Management of Wireless Sensor Networks using TCP/IP

Markus Anwander, Gerald Wagenknecht, and Torsten Braun
Institute of Computer Science and Applied Mathematics
University of Bern, Switzerland
Email: {anwander, wagen, braun}@iam.unibe.ch

Abstract— **To allow remote management of heterogeneous wireless sensor networks (WSNs), the WSNs should be connected to the Internet. To overcome this problem, we propose a communication infrastructure, which includes a wireless mesh network (WMN) operating as a backbone network. In order to realize such interconnection between the WSN and an external network without any proxies or middle-boxes, we propose to use TCP/IP as the standard protocol for all network entities, e.g., for configuration and uploading application code to the sensor nodes. We present a cross layer designed communication architecture which contains a MAC protocol, TCP/IP, and a protocol called TCP Support for Sensor Nodes (TSS). The MAC protocol implements the MAC layer of nonbeacon-enabled personal area networks (PANs) defined in the IEEE 802.15.4 standard for peer-to-peer topologies. TSS is located between TCP and IP and implements mechanisms such as caching and local retransmission of TCP data packets, local TCP acknowledgment regeneration, aggressive TCP acknowledgment recovery, congestion and flow control algorithms. We show that our communication architecture increases the performance of TCP/IP in WSNs significantly.**

I. Introduction

Wireless sensor networks (WSNs) consist of a large number of sensor nodes. WSNs are used for various applications such as building monitoring, environment control, wild-life habitat monitoring, forest fire detection, industry automation, military, security, and health-care. For such applications, the WSNs cannot operate in complete isolation. The WSN must be connected to an external network, such as the Internet, through which monitoring and controlling entities can reach the WSN. MARWIS (Management Architecture for Heterogeneous Wireless Sensor Networks) [1] supports common management tasks such as monitoring, (re)configuration, and updating program code in a WSN. To handle large heterogeneous WSNs it is proposed to subdivide it into smaller sensor subnetworks and use a wireless mesh network (WMN) as backbone.

TCP/IP is the de facto standard protocol suite for wired communication. By running TCP/IP in the WSN, it is possible to directly connect the WSN to a wired network infrastructure, without proxies or middle-boxes [2]. While UDP can be used to transmit sensor data to a sink, TCP is used for administrative tasks such as sensor node configuration and updating program code. Because of the limited resources of the sensor nodes, the high packet loss, and the inefficiency in memory and energy consumption of TCP [3], it is rather difficult to implement TCP/IP on sensor nodes. Optimizations, e.g. distributed caching of TCP data packets, local retransmissions, and regeneration of TCP acknowledgment packets can reduce theses problems [4].

Using TCP/IP to manage WSNs requires an optimization and harmonization of the different layers, such as physical, data link, network, and transport layer. Protocols at the different layers have to exchange information across the layers. We present a cross layer designed communication architecture, which enables the use of TCP/IP to manage heterogeneous WSNs. The main parts of the communication architecture are a MAC protocol, which implements the MAC layer of nonbeacon-enabled PANs defined in the IEEE 802.15.4 standard [5] for peer-to-peer topologies, and the TCP Support for Sensor Nodes (TSS) protocol [6], which is a new protocol between TCP and IP. We implemented the whole protocol stack in OMNeT++ [7] using the INET framework and an accurate radio model of the CC2420 radio transceiver (thus the bit error rate is much higher than in the standard radio model). We use the cross layer design paradigm to optimize and harmonize the protocols at the different layers. While the original TSS has been implemented on top of a standard TDMA MAC protocol, we implemented a 802.15.4 conform MAC protocol. Further, we extend the original TSS by caching more than one data packet. The size of the buffer can be configured. The simulations presented in this paper are more detailed and performed in more complex scenarios as in [6].

The paper is organized as follows. Section II introduces the related work regarding transport protocols and using TCP/IP in WSNs. In Section III we present a topology of a heterogeneous WSN and introduce briefly MARWIS, our management architecture for such WSNs. In Section IV the MAC protocol and in Section V the TSS protocol are presented. The developed protocols are simulated and the results are presented in Section VI. Section VII concludes the paper and gives an outlook.

II. Related Work

Many new sophisticated transport protocols for WSNs have been developed. Some of them explicitly focus on reliable transport of data from sensor nodes to base stations, while only a few protocols support data transport to the sensor nodes. The reliability requirements differ significantly depending on the application, since sensor data often have some redundancy and some loss can be tolerated. However, packet loss cannot be tolerated in the case of distributing program code to sensor nodes. Thus using, adapting, and enhancing standard transport protocols such as TCP is an alternative.

