Hop-to-Hop Reliability in IP-based Wireless Sensor Networks - a Cross-Layer Approach

Gerald Wagenknecht, Markus Anwander, and Torsten Braun

Institute of Computer Science and Applied Mathematics
University of Bern, Switzerland
Email: {wagen, anwander, braun}@iam.unibe.ch

Abstract. To interconnect a wireless sensor network (WSN) to the Internet, we propose to use TCP/IP as the standard protocol for all network entities. We present a cross layer designed communication architecture, which contains a MAC protocol, IP, a new protocol called Hop-to-Hop Reliability (H2HR) protocol, and the TCP Support for Sensor Nodes (TSS) protocol. The MAC protocol implements the MAC layer of beacon-less personal area networks (PANs) as defined in IEEE 802.15.4. H2HR implements hop-to-hop reliability mechanisms. Two acknowledgment mechanisms, explicit and implicit ACK are supported. TSS optimizes using TCP in WSNs by implementing local retransmission of TCP data packets, local TCP ACK regeneration, aggressive TCP ACK recovery, congestion and flow control algorithms. We show that H2HR increases the performance of UDP, TCP, and RMST in WSNs significantly. The throughput is increased and the packet loss ratio is decreased. As a result, WSNs can be operated and managed using TCP/IP.

1 Introduction

Wireless sensor networks (WSN) consist of a large number of sensor nodes. They are used for various applications, e.g., office buildings, environment control, wild-life habitat monitoring, forest fire detection, industry automation, military, security, and health-care. For such applications, a WSN cannot operate in complete isolation. It must be connected to an external network, e.g, the Internet. Through such a network a WSN can be monitored and controlled. The operation of a WSN needs a uniform communication protocol.

The TCP/IP protocol is the de facto standard protocol suite for wired communication. Using TCP/IP has a number of advantages and disadvantages. Thus, it is possible to directly connect a WSN to a wired network infrastructure without proxies or middle-boxes [1]. While in a TCP/IP supported WSN, UDP is used to transmit sensor data to a sink, TCP would be used for administrative tasks such as sensor configuration and code updates. Among the advantages, there are a couple of difficulties running TCP/IP on sensor nodes. The resource constraints of sensor nodes and the high packet loss, which leads to a high number of end-to-end retransmissions, result in a generally bad performance.

A couple of optimizations on different layers reduce the performance problems when using TCP/IP. Avoiding the need of end-to-end retransmissions is the key to increase the performance of TCP. This can be achieved by introducing hop-to-hop reliability mechanisms. UDP in WSNs benefits from hop-to-hop reliability as well. Furthermore, harmonizations between the protocols across different layers are an important target for optimizations. Thus, it is possible to achieve similar performance in terms of data throughput and packet loss rate with TCP/IP as if using common communication protocols for WSNs.

The following research questions arise. How can we design a protocol, which supports hop-to-hop reliability between two neighbor nodes? Can a cross-layer interface harmonize the different protocols on several network layers and increase the performance in terms of a better throughput (equivalent to lower transmission time) and lower error rate? Can TCP and UDP benefit from this architecture and run efficiently on sensor nodes?

The remainder of the paper is structured as follows. After the introduction in Section I, we present related work in the area of reliable transport protocols, TCP/IP adaptation, and cross-layer design for WSNs in Section II. In Section III we present the protocol stack and introduce our cross-layer interface. The Hop-to-Hop Reliability (H2HR) protocol, its cross-layer collaboration with the TCP Support for Sensor Nodes (TSS) [2] protocol and our beacon-less 802.15.4 MAC protocol [3] are described in Section IV. In the evaluation part in Section V, the implementation of the protocol using the OMNeT++ simulator is briefly described and the simulation results are presented. Section VI concludes the paper and gives an outlook.

2 Related Work

The use of TCP in wireless networks causes some serious performance problems [4], caused by end-to-end ACKs and retransmissions. A number of papers propose mechanisms to overcome these problems. In [5] the trade-off between TCP throughput and the amount of Forward Error Correction (FEC) is analyzed and simulated. In [6] the TCP performance is improved by establishing the optimal TCP window size. Caching and local retransmission are promising approaches to reduce the number of end-to-end retransmissions and make TCP feasible for WSNs. TCP Snoop [4] introduces first this approach. In [7] Distributed TCP Caching (DTC) is presented. TCP Support for Sensor Networks (TSS) [2] extends DTC by a novel congestion control mechanism that is very effective as well as easy to implement and deploy.

