSim and testbed did not reflect the actual transmission range of the nodes (see 4.2), and the "real" network diameter most of the time was rather 4 instead of 7. Therefore, the direct modes



(a) Assembly Mode

(b) Direct Mode

(c) Direct-RR Mode

(d) Direct-ARR Mode

Figure 8. Per hop latency and PRR in the RealSim Network with 37.5 B/s and 1200 Byte payload; first set of experiments

could not benefit from pipelining and exhibit latencies not significantly better than the Assembly Mode.

Second Set of Experiments. To validate the obtained results for other settings of the MAC layer, we conducted an additional set of experiments. Especially the minimum backoff exponent parameter of the 802.15.4 MAC is a candidate which possibly has a large impact on the overall performance, especially for the Direct Mode. With the minimum exponent of 3 we used for our first set of experiments, the first backoff time is chosen from the interval $[0, 2.56\,\text{ms}]$. This is slightly smaller than the raw transmission time (including all headers) of 3.072 ms for one of our 96 bytes fragments. Using this minimum backoff exponent, consecutive transmissions of fragments of the same datagram are virtually guaranteed to overlap, yielding a high probability for a harmful collision in presence of hidden terminals (confer Section 3). Note also that a node which has not received an acknowledgement for a frame will start again with the minimum backoff exponent for its CSMA/CA mechanism.

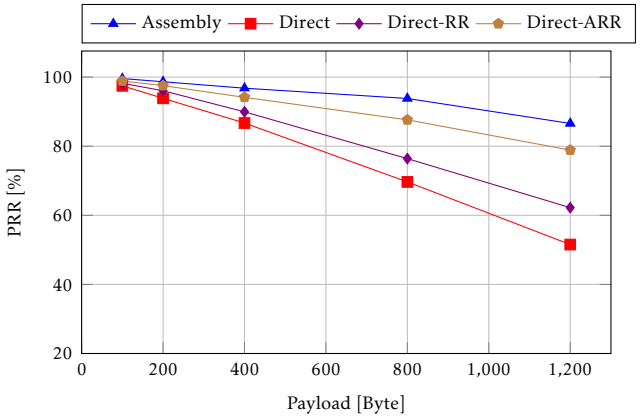
To evaluate if a larger backoff-exponent possibly already solves the problem of collisions (and thereby partially invalidate the usefulness of the Direct-ARR Mode), we repeated all configurations with an minimum backoff exponent of 5, as shown in Table 2. This yields an interval of $[0, 10.24\,\text{ms}]$ for the first random backoff. As the Direct-ARR Mode has outperformed the Direct-RR Mode in all scenarios and the fixed rate restriction of Direct-RR is rather inflexible with regard to changes of MAC parameters, we excluded it from further investigations.

Because some time passed between the first and second sets of experiments, we again used the method described in Section 4, to refresh the routing topology for our testbed as shown in Figure 2e. The resulting topologies are similar with regard to their average path length (3.62 for the first, 3.92 for the second set of experiments).

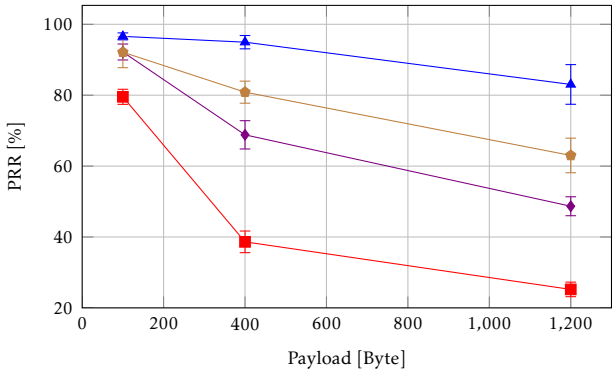
The results in Figure 10a show a dramatically improved reception rate for the Direct Mode for the different simulation configurations, compared to the first set of experiments (see Figure 9a for reference). The PRR improved from 50 % to 90 % for 1200 byte datagrams and thereby is nearly on par with the Direct-ARR Mode, which only increases its PRR from