Reliable Data Transport in Sensor Networks (RMST) [8], Pump Slowly Fetch Frequently (PSFQ) [9], and Congestion Detection and Avoidance (CODA) [10] are built on top of Directed Diffusion [11], which is a common data dissemination scheme. Event-to-Sink Reliable Transport (ESRT) [12] provides congestion control.

The use of TCP in wireless networks causes some serious performance problems. The end-to-end acknowledgment and retransmission scheme requires expensive retransmissions along every hop on the path between the sender and the receiver, if a packet is dropped. In [13], [4] local caching of TCP segments and local retransmission at intermediate nodes is proposed. A further problem is the energy consumption of TCP [3]. In [14] FEC is used to shield TCP from losses not caused by congestion and helps to improve its throughput. In [15] the authors try to improve TCP performance by establishing the optimal TCP window size. In [16], an energy-efficient protocol called E^{2}TCP is presented. It provides a selective acknowledgment mechanism and uses header compression.

Approaches for management of WSNs and code distribution does not support heterogeneous WSN environments. MANNA [17] is a management architecture for WSNs, which provides functions to establish configurations for WSN entities. TinyCubus [18] is a management and configuration framework for WSNs, which focuses on code distribution and minimizing code fragments to be distributed in a WSN. Multi-hop Over-the-Air Programming (MOAP) [19] is a code distribution mechanism focused on energy-efficient and reliable code distribution.

III. Management Architecture for Wireless Sensor Networks

To connect a WSN to an external network, such as the Internet, additional infrastructure is required. One possible solution is a wireless mesh network (WMN), which works as backbone and supports the connection to the external network. In addition, it supports the connection between the different types of sensor nodes in a heterogeneous WSN. To operate such a WSN the following devices are required: one (or more) management stations, several wireless mesh nodes and a rather high number of heterogeneous sensor nodes. A possible scenario is shown in Figure 1. The sensor nodes of the different subnetworks are depicted as small circles, triangles and rhombuses. The mesh nodes are depicted as large circles. In [1] it is shown that subdivision of a large WSN into smaller sensor subnetworks improves the performance of a WSN by reducing the number of hops in a sensor subnetwork.

The WSN is connected to the Internet using the WMN as backbone. A sensor node can be plugged into a mesh node via serial interface such as USB. Such sensor node operates as gateway between the WMN and the WSN. The mesh nodes communicate via IEEE 802.11b/g/n. The WMN is connected to the Internet via Ethernet or IEEE 802.11. A TCP/IP implementation runs on all entities of the network and

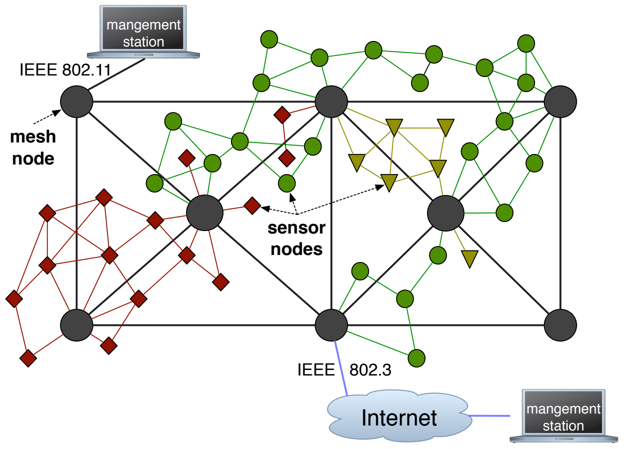


Fig. 1. Topology of a heterogeneous wireless sensor network

is used for reliable point-to-point connections. Every sensor node and mesh node has its own IP address. Thus, the user can control and monitor the WSN and every single sensor node via a management station located in the Internet. Different types of sensor nodes, which build different subnetworks, can communicate to each other. Figure 2 shows the protocols stacks on several entities (management station, mesh nodes, sensor nodes).

