Route-Over Forwarding Techniques in a 6LoWPAN

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

[1] Institute of Telematics
[2] Institute of Communication Networks
Hamburg University of Technology, Hamburg, Germany
{andreas.weigel,martin.ringwelski,turau,timm-giel}@tuhh.de

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 wireless sensor networks. With the envisioned 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. Our evaluation of the basic and enhanced forwarding modes within simulations and a hardware testbed indicate that the proposed enhancements can considerably improve packet reception rate and latency within 6LoWPAN networks.

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

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 wireless sensor networks.

The 802.15.4 standard offers physical- and MAC-layers for low power wireless personal area networks (LoWPAN). While the MAC-frame size of those networks is only 127 bytes, IPv6 depends on a maximum transmission unit (MTU) of at least 1280 bytes.

6LoWPAN offers an intermediate layer between the IP- and the data link layer to overcome this issue. It defines compression algorithms for IPv6 headers and a fragmentation mechanism for larger IPv6 datagrams to be transportable within 802.15.4 MAC frames. Concerning the routing within 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, a node 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 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.

This is a known issue and the informational implementation guidelines [5] recommend the use of a virtual fragmentation buffer, which immediately forwards fragments which 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.

Considering that lost fragments will inevitably lead to lost datagrams, the forwarding strategy has a tremendous impact on the performance within a 6LoWPAN-based wireless sensor network. 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 past research in 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 within 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 approached 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]. Within a similar setup, they analyzed the end-to-end delay and loss rate for a single sender node within 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] describe an extension to RFC4944 which adds negative acknowledgements and fragment recovery mechanisms 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. While certainly worth investigating, this can be seen as an orthogonal approach to the mechanisms proposed by us and will not be further evaluated here.

Wang et al. [9] proposed a method 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 within the slightly different context of their routing protocol for WSNs (CTP: Collection Tree Protocol [10]). To minimize the possibility for such self-interference, a restriction is introduced to the rate with which frames are forwarded by CTP. This technique 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/`

3 Forwarding Techniques

In the following, we call the approach of treating fragments of IPv6 datagrams corresponding to a strictly layered network stack **Assembly Mode**: Each datagram is completely reassembled at each intermediate IPv6 hop. In contrast, we use the term **Direct Mode** for 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 $t_{\mathrm{tx}} = 6\,\mathrm{ms}$, including backoffs and transmission time. Under the assumption that a routing protocol will choose shortest paths, the channel will be free again after waiting for the duration of two transmissions following the initial one[3]. Therefore, after each transmission, we randomly schedule a delay t_d, with

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

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 within 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

[3] Consider $A \rightarrow B \rightarrow C$ – when C has finished, the danger of a collision at B is greatly reduced.

weighted moving average (EWMA) to estimate the average transmission time: $t_{\mathrm{tx}} = \alpha t_{\mathrm{tx}} + (1-\alpha) t_{\mathrm{tx,curr}}$. 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:

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

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 within our testbed.

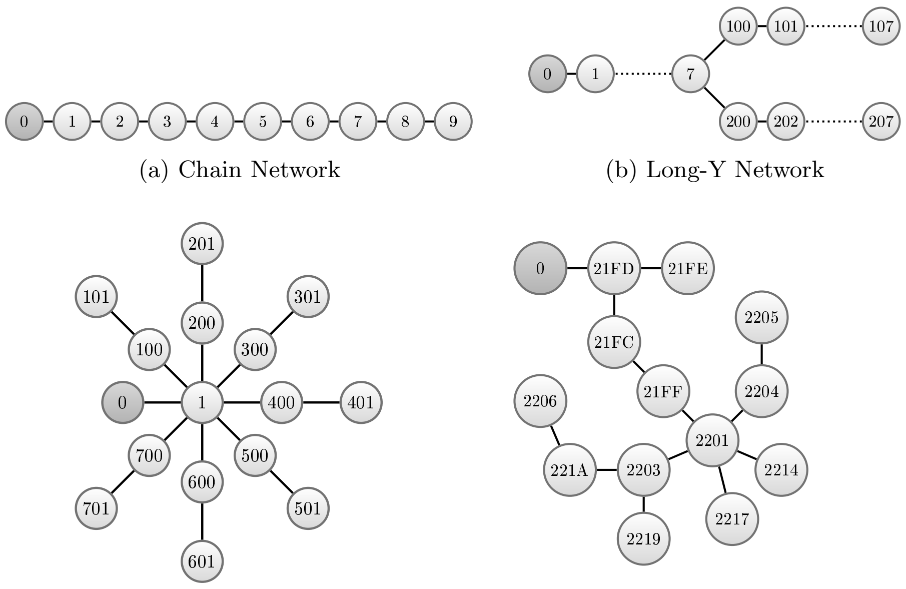
4 Methodology

We integrated all forwarding techniques into our 6LoWPAN implementation for CometOS[4] [11]. CometOS enables the reuse of its C++ module implementations for simulations within the OMNeT++ framework and for the testbed deployment. To avoid 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[5]. 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 four different network topologies as shown in Figure 1. The chain-like network (Fig. 1a) was chosen because we expect that the