macMinBE	macMaxBE	macMaxCSMABackoffs	macMaxFrameRetries
5	8	5	7

Table 2. Configuration of the underlying 802.15.4-based MAC layer; Experiment 2

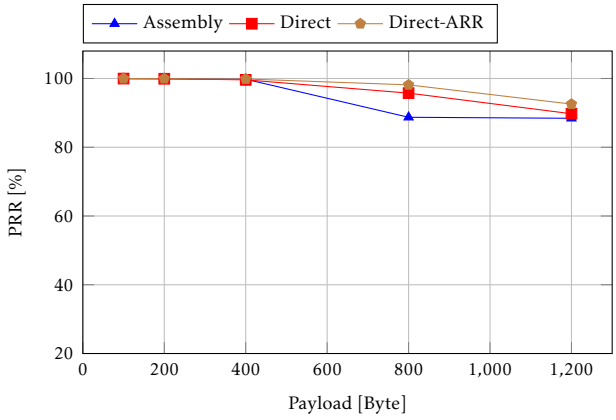


(a) RealSim PRR

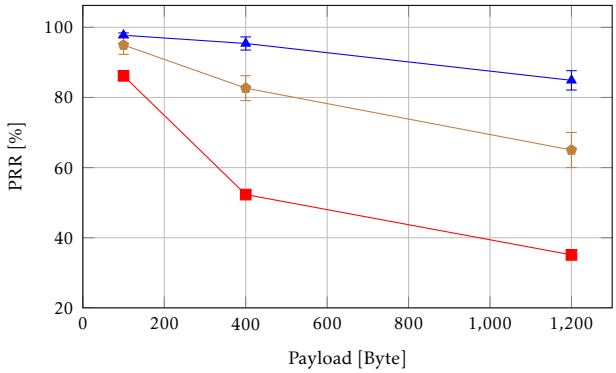


(b) Testbed PRR with 95% confidence intervals

Figure 9. Comparing the packet reception rates of the RealSim and the Testbed Network with a byterate of 37.5 B/s; first set of experiments



(a) RealSim PRR



(b) Testbed PRR with 95% confidence intervals

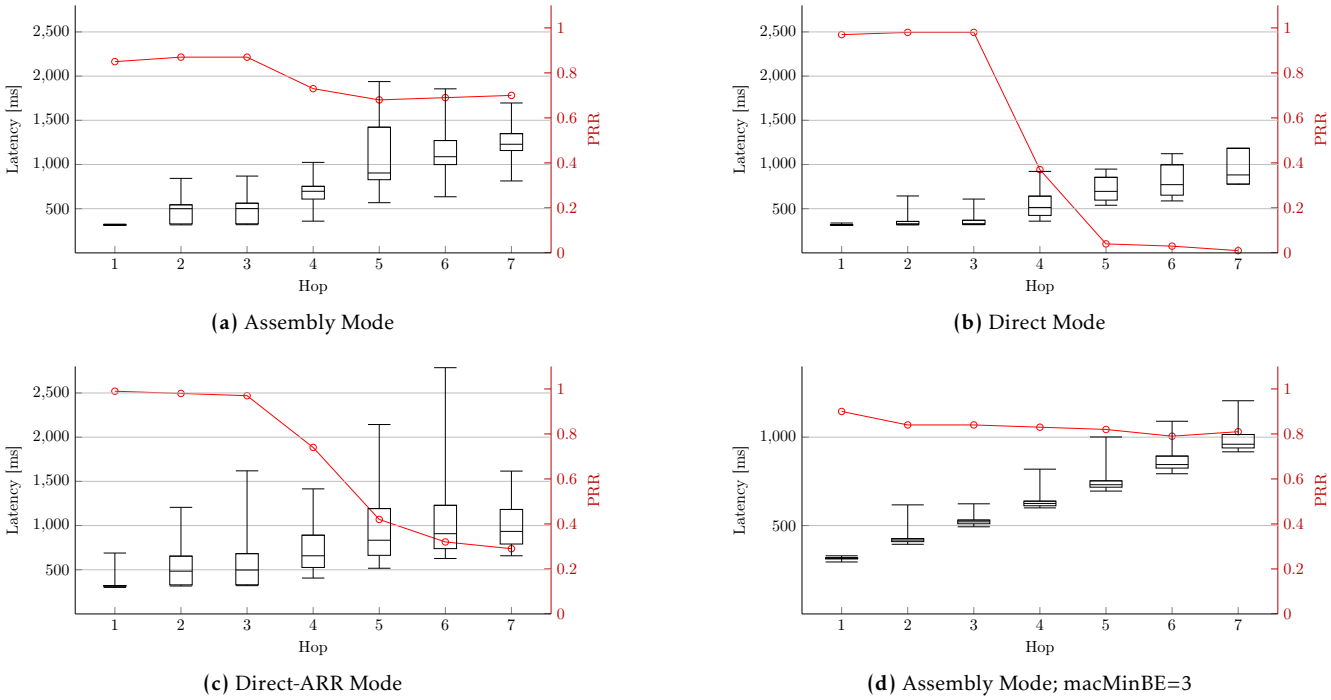
Figure 10. Comparing the packet reception rates of the RealSim and the Testbed Network with a byterate of 37.5 B/s; second set of experiments

79 % to about 92 %. The Assembly Mode does not show remarkable differences. Note that the decrease for 800 byte datagrams can be explained by the reduced buffer size we used for the second set of experiments, which otherwise does not have a remarkable influence on the results.