In the following, some common reliable transport protocols for WSNs are introduced. Directed Diffusion [8] is a popular data dissemination scheme. Reliable Multi-Segment Transport (RMST) [9] has been designed as a new transport layer for Directed Diffusion. It uses a NACK-based transport layer running over a selective-ARQ MAC layer to ensure reliability. It supports two modes: hop-to-hop and end-to-end mode. In the hop-to-hop mode it provides a caching mechanism on intermediate nodes. A lost packet is retransmitted from an in-

termediate node. In the end-to-end mode, lost packets are retransmitted from the source. Pump Slowly Fetch Frequently (PSFQ) [10] is also built on top of Directed Diffusion. It runs over a non-ARQ MAC layer and ensures reliability by using sequence numbers and hop-to-hop recovery based on NACKs.

Besides ARQ, reliability ensuring link layer protocols have been developed for wired networks. Logical Link Control (LLC) is defined in IEEE 802.2. It provides a connection-oriented mode, which works with sequence numbers, and a connection-less mode with ACK frames. The LLC header includes a 16 bit control field and optionally the sequence number. The High-Level Data Link Control (HDLC) is bit-oriented and allows point-to-point and point-to-multipoint connections. Both protocols increase the complexity and overhead of the link layer significantly. The use of sequence numbers and additional header information waste space in the frames. The length of the frames defined in the 802.25.4 standard is limited to 128 bytes. Our H2HR protocol has no own header and does not use sequence numbers.

In [11], current activities in the area of cross-layer designs in WSNs are presented. Different cross-layer approaches are analyzed and a taxonomy to classify them is defined. Open challenges in the area of cross-layer designs are depicted.

3 Protocol Stack and Cross-Layer Interface

The protocol stack as shown in Fig. 1 includes the standard TCP/IP protocol suite with IP, TCP and UDP. The MAC protocol implements the beacon-less mode of the 802.15.4 MAC layer for peer-to-peer topologies. It supports two kinds of acknowledgment mechanisms, explicit ACK using ACK frames and implicit ACK using overhearing. The Internet Protocol remains unmodified. The Hop-to-Hop Reliability (H2HR) protocol is located between the Internet layer and the link layer. It increases the probability of successful delivery of a frame between two neighbor nodes, but does not guarantee reliability. It collaborates with the MAC protocol and the TSS protocol using the cross-layer interface. The UDP protocol is unmodified. Sensor data are transmitted using UDP from the sensor nodes to the base station. Although UDP is not a reliable transport protocol, it benefits from hop-to-hop reliability offered by the H2HR protocol. TCP is a protocol with end-to-end reliability, but it also benefits from H2HR in collaboration with TSS, which optimizes using TCP in WSNs. This is achieved by intermediate caching and local retransmission of TCP data packets, local TCP ACK regeneration, aggressive TCP ACK recovery, congestion and flow control algorithms.

The cross-layer interface offers protocol interaction. Every protocol provides information that other protocols can use to optimize their operation. A cross-layer message consists of a unique ID and a pointer to the exchanged information. Fig. 2(a) shows the cross-layer interface and the interaction between the protocols.

There are two kinds of information exchange. First, a protocol subscribes to certain information. It uses the ID to identify the offered information uniquely.

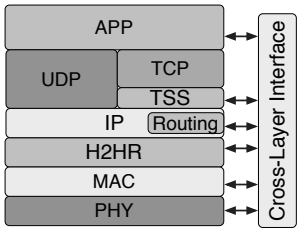
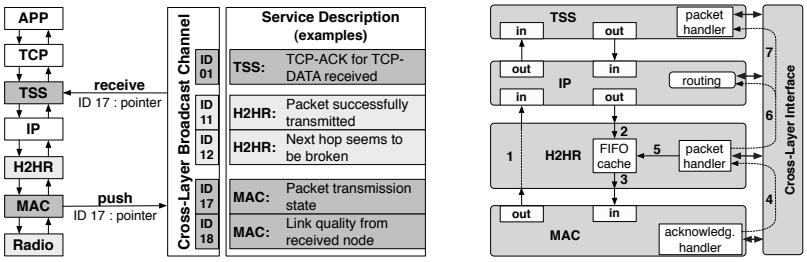


Fig. 1. Protocol Stack.