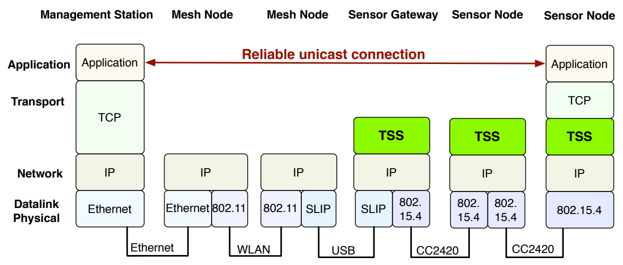


Fig. 2. A reliable unicast connection with TSS

The design of efficient communication involves all layers from physical layer up to the application layer. It is not sufficient to develop a protocol in a single layer, which is put on the top of another single layer protocol. Thus, it is necessary to coordinate and harmonize all affected layers. Important information and states have to be known at all layers immediately. Such a cross layer design can obtain a reliable and efficient data transport within the WSN. We provide a cross layer designed protocol stack on the sensor nodes. This contains a new MAC protocol for 802.15.4 nonbeacon-enabled Personal Area Networks (PANs) and TSS between IP and TCP.

IV. MAC protocol

The MAC protocol at the data link layer depends strongly on the underlying radio transceiver and is specially tailored to it. We are using the CC2420 radio transceiver [20], which implements the physical layer of the IEEE 802.15.4 standard. The proposed MAC protocol implements the MAC layer of nonbeacon-enabled PANs defined in the IEEE 802.15.4

standard for peer-to-peer topologies. To our knowledge it is the first implementation in this way. It provides multihop communication. All nodes are full function devices (FFD). The MAC protocol holds a buffer to store the incoming frames from the lower layer (radio transceiver) and the upper layers (TCP→TSS→μIP). The size of the buffer can be configured. The format of the MAC frame is conform to the IEEE 802.15.4 standard and shown in Figure 3. There are no energy-saving functionalities such as sleeping periods implemented yet.

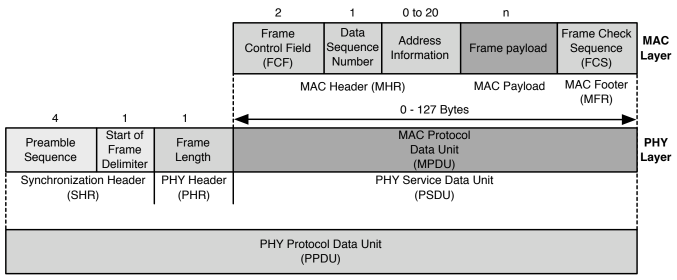


Fig. 3. IEEE 802.15.4 MAC frame.

A. Transmitting and Receiving

The receiving and transmission process is realized as a state machine. The MAC protocol operates according to the states of the CC2420 radio transceiver. We provide two modes for acknowledgments: an explicit acknowledgment (MAC-ACK) and an implicit acknowledgment (MAC-OVERHEARING). The state machines for transmitting and receiving are shown in Figure 4 and work as follows:

1) For transmission of frames the unslotted CSMA-CA algorithm is used.
2) Data coming from the upper layer are written to the TX buffer of the radio transceiver. This step depends on the underlying radio transceiver.This step depends on the underlying radio transceiver.
3) This step contains the unslotted CSMA-CA algorithm. Two variables are maintained: *NB*, and *BE*. *NB* is the number of times the CSMA-CA algorithm was required to backoff while attempting the current transmission. *BE* is the backoff exponent, which is related to how many backoff periods a device shall wait before attempting to assess a channel. The MAC protocol delays for a random number of backoff periods. Via CCA (Clear Channel Assessment) the radio transceiver indicates, whether the channel is free. In case of a busy channel the backoff period is newly calculated and the maximum number of retransmission attempts is checked.
4) The frame stored in the TX buffer is transmitted. By a successful transmission it deletes the frame. It retransmits the frame, if there is no confirmation (MAC-ACK / MAC-OVERHEARING).
5) When a frame is in the RX buffer, the CC2420 radio transceiver confirms it via the SFD and the FIFO pin.
6) The received frame is processed (CRC) and handled according the frame type.
7) In the MAC-ACK mode the *MAC-ACK* is transmitted.
8) *MAC-DATA* is delivered to the upper layer.