A completely different result was obtained for the testbed installation. While the Direct Mode's PRR is increased by a rather large amount of about 15 %, it is by no means able to catch up with the performance of the Direct-ARR Mode. Also, while in the simulation environment the two direct modes reach a higher PRR than even the Assembly Mode, this can not be observed in the testbed.

Finally, Figure 11 shows the observed latencies and the PRR depending on the number of hops from the base station node for the three different modes. First, we can observe – in contrast to the results of the simulation with macMinBE=3 (Figure 8) and macMinBE=5, which is not shown – that the direct modes achieve better median latencies than the Assembly Mode. Second, we can see a significant increase in overall latency (Figure 11 shows results from the first set of experiments) for all hops due to the larger average backoff time preceding each transmission of an 802.15.4 frame.

On the other hand, the plot shows in a more drastic way, how severely the PRR drops after a small number of hops in the network. The main reasons for drops of fragments and thereby datagrams in the testbed are lack of buffer space and unsuccessful link-layer transmissions. The former only occurs with the Assembly Mode, the later is the sole significant reason



(a) Assembly Mode

(b) Direct Mode

(c) Direct-ARR Mode

(d) Assembly Mode; macMinBE=3

Figure 11. Per hop latency and PRR in the testbed with 37.5 B/s and 1200 Byte payload; second set of experiments, except 11d, which is a result for macMinBE=3 and is shown for reference (note the different scale on the left y-axis)

for losses for the direct modes in the RealSim and testbed networks.

Thus, differing from the simulation results, the increase of the minimum backoff exponent for the testbed does not dramatically increase the PRR while at the same time significantly increasing the end-to-end latency. In contrast, the Direct-ARR Mode achieves a much higher PRR at the expense of an only slight increase and higher variance of the end-to-end latency, compared to the unmodified Direct Mode.

The tremendous difference between the simulation and the testbed still needs to be explained. We cannot get explicit data on collisions from our testbed and therefore have to draw conclusions from the existing data. Considering the fact that for all datagram sizes with the Assembly Mode and for smaller datagram sizes of 100 bytes and 400 bytes (which put an even slightly larger load on the network!) the PRR is significantly better, we rule out the possibility that sheer bad luck and fluctuating link properties in the testbed lead to the bad performance for the large datagrams. The main reason for losses of frames and thereby datagrams therefore have to be collisions with the fragments of the same datagram or the transmissions of other nodes. As the results have shown, the former can be mitigated by applying the proposed mechanism for adaptive rate restriction.

In the simulation environment, the impact of (self-induced and other) collisions is mitigated by the properties of the used physical channel model. In our model, the received signal strength for a transmission is determined once for each receiving node and stays the same for the whole transmission. Additionally, for each new transmission, e.g., a retransmission after a collision occurred, the signal strength is again determined by drawing a number from the provided lognormal distribution and adding it to the average signal strength for the link. For those two reasons, there remains a chance that of even consecutively colliding fragments, one is still successfully received. In the real testbed, collisions are deemed to have a much larger probability of corrupting an ongoing transmission. Together with the larger backoff interval, which additionally causes collisions to be avoided or at least reduced in duration, this yields a possible explanation for the observed results.

6. Conclusion and Outlook

Using route-over forwarding, 6LoWPAN enables wireless sensor nodes to use a completely standardized IPv6 protocol stack. Two straightforward forwarding techniques are the Assembly Mode and the Direct Mode. Using the Assembly Mode, the transmission of large datagrams exhibits high end-to-end latencies and suffers from a decreased PRR due to buffer size limitations in multi-hop networks. Directly forwarding incoming frames greatly mitigates the problems stemming from

the lack of buffer space, but suffers from a dramatically lower PRR, mainly due to hidden terminal collisions.