(a) Cross-Layer Interface. (b) H2HR.

Fig. 2. Protocol Architecture.

When an event has been registered, all subscribers are informed using the cross-layer broadcast channel. The second possibility for cross-layer information exchange is to request information directly from the protocols. The cross-layer interface gets the unique ID of the requested information and transmits the request to the according protocol using the cross-layer broadcast channel as well.

4 Hop-to-Hop Reliability Protocol and Cross-Layer Collaboration

The reliability between two neighbor nodes is ensured by the collaboration of the H2HR protocol with our beacon-less 802.15.4 MAC protocol and TSS across the layers. Fig. 2(b) shows the cross-layer collaboration between the protocols. The MAC protocol [3] supports two kinds of acknowledgment modes: explicit ACK using ACK frames and implicit ACK using overhearing. Both are 802.15.4 conform. In case of explicit ACK, the MAC protocol initiates the transmission of an ACK frame immediately after receiving the frame. The number of retransmission attempts is limited to three. In case of overhearing, no ACK frames are transmitted. Instead, the radio transceiver listens whether the next node forwards the frame. If a frame could not be overheard, it is retransmitted once. The upper layers are informed about the state of acknowledgments using the cross-layer interface. There are three states for the transmission success of a frame:

– The frame has been successfully transmitted (confirmed via ACK frame or overhearing).
– The frame has been transmitted, but there is no confirmation.
– Frame transmission failed.

The H2HR protocol is located between the IP layer and the link layer. Packets from layers below are delivered to the upper protocols according to the type of the packet without any processing (**1** in Fig. 2(b)). Packets from upper protocols are processed as follows. H2HR buffers the packets, which are delivered by IP (**2**) and delivers just one packet at a time to the underlying layers (MAC) (**3**). The other packets are buffered. The MAC layer initiates the transmission to the neighbor node. The H2HR protocol is informed about the transmission state by the MAC protocol using the cross-layer interface, either after the transmission has been successful or after three unsuccessful attempts (**4**). H2HR decides according to the state how the packet is handled (**5**). When the packet has successfully been transmitted to the next node, H2HR deletes this packet from the buffer and delivers the next packet to the underlying layer (**3**). Informed by the MAC protocol, H2HR reacts on two different kinds of problems. A packet can be lost due to interferences or due to congestion. If a packet has been transmitted, but there are no confirmations (neither ACK frame nor overhearing), the packet is lost due to interferences by a hidden node. In this case the transmission is retried immediately after 0.7-1.5 * frame_length (**3**). If the transmission has failed because the channel is busy, congestion is detected. H2HR initiates the retransmission after a random time between 1-2.5 * frame_length. The transmission of a 128 byte 802.15.4 MAC frame takes approximately 4ms with a data rate of 250kbps. After the 6th retransmission attempt initiated by H2HR, it can be expected that the neighbor node has serious problems or the channel is extremely busy. Thus, the packet is deleted and the routing protocol and TSS are informed using the cross-layer interface (**6**, **7**). Then, the routing protocol has to find an alternative route. The TSS protocol stores the packet and gets the control of the retransmission. After the route has been repaired or the channel is again free, retransmission is initiated by TSS.

UDP as an unreliable transport protocol can benefit from hop-to-hop reliability. The number of successfully transmitted packets from the sender to the receiver increases significantly. Overhearing with UDP is more challenging than using ACK frames, because UDP packets do not have any sequence numbers. The forwarded UDP packets can be identified by using a checksum over the UDP payload. In general, every transport protocol, which does not add sequence numbers to its packets, can use overhearing mechanisms at the link layer in this way.