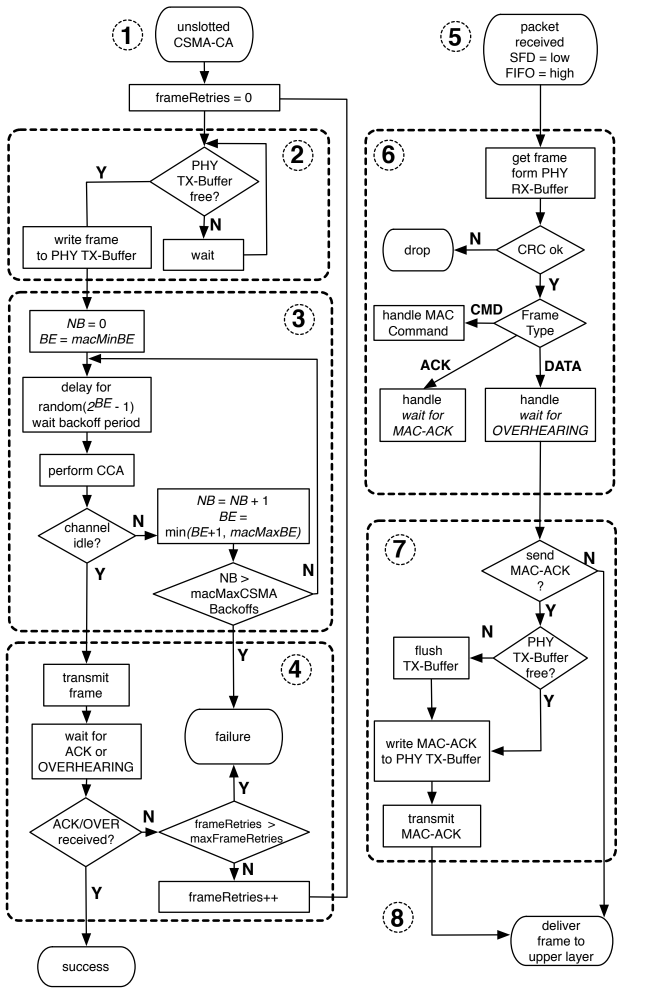


Fig. 4. State machine of the MAC protocol.

B. MAC-ACK vs MAC-OVERHEARING

There are two kinds of acknowledgment modes: MAC-ACK and MAC-OVERHEARING. In case of MAC-ACK the MAC protocol initiates the transmission of an acknowledgment. In case of MAC-OVERHEARING no acknowledgment frame is transmitted. Instead the radio transceiver listens whether the following node forwards the frame. The upper layers get informed about the state of acknowledgment (cross layer information). There are three states:

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

The MAC-ACK mode shown in Figure 5 works as follows:

1) *MAC-DATA* is sent after checking whether the channel is free and whether there is no *Send Stop* active.

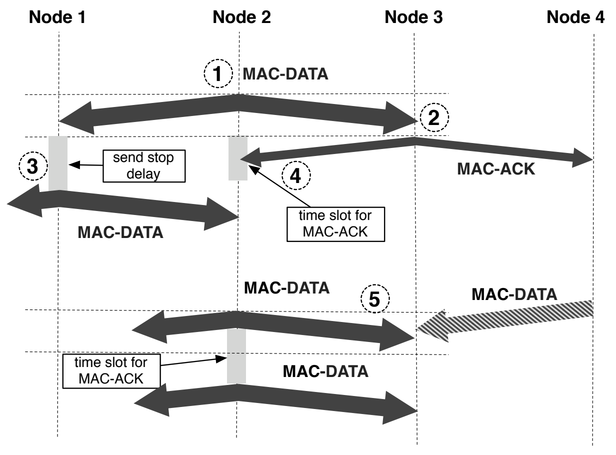


Fig. 5. MAC-ACK

2) After successfully receiving the frame a *MAC-ACK* is transmitted immediately (after 12 symbols).
3) A *MAC-DATA* is received which is not addressed for the receiver. It sets the *Send Stop* timer, thus corresponding *MAC-ACK*s from the receiver to the sender cannot be destroyed. After the timer has expired it can transmit regularly.
4) The *MAC-ACK* is received within the *time slot for MAC-ACK*.
5) Collisions can occur. If CRC checks fail, *MAC-DATA* frames are destroyed. A *MAC-ACK* cannot be transmitted and after expiring the *time slot for MAC-ACK* the frame is retransmitted.