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



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

(c) Star Network (d) RealSim

Fig. 1. Simulated networks. Edges represent static routes, the dark gray node is the sink.

benefits of pipelining fragments are most clearly visible in this setup. In contrast, the "Star" network (Fig. 1c) 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. 1b) again provides tremendous potential for pipelining while at the same time it contains a potential bottleneck.

The RealSim network (1d) was modeled after a real world network and thereby represents a more typical WSN topology. It was created by collecting link data (received signal strength indicator (RSSI) mean and variance, packet reception rate (PRR)) from the testbed itself and installing the corresponding links into our physical channel model. Static routes for this scenario were created by executing the Dijkstra algorithm on the collected link data, where the weight of the edges was set as the product of the ETX values for incoming and outgoing links. Although this approach does not capture the transient properties of links within 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 a equivalent simulated network topology.

Within 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. Within 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 within wireless sensor networks 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 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 are used to create the RealSim. To overcome the problem of transient link behavior 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 pessimistically and thereby artificially increased the diameter of the resulting routing topology, it turned out that it was absolutely necessary to guarantee that the network was connected most of the time.

To be able to determine the latencies of UDP packets within the testbed we introduced a time synchronization mechanism which makes use of timestamps within 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.

The 6LoWPAN layer of our implementation has an assembly buffer of 2000 bytes, which is also used for buffering enqueued fragments. In the assembly mode it is possible to reassemble up to 10 datagrams (given that their combined size fits into the buffer). In the direct modes, only 4 datagrams can be reassembled; instead a tiny fragment buffer can forward up to 15 datagrams. With this configuration both modes use exactly the same amount of RAM yielding a basis for a fair comparison.

5 Evaluation

In this section we compare the different forwarding techniques in terms of packet reception rate (PRR) 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 2 000 UDP packets each run. In the testbed we send 48 000 bytes in UDP packets of 100, 400 and 1200 bytes payload. This results in 40 packets of 1200 bytes to 480 packets of 100 bytes per

Table 1. Configuration of the underlying 802.15.4-based MAC layer

macMinBE	macMaxBE	macMaxCSMABackoffs	macMaxFrameRetries
3	8	5	7

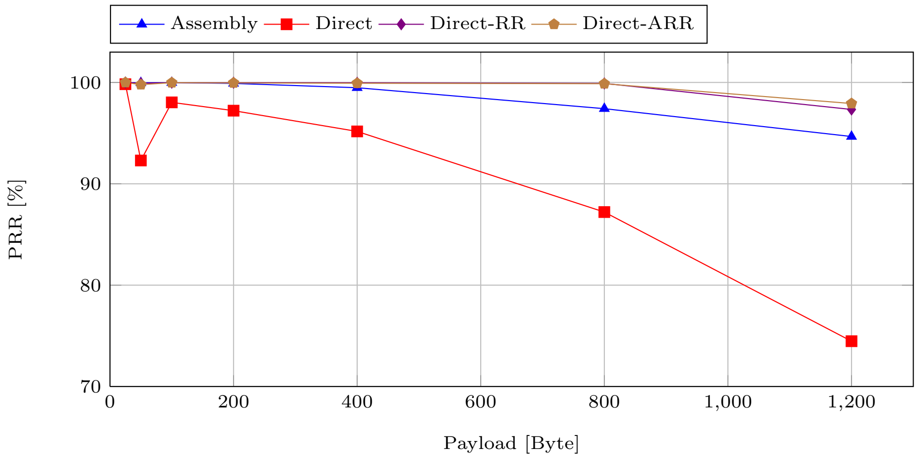


Fig. 2. PRR of the Chain Network 37.5 B/s

run in the testbed. Nine runs were executed per configuration. During all experiments and simulations, the 802.15.4 MAC was set to use the configuration shown in table 1.