TCP supports end-to-end reliability and lost packets are retransmitted by the sender. Because of the high packet loss ratio in WSNs, this happens quite often and pure TCP in WSNs is not feasible [4]. Hop-to-hop reliability supported by H2HR decreases the number of required end-to-end retransmissions, because it shifts the reliability assurance to intermediate nodes. Because H2HR can fail and does not guarantee the successful transmission of a frame to the next node, TCP end-to-end retransmissions can still occur. Thus, another protocol is re-

quired to support TCP in WSNs. TSS implements intermediate caching and local retransmission of TCP data packets, local TCP ACK regeneration, aggressive TCP ACK recovery, and congestion and flow control algorithms. If a packet is dropped by the H2HR protocol after 6 retransmission attempts due to busy channel, TSS still stores this packet in a buffer. If a TCP-ACK reaches the intermediate node and requests the presumably lost TCP data packet, then TSS transmits the cached packet to the receiver. The request for retransmission does not need to be transmitted to the sending side of the TCP connection. Finally, end-to-end retransmissions are only very rarely required and TCP can be used efficiently in WSNs.

As described above, ensuring reliability happens on different layers in different protocols. The MAC protocol reacts immediately on packet loss. If the frame transmission fails, H2HR intervenes and retransmits the packets depending on the detected problem (interferences or congestion). If H2HR collapses, the reliability mechanisms of the overlaying transport protocols handle the problem. This results in a hierarchy of reliability mechanisms.

H2HR does not require an own header and does not insert any information or sequence numbers into the frames. The length of the payload remains as large as possible. Furthermore, our protocols (802.15.4 MAC, H2HR, TSS) have low complexity and are easy to implement on sensor nodes, which have constraints in memory and processing power. Using the cross-layer interface, the protocols collaborate efficiently. A light-weight H2HR, a light-weight 802.15.4 MAC protocol and a light-weight TSS together have lower complexity as the combination of those mechanisms in one single protocol.

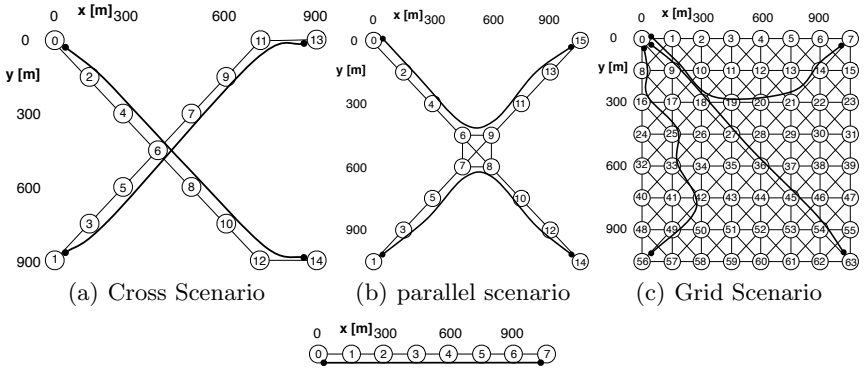
5 Evaluation

We implemented the MAC protocol, H2HR and TSS using the OMNeT++ simulator [12]. To compare a common transport protocol for WSNs with UDP and TCP, we implemented RMST in both modes. RMST is running over IP and 802.15.4, and not over Directed Diffusion [8] and 802.11 as proposed in [9]. Our implementation is based on the NS2 sources for RMST.

We analyze the influence of hop-to-hop reliability on the performance of UDP, TCP and RMST. We compare the packet loss ratio of UDP packets between the MAC protocol without reliability, implicit and explicit ACK, and the combination of H2HR with it. Afterwards, we compare the impact of the reliability mechanisms on the throughput using UDP, TCP with and without TSS, and RMST in the hop-to-hop and the end-to-end mode. Throughput is considered as the time required to transmit a certain number of bytes.

Four different scenarios are used to evaluate our cross-layer design communication architecture (*line scenario*, *cross scenario*, *parallel scenario*, and *grid scenario*, as shown in Fig. 3. To compare the transmission time, the paths in every scenario have 7 hops each. Data of 20 bytes or 1000 bytes are transmitted. In the line scenario, there is one connection (0→7). In the cross scenario, there are two connections (0→14, 1→13). In the parallel scenario, there are two paral-

lel connections (0→15, 1→14). In the grid scenario, there are three connections, which end all on node 0. The connections are routed with the shortest path first algorithm. Every link is weighted equally. Thus, there are nodes (9, 18), which hold two connections. We measure the transmission time of each connection separately. In all scenarios, no energy-saving functions such as duty cycles are implemented, because the focus is on the transmission performance.