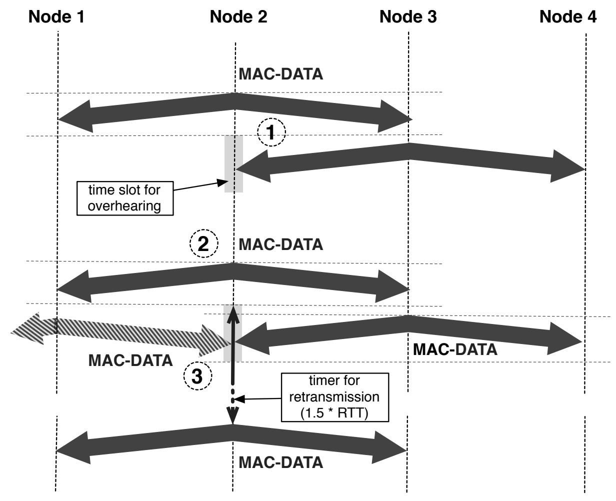


Fig. 6. MAC-OVERHEARING on intermediate nodes.

The MAC-OVERHEARING mode is shown in Figure 6. There exists no explicit acknowledgment. The sender listens when the receiver forwards the frame to the next hop. The mechanism works as follows:

1) The sender transmits a *MAC-DATA* frame. The receiver forwards the frame and the sender overhears it (within the *time slot for overhearing*).
2) Also in the MAC-OVERHEARING mode collisions can occur. Because nodes 1 and nodes 3 cannot hear each other, they transmit frames at the same time. Thus frames collide and overhearing is impossible.
3) On expiration of the *time slot for overhearing* there are no retransmissions. The TSS protocol gets the information that the frame is transmitted, but not acknowledged (not overheard). Retransmission is initiated by TSS.

C. Cross Layer Support of the MAC Protocol

The MAC protocol is responsible to establish a stable link to the neighbor nodes. The communication should be energy-efficient. Therefore, the physical layer has to provide additional information about the transmissions. The radio transceiver provides CCA (Clear Channel Assessment), LQI (Link Quality Index), and RSSI (Receive Signal Strength Indicator). CCA provides information whether the channel is used or not. LQI indicates the quality of a link between two nodes and RSSI provides the signal strength. The MAC protocol decides with this information, if a frame can be transmitted to a neighbor node. The exchange of information between the MAC protocol and the TSS protocol is very important for the reliability mechanisms. The MAC protocol informs the TSS protocol about the transmission state of a frame. It can be transmitted successfully or the transmission can fail (e.g. if the channel was busy). Further, the MAC protocol gives information about retransmissions of segments and the traffic between the nodes in the neighborhood. This information is important for congestion control and avoidance in the TSS protocol. The TSS layer provides the information about the buffer size and the free space in the buffer. The MAC layer uses this information and drops a frame (with payload), if the buffer is full.

V. TCP Support for Sensor Nodes (TSS)

To ensure a reliable transport in WSNs a new protocol between TCP and IP, the TCP Support for Sensor Networks (TSS) [6] protocol has been inserted. This protocol is responsible for the following tasks:

- Caching and local retransmission of TCP data packets
- Improving the TCP acknowledgment mechanism
- Flow and congestion control

TSS supports energy-efficient operation of sensor nodes by reducing the number of transmissions. The TSS protocol holds a buffer to store the incoming frames from the lower layers (radio transceiver→MAC) and the upper layer (TCP). The reliability mechanisms of TSS have an effect, when the network is overloaded (e.g. because of a high load of packets of other protocols or interferences). The MAC protocol must not be too reliable, because in case of unreliable transport protocols (such as UDP) there can be too high overload and delays.

A. TSS Caching Local Retransmissions of TCP Data Packets

In general, TCP is a reliable byte stream protocol designed for wired networks. Reliability is provided by positive acknowledgment with end-to-end retransmissions. In contrast to a wired connection, a wireless network possesses a high bit error rate. This leads to many end-to-end retransmissions. These extra packets reduce the throughput and increase the round-trip-time (RTT). The extra energy for retransmissions also reduces the lifetime of the individual sensor nodes. One of the basic ideas of TSS is to cache a *TCP-DATA* packet in the WSN until the packet is acknowledged (shown in Figure 7):

1) The node caches *TCP-DATA n* and forwards it.
2) The receiver acknowledges *TCP-DATA n* by transmitting *TCP-ACK n+1*.
3) The intermediate node receives *TCP-ACK n+1* and deletes *TCP-DATA n*. In general, it can delete all *TCP-DATA* packets with a lower sequence number than *n*.