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

5.1 Chain Network

In the Chain Network, the Direct-RR Mode achieves a better PRR and latency than the Assembly Mode (Figure 2) while the Direct mode suffers from heavy packet losses due to collisions caused by self-interference. Up to packet sizes of 800 bytes, Direct-ARR has a PRR of almost 100%, which drops to 96.9% at 1200 bytes packet size. As expected, the Direct modes exhibited significantly better latency for large fragmented datagrams (shown for 1200 bytes in Figure 3), 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 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.

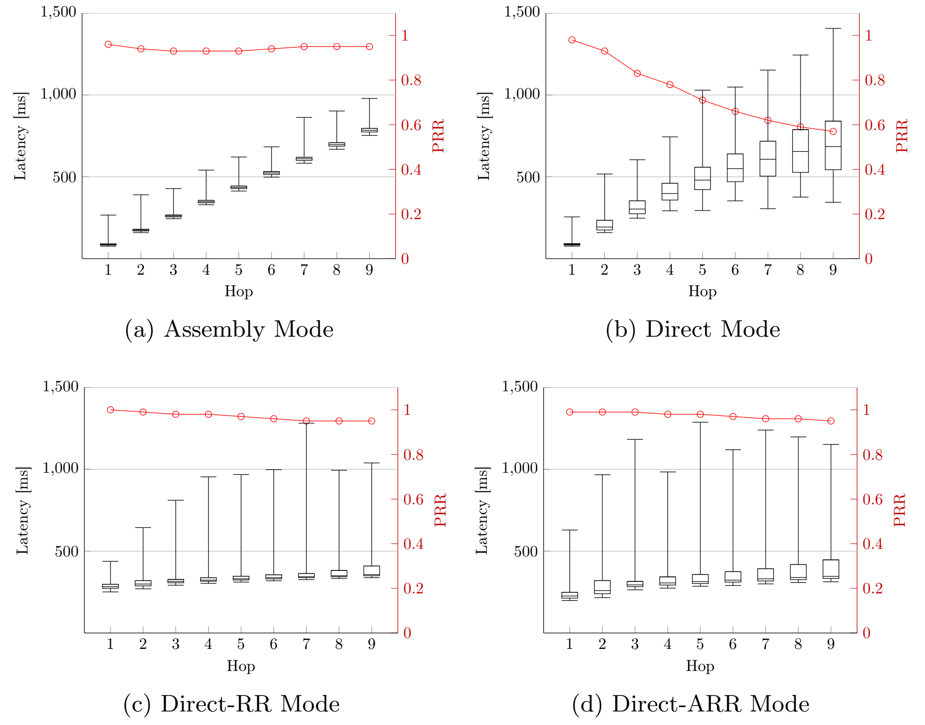


Fig. 3. Per hop latency and PRR in the Chain Network with 37.5 B/s and 1200 bytes payload