(a) Cross Scenario (b) parallel scenario (c) Grid Scenario

(d) Line Scenario

Fig. 3. Evaluation Scenarios.

For our simulation, we use an as realistic as possible radio model. According to the CC2420 manual [13] and the Castalia Simulator [14], we tuned the following values: carrierFrequency: 2.4E+9 Hz; bit-rate: 250 kbps; bandwidth: 10000 Hz; pathLossAlpha: 2; sensitivity: -95 dBm; thermalNoise: -110 dBm; dataLatency: 0.002 ms; CarrierSenseLevel: -77 dBm; transmission power: 1mW. The error rates between the nodes are very high, but reflect a real life scenario. In the line scenario, the average error rate between two neighbor nodes is around 20-25% according to previous measurements [3]. In the cross and square grid scenarios, the error rates are higher, because more nodes interfere with each other, especially in high traffic situations. The global buffer for the packets is limited to 5. In the explicit ACK mode, the MAC protocol has 3 retransmission attempts in the failure case. In the implicit ACK mode, a retransmission is retried twice. We use TCP Reno with a TCP window of 312 bytes. H2HR is configured as follows: the number of retransmission attempts is 6. Detecting a congestion, it waits chosen randomly between 4 ms and 11 ms. After detecting interferences H2HR, waits between 3 ms and 6 ms. Because the length of the MAC frame is limited to 128 bytes, the payload of a TCP packet is 78 bytes long and of a UDP packet is 90 bytes long. In the 20 bytes scenario, one packet is transmitted by UDP and RMST, and 5 packets by TCP. The 1000 bytes scenario requires 29 packets for TCP (including 13 data packets), 12 packets for UDP, and 11 packets for RMST. 50 different simulation runs were executed.

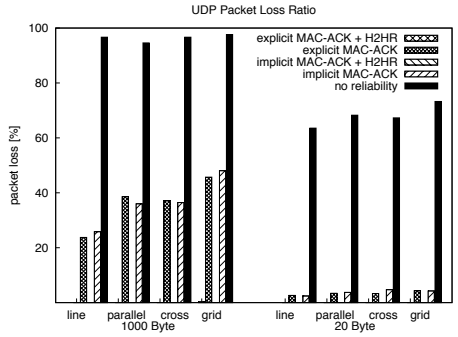


Fig. 4. Packet Loss Ratio for UDP with and without Reliability Mechanisms.

Fig. 4 shows the packet loss ratio of UDP packets transmitted in the four scenarios with a stream of 20 bytes and 1000 bytes. The percentage values are calculated over the absolute number of transmitted packets for all simulation runs. Without any reliability mechanism, the packet loss ratio is very high. When transmitting 1000 bytes, approximately 94% to 98% of packets are lost. When transmitting 20 bytes, the packet loss ratio is between 63% and 73%. One packet leads to less interferences. The packet loss ratio decreased when the simple MAC reliability mechanisms without H2HR support is used. Using overhearing, the loss ratio is higher, because there are only two retransmission attempts. Due to less interferences between the nodes, the packet loss ratio in the line scenarios is lower than in the grid scenarios. Using H2HR in collaboration with the MAC protocol decreases the packet loss ratio dramatically. The probability of successful delivery of a packet is almost 100%, but there is no guaranteed reliability.

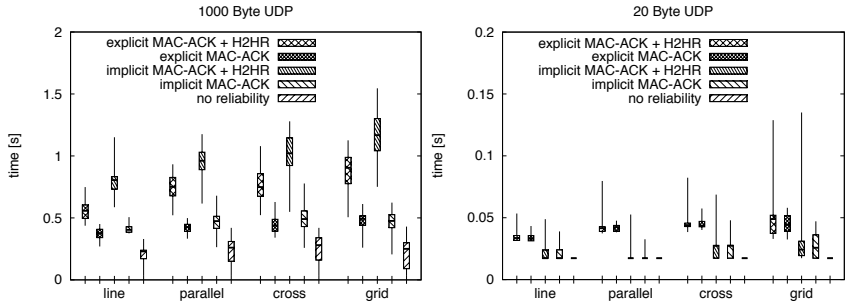


Fig. 5. Time Transmitting Data using UDP.