We introduced three modified forwarding techniques that are compliant with the 6LoWPAN standard, two of those based on rate-restriction (Direct-RR, Direct-ARR) and the other on retry control. The former two increase the PRR of the Direct Mode to almost the same level as the Assembly Mode in our simulation scenarios. In scenarios with many hops tailored for pipelining these direct modes with a rate restriction exhibit a lower latency than the Assembly Mode while at the same time having a better or similar PPR. On the other hand, the Assembly Mode beats all direct modes in the testbed configuration, in spite of the drops caused by buffer size limitations. Of the two enhanced direct modes, Direct-ARR yield the better results regarding PRR and latency in all simulations and the testbed. The PRR of all modes could be slightly increased by the introduced retry control, although the impact is not as large as hoped.

In a second set of experiments with an increased minimum backoff exponent, which we consider the most-influential link-layer parameter, we observed a dramatic increase of the PRR of the Direct Mode, nearly equaling that of the Direct-ARR for the simulation scenarios. For the testbed, however, this effect is significantly less pronounced and the Direct-ARR Mode still clearly outperforms the plain Direct Mode. The large distinction between the impact of differing settings for macMinBE are attributed to the way the physical channel model used for the simulations determines the received signal strength and the calculation of the resulting bit error rate.

We observed that the Direct-ARR Mode increases the PRR, while only slightly increasing the average latency. However, in its current form it relies on a constant frame size, which obviously is not a very realistic assumption for any real-world application. We plan to improve it in the future especially to make it useful for scenarios with different frame sizes.

The simulations and experiments have shown that the overall performance of 6LoWPAN fragmentation is improvable. An promising approach to overcome the problems of lost fragments is the fragment recovery mechanism (see Section 2) of Thubert and Hui. We plan to extend our 6LoWPAN implementation to support it and are going to evaluate the combination of our Direct-ARR mechanism and SFFR.

References

[1] Montenegro, G., Kushalnagar, N., Hui, J. and Culler, D. (2007), Transmission of IPv6 Packets over IEEE 802.15.4 Networks, RFC 4944. URL http://www.rfc-editor.org/rfc/pdfrfc/rfc4944.txt.pdf.

[2] Institute of Electrical and Electronics Engineering (2011), IEEE 802.15.4-2011 - IEEE Standard for Local and Metropolitan Area Networks— Part 15.4: Low-Rate Wireless Personal Area Networks.

[3] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K. *et al.* (2012), RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, RFC 6550. URL http://www.ietf.org/rfc/rfc6550.txt.

[4] Shelby, Z., Hartke, K. and Bormann, C. (2014), The Constrained Application Protocol (CoAP), RFC 7252. URL http://www.ietf.org/rfc/rfc7252.txt.

[5] Bormann, C. (2013), 6LoWPAN Roadmap and Implementation Guide (draft). URL http://tools.ietf.org/pdf/draft-bormann-6lowpan-roadmap-04.pdf. Accessed: 2014-09-10.

[6] Ludovici, A., Calveras, A. and Casademont, J. (2011) Forwarding Techniques for IP Fragmented Packets in a Real 6LoWPAN Network. *Sensors* **11**(1): 992–1008.

[7] Bhunia, S.S., Sikder, D.K., Roy, S. and Mukherjee, N. (2012) A comparative study on routing schemes of IP based wireless sensor network. In *Wireless and Optical Communications Networks (WOCN), 2012 Ninth International Conference on*: 1–5.

[8] Thubert, P. and Hui, J. (2014), LLN Fragment Forwarding and Recovery (draft). URL https://tools.ietf.org/html/draft-thubert-6lo-forwarding-fragments-01. Accessed: 2014-09-10.

[9] Ayadi, A., Mailles, P., Ros, D., Toutain, L. and Thubert, P. (2011) Energy-efficient fragment recovery techniques for low-power and lossy networks. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*: 601–606.

[10] Ayadi, A., Mailles, P. and Ros, D. (2011) Tcp over low-power and lossy networks: Tuning the segment size to minimize energy consumption. In *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*: 1–5.

[11] Zhu, Y.H., Chen, G., Chi, K. and Li, Y. (2013) The Chained Mesh-Under Routing (C-MUR) for Improving IPv6 Packet Arrival Rate over Wireless Sensor Networks. In *Advances in Wireless Sensor Networks* (Springer Berlin Heidelberg), *Communications in Computer and Information Science* **334**, 734–743.

[12] Gnawali, O., Fonseca, R., Jamieson, K., Moss, D. and Levis, P. (2009) Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*: 1–14.

[13] Unterschütz, S., Weigel, A. and Turau, V. (2012) Cross-Platform Protocol Development Based on OMNeT++. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques (SIMUTOOLS '12)*: 278–282.

[14] Rusak, T. and Levis, P.A. (2008) Investigating a physically-based signal power model for robust low power wireless link simulation. In *Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '08)*: 37–46.