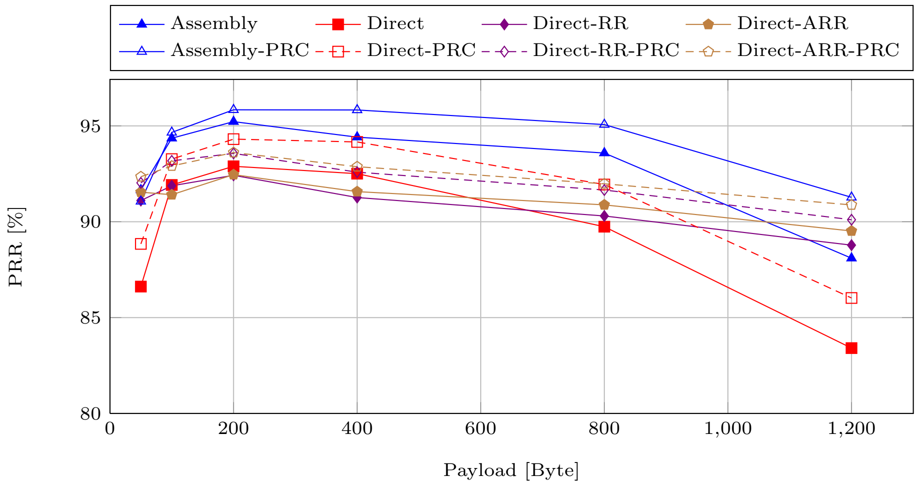


Fig. 4. PRR of the Star Network 37.5 B/s

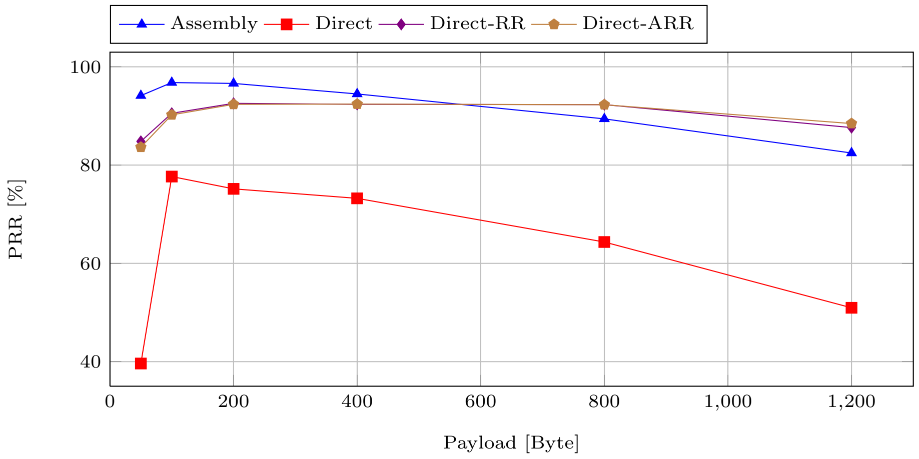


Fig. 5. PRR of the Long-Y Network 37.5 B/s

5.2 Star Network

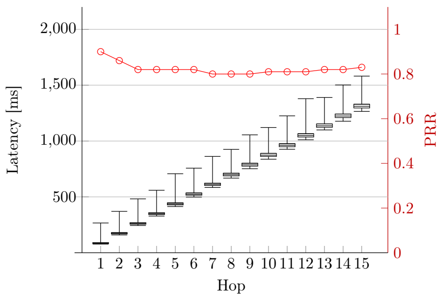
Within the Star Network, no forwarding mode achieves a PRR of 100% (Figure 4). Multiple opportunities for hidden-terminal-caused collisions exist in every branch and at the centering node, whereas the possibilities for self-interference (and pipelining) on the short way are rare. For these reasons, assembly and direct modes perform similarly in terms of PRR and latency (which we omitted). The comparatively steep drop in PRR of the assembly mode at 1200 bytes is due 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 %.

5.3 Long-Y Network

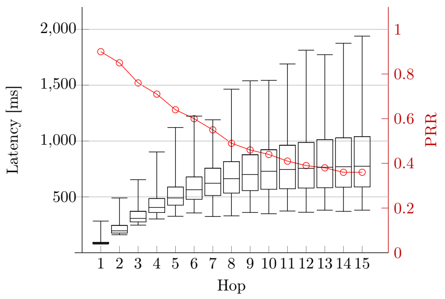
In the Long-Y Network, the Direct-RR and Direct-ARR Mode show almost no difference and perform comparably to the Assembly Mode regarding the PRR (Figure 5). For payloads over 800 bytes these modes exhibit an even better PRR than the Assembly Mode. With PRR getting down to 60% and only up to less than 90%, the classical Direct Mode performs impractically even with the PRC enhancement.

In terms of average latency (see Fig. 6), 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).

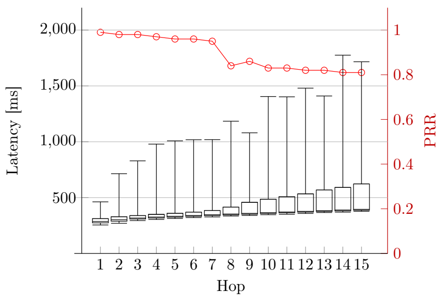
All of the PRR results show that the PRR with 100 bytes is higher than with 50 bytes payload. This can be explained by the fact that 50 and 100 bytes payload both result in a datagram with two fragments (with the corresponding control overhead), but the datagrams of 100 bytes payload are sent at only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