Reliability mechanisms increase the probability of successfully delivered packets to the next hop, but it decreases the performance. Fig. 5 shows the influence of the reliability mechanism on the transmission time of 1000 bytes and 20 bytes respectively for the four different evaluation scenarios. Without reliability mech-

anisms, the required time to transmit the number of bytes is the shortest, but the packet loss ratio is very high. Just a few of the 12 packets of the 1000 bytes stream are successfully transmitted (usually just one or two). For example, the time required to reach the receiver is lower for packets number 3 and 8 than for packets number 2 and 12. In the 20 bytes scenarios, often no packet is successfully transmitted. In this case, the result is not used for the simulation results. The random backoff time of the MAC protocol has a small influence on the transmission time and adds some delay. Increasing the reliability with H2HR together with MAC retransmissions decreases the performance. Transmitting 1000 bytes takes between 550 ms (line scenario) and 1160 ms (grid scenario) compared to 370 ms up to 490 ms without any reliability mechanism. Using the MAC retransmissions without H2HR collaboration, the transmission time increases approximately 35% to 60%. The increased transmission time is caused by the retransmission attempts of H2HR, and the retransmission attempts of the MAC protocol. TCP tries to guarantee the delivery of packets. Without any additional reliability mechanisms, TCP does not work in typical WSNs. Even connection establishment does not work then. The TCP handshake needs two transmissions from sender to receiver and back (in our scenario this means 14 hops). The probability of a packet loss in the network is very high.

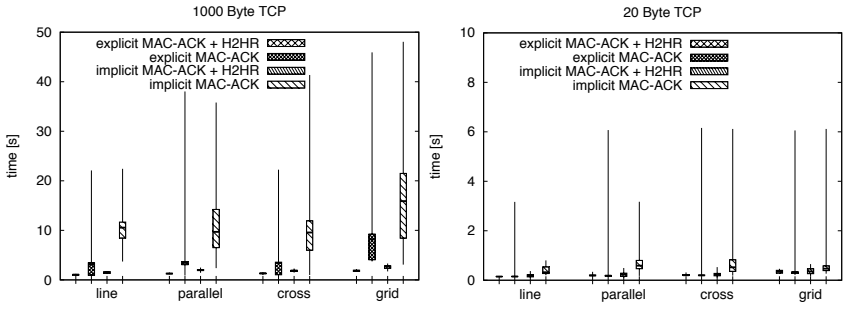


Fig. 6. Time Transmitting Data using Pure TCP without TSS.

Fig. 6 shows the influence of H2HR on the transmission time for a TCP connection. We measured the time interval between the time a connection has been established and closed after the last packet has been transmitted successfully. We compared the influence of using H2HR with both ACK modes. In every scenario, transmitting 1000 bytes or 20 bytes using H2HR decreases the transmission time dramatically. In no case, a TCP end-to-end retransmission is necessary. The outliers in the boxplots without using H2HR occurred because of one or more TCP retransmissions. Each TCP retransmission costs approximately 3 seconds, which is the default value for a retransmission timer used by TCP Reno.

Fig. 7 shows the influence of having TSS as additional protocol. In general, using TCP with TSS is much faster than pure TCP. It optimizes the connection establishment. If H2HR is not used, it improves the performance by retransmit-

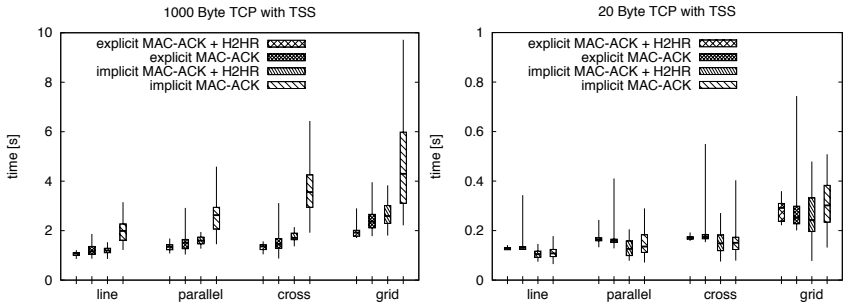


Fig. 7. Time Transmitting Data using TCP with TSS.

ting lost packets within 1.5*RTT [2] by an intermediate node. Using H2HR with pure TCP and TCP with TSS has similar results. Using TSS smooths the outliers by preventing end-to-end retransmissions. Especially in scenarios with high traffic (1000 byte grid scenario for example), TSS has a positive influence on the transmission time. Because H2HR can react on interferences and congestion much faster than TSS or TCP, the influence of H2HR is much stronger. This can be seen in the outliers in Fig. 7(b). In these cases, the MAC protocol collapses and TSS intervenes (if H2HR is not active). If there are no problems, implicit ACK is faster then explicit ACK. No ACK frames have to be transmitted.

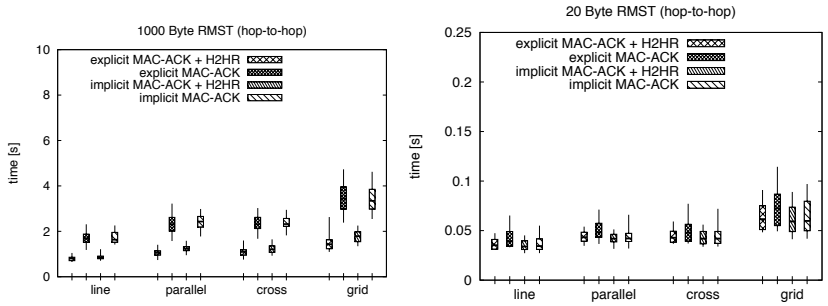


Fig. 8. Time Transmitting Data using RMST in the Hop-to-Hop Mode.

Fig. 8 shows the measurements for the RMST protocol in the hop-to-hop mode. In the 20 bytes scenarios, there are almost no problems with interferences and congestion. H2HR has no influence, all problems are solved by the MAC protocol. The performance is almost as good as using UDP. It is much better than using TCP with TSS, because of additional packets for the connection establishment and the positive ACK frames. In the 1000 bytes scenarios, H2HR and the reliability mechanisms of RMST have to intervene. If H2HR is not active, RMST handles the retransmissions. This takes a long time, especially in the grid

scenario with a lot of retransmissions caused by congestion. In the scenarios with less interferences and congestions, implicit ACK is faster than explicit ACK, because in the implicit ACK mode, no ACK frames have to be transmitted (the same has been observed for TCP with TSS, see Fig. 7). In the grid scenario, there are interferences and congestion. Node 9 and 18 hold 2 connections each. The third connection produces additional interferences (hidden node problem). Thus, packets have to be retransmitted (by the MAC protocol). The implicit ACK mode is slower than the explicit ACK mode, because the retransmission costs much more time (due to the random backoff time).

Fig. 9 shows the comparison of the several transport protocols (UDP, pure TCP, TCP with TSS, RMST end-to-end mode, and RMST hop-to-hop mode). H2HR is used in combination with explicit ACK mode. UDP has the best performance in all scenarios. The reasons are clear: no connection establishment and no ACK frames are required. With H2HR, the successful delivery of all packets is very probable (see Fig. 4). But unlike to the reliable transport protocols, this is not guaranteed. Using UDP, there are outliers in the cross, parallel and grid scenarios. In these cases, the MAC protocol could not handle the situation (congestion, interferences) and H2HR retransmits the lost packets. RMST in both modes has a very good performance, because it uses negative ACKs and has no connection establishment. TCP with TSS performs better than pure TCP, because TSS prevents TCP end-to-end retransmissions (in case H2HR collapses) and optimizes the connection establishment. In the 20 bytes scenario, UDP and RMST have almost the same performance. The MAC protocol (or H2HR) can handle all situations and no retransmissions (end-to-end or hop-to-hop) are necessary in RMST. Because of the connection establishment and positive ACKs, the performance of pure TCP and TCP with TSS is significantly lower. In the 20 bytes scenario, there are less congestion situations and thus, the increased transmission time is caused by connection establishment. With TSS, the connection establishment and the acknowledgment handling is optimized. Generally we can say, with higher traffic (bigger packets to transmit), TCP with TSS has a similar performance as RMST. With lower traffic (just one packet to transmit), RMST is much better and performs similar to UDP. Bigger packets are typical for management tasks, e.g., code update.