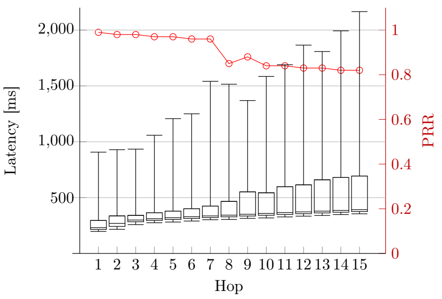
(a) Assembly Mode



(b) Direct Mode



(c) Direct-RR Mode



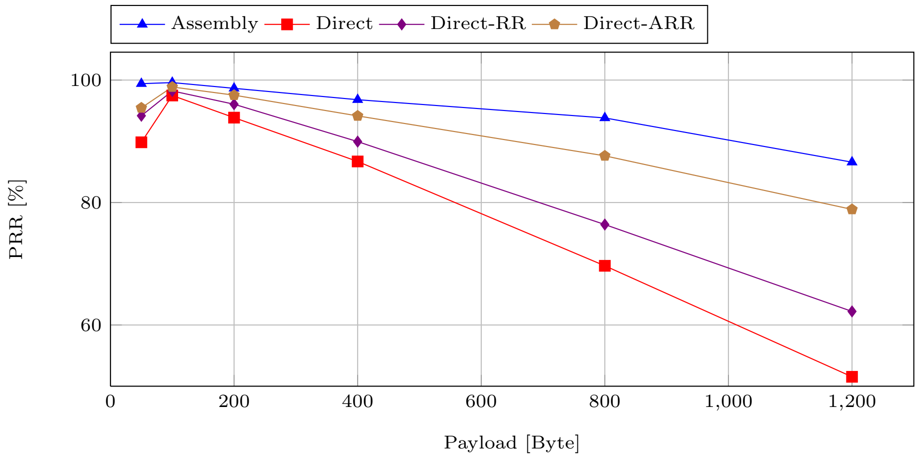
(d) Direct-ARR Mode

Fig. 6. Per hop latency and PRR in the Long-Y Network with 37.5 B/s and 1200 bytes payload

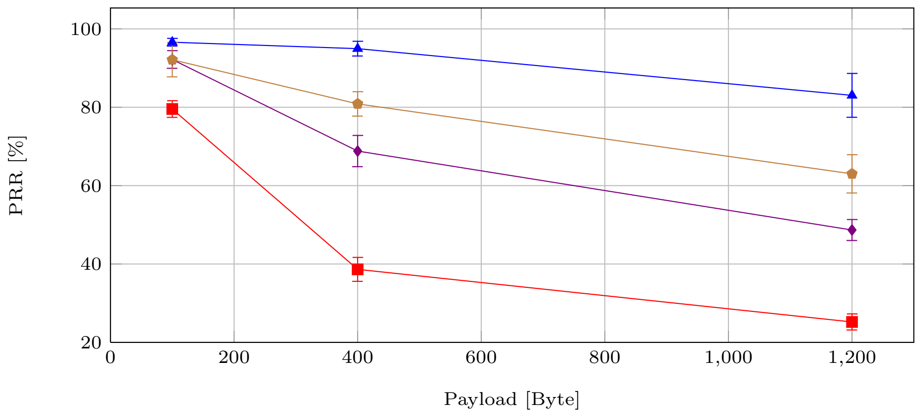
5.4 RealSim and Testbed

Figures 7a and 7b show the PRR of the RealSim and Testbed Network with a byte rate of 37.5 B/s. Note that payloads of 50, 200 and 800 Bytes have not been used in the testbed, but only within 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 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 mechanism for time synchronization additionally puts a small load on the real network. Nevertheless, the results show similar tendencies and 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.



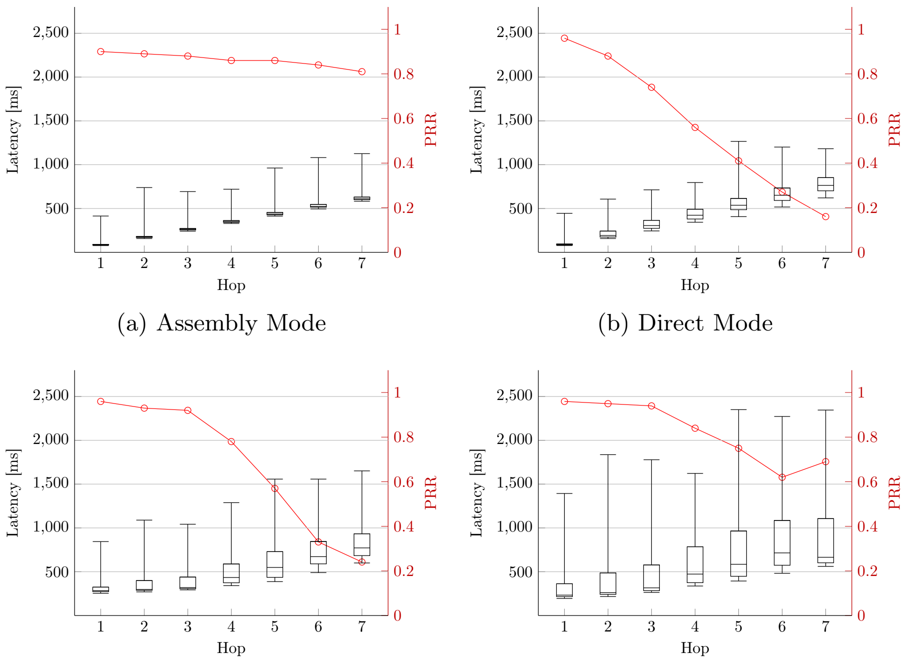
(a) RealSim PRR



(b) Testbed PRR with 95% confidence intervals

Fig. 7. Comparing the packet reception rates of the RealSim and the Testbed Network with a byte rate of 37.5 B/s. Note the different scaling of the y-axes.

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, while achieving a higher PRR, though the latency is more widespread. It has to be noted that the static routes chosen for 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 could not benefit from pipelining and exhibit latencies not better than the Assembly Mode.



(a) Assembly Mode (b) Direct Mode

(c) Direct-RR Mode (d) Direct-ARR Mode

Fig. 8. Per hop latency and PRR in the RealSim Network with 37.5 B/s and 1200 Byte payload

6 Conclusion and Outlook

6LoWPAN enables wireless sensor nodes to use IPv6, but also needs a lot of RAM to be able to forward packets. Directly forwarding incoming frames solves that problem, while suffering from a significantly lower PRR.

We introduced three advanced forwarding techniques that are compliant with the 6LoWPAN standard. These can increase the PRR of the direct mode to almost the same level as the Assembly Mode. In scenarios with many hops tailored for pipelining these direct modes with a rate restriction exhibited a significantly 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.

Of the two enhanced direct modes, Direct-ARR yielded the better results regarding PRR and latency within 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 the future work the selective retry control will have to prove itself against a flat increase of retries. While the latter may even further increase the PRR in many situations, we want to explore the behavior in high traffic situations, where it may also cause additional congestion.

To further increase the performance of 6LoWPAN implementations, we plan to implement a fragment recovery mechanism (see Section 2) and combine it with the (adaptive) rate restriction and/or retry control.

So far, we used only a single and rather aggressive configuration of the 802.15.4 MAC for our experiments. We are going to explore the influence of different parameter sets in the future.

References

1. Montenegro, G., Kushalnagar, N., Hui, J., Culler, D.: Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard) (September 2007)
2. IEEE Standard for Local and Metropolitan Area Networks— Part 15.4: Low-Rate Wireless Personal Area Networks (2011)
3. Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., Alexander, R.: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard) (March 2012)
4. Shelby, Z., Hartke, K., Bormann, C.: Constrained Application Protocol (CoAP) (May 2013), `http://tools.ietf.org/pdf/draft-ietf-core-coap-17.pdf` (accessed: June 11 2013)
5. Bormann, C.: 6LoWPAN Roadmap and Implementation Guide (draft) (April 2013), `http://tools.ietf.org/pdf/draft-bormann-6lowpan-roadmap-04.pdf` (accessed: May 30, 2013)
6. Ludovici, A., Calveras, A., Casademont, J.: Forwarding Techniques for IP Fragmented Packets in a Real 6LoWPAN Network. Sensors (Basel) 11(1), 992–1008 (2011)
7. Bhunia, S.S., Sikder, D.K., Roy, S., Mukherjee, N.: A comparative study on routing schemes of IP based wireless sensor network. In: 2012 Ninth International Conference on Wireless and Optical Communications Networks (WOCN), pp. 1–5 (September 2012)
8. Thubert, P., Hui, J.: LLN Fragment Forwarding and Recovery (draft) (February 2013),`http://tools.ietf.org/html/draft-thubert-roll-forwarding-frags-01` (accessed: June 11, 2013)
9. Zhu, Y.-H., Chen, G., Chi, K., Li, Y.: The Chained Mesh-Under Routing (C-MUR) for Improving IPv6 Packet Arrival Rate over Wireless Sensor Networks. In: Wang, R., Xiao, F. (eds.) CWSN 2012. CCIS, vol. 334, pp. 734–743. Springer, Heidelberg (2013)
10. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection tree protocol. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, SenSys 2009, pp. 1–14. ACM, New York (2009)
11. Unterschütz, S., Weigel, A., Turau, V.: Cross-Platform Protocol Development Based on OMNeT++. In: Proceedings of the 5th International Workshop on OMNeT++ (OMNeT++ 2012) (March 2012)