6 Conclusion

In this paper we presented a cross-layer designed communication architecture containing a 802.15.4 conform beacon-less MAC protocol, the Hop-to-Hop Reliability (H2HR) protocol and TCP Support for Sensor Nodes (TSS). These protocols collaborate via a cross-layer interface. The H2HR protocol is harmonized with the beacon-less 802.15.4 MAC protocol and uses the acknowledgment mechanisms implemented by the MAC protocol. In general, hop-to-hop reliability mechanisms affect the performance of TCP, UDP, and RMST in WSNs. In case of UDP, H2HR increases the ratio of successfully delivered packets dramatically but at the expense of a higher transmission time. In case RMST, (in both

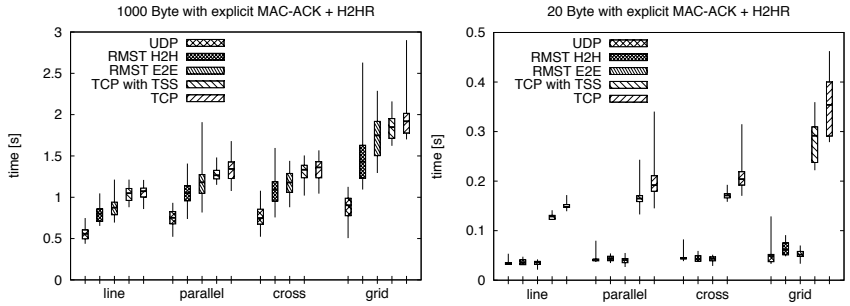


Fig. 9. Comparing the Transport Protocols using Explicit ACK and H2HR.

modes), we showed that using H2HR produces the best results regarding the transmission performance. In case of TCP, the collaboration of H2HR and TSS has the strongest effect on the performance.

References

1. A. Dunkels, T. Voigt, J. Alonso, H. Ritter, J. Schiller: Connecting Wireless Sensornets with TCP/IP Networks. *WWIC'04, 143-152, Frankfurt/O., Germany, Feb'04.*
2. T. Braun, T. Voigt, A. Dunkels: TCP Support for Sensor Networks. *WONS'07, 162-169, Obergurgl, Austria, Jan'07.*
3. M. Anwander, G. Wagenknecht, T. Braun: Management of Wireless Sensor Networks using TCP/IP. *IWSNE'08, 1-8, Santorini Island, Greece, Jun'08.*
4. H. Balakrishnan, S. Seshan, E. Amir, R. H. Katz: Improving TCP/IP Performance over Wireless Networks. *Mobicom'95, 2-11, Berkeley, CA, USA, Nov'95.*
5. C. Barakat, E. Altman: Bandwidth Tradeoff between TCP and Link-level FEC. *Computer Networks, 39(2):133-150, Jun'01.*
6. Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, M. Gerla: The Impact of Multihop Wireless Channel on TCP Throughput and Loss. *INFOCOM'03, 1744-1753, San Francisco, CA, USA, Apr'03.*
7. A. Dunkels, T. Voigt, J. Alonso, H. Ritter: Distributed TCP Caching for Wireless Sensor Networks. *MedHocNet'04, Bodrum, Turkey, Jun'04.*
8. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva: Directed Diffusion for Wireless Sensor Networking. *IEEE/ACM Transaction on Networking, 11(1):2-16, Feb'02.*
9. F. Stann, J. Heidemann: RMST: Reliable Data Transport in Sensor Networks. *SNPA'03, 102-112, Anchorage, AK, USA, May'03.*
10. C. Y. Wan, A. T. Campbell, L. Krishnamurthy: PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. *WSNA'02, 1-11, Atlanta, GA, USA, Sep'02.*
11. V. Srivastava, M. Motani: Cross-Layer Design: A Survey and the Road Ahead. *IEEE Communications Magazine, 43(12):112-119, Dec'05.*
12. OMNeT++: Discrete Event Simulation System, *http://www.omnetpp.org.*
13. CC2420: Datasheet for the Chipcon CC2420 2.4 GHz IEEE 802.15.4 compliant RF Transceiver, *Online, Jan'09.*
14. H. N. Pham, D. Pediaditakis, A. Boulis: From Simulation to Real Deployments in WSN and Back. *WoWMoM'07, 1-6, Helsinki, Finland, Jun'07.*