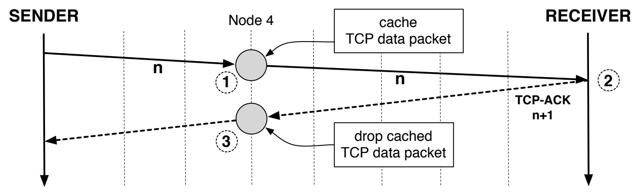


Fig. 7. Basic idea of TSS: caching of TCP data packets.

The case of a packet loss is shown in Figure 8 and works as follows:

1) The node caches *TCP-DATA n* and forwards it.
2) After forwarding a *TCP-DATA* packet, the node expects the acknowledgment (*TCP-ACK*) within 1.5 * RTT. Otherwise, the cached packet is retransmitted.
3) After receiving *TCP-ACK n+1* it deletes *TCP-DATA n* (and all lower *TCP-DATA* packets) from the buffer.

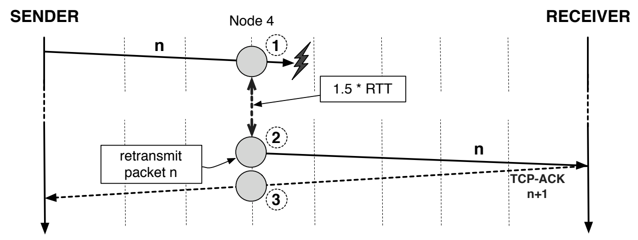


Fig. 8. Basic idea of TSS: retransmission of lost TCP data packets.

Too long retransmission timeouts cause retransmissions by the sender. Simulations in [6] show that a retransmission timeout of 1.5 * RTT is adequate to also retransmit multiple losses of *TCP-DATA* packets without triggering end-to-end retransmission. The RTT is measured between the node and the destination. The RTT coefficient (1.5 in this case) can be configured. Both ideas above assume that only one *TCP-DATA* packet can be cached on every intermediate node and only one packet is in the network at the same time. If it is assumed that every intermediate node can buffer more than one packet the retransmission mechanisms can be extended and optimized. The throughput can be increased, because of more packets can be in the network at the same time. The order of the packets is preserved. Figure 9 shows a packet loss in that case:

1) *TCP-DATA n* and *TCP-DATA n+1* are cached on intermediate nodes and get lost.
2) *TCP-DATA n+2* reaches the receiver, which acknowledges the packet.
3) The node recognizes that *TCP-DATA n* is lost. It forwards the acknowledgment, because it does not have *TCP-DATA n* cached.
4) Intermediate node 3 has *TCP-DATA n* in the buffer, forwards it and deletes the *TCP-ACK n*.
5) This packet is acknowledged and *TCP-DATA n+1* is retransmitted (from node 6). The next packet is requested by transmitting *TCP-ACK n+3*.

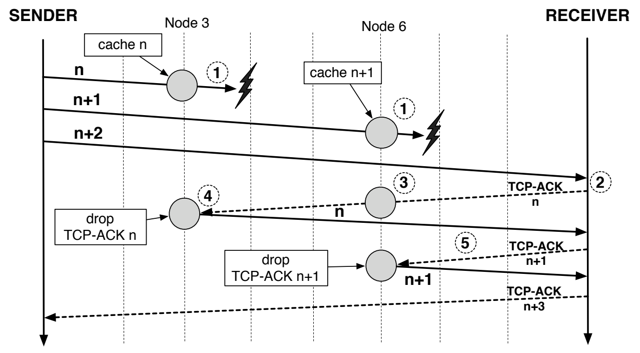


Fig. 9. Packet loss in an extended scenario.

B. TSS Acknowledgment Mechanisms

Because of the limited memory, only a few packets can be cached and an efficient caching algorithm is required. A node caches a packet until it knows that the successor node has successfully received the packet. This can be discovered by receiving an acknowledgment on the data link layer or through overhearing an implicit acknowledgment (shown in Figure 10):

1) Node 5 receives, caches, and forwards *TCP-DATA n* and *n+1*. The MAC protocol informs the TSS protocol that forwarding has failed.
2) Now node 5 tries to retransmit *TCP-DATA n*. After confirmation it deletes it from the buffer. and transmits *TCP-DATA n+1*.

TSS passes only one packet to the MAC protocol (directed through the IP) and waits afterwards for confirmation. There are three states after packet transmission:

- The packet has been transmitted successfully to the next hop, and it is confirmed through *MAC-ACK* or *MAC-OVERHEARING*.
- The packet has been transmitted, but the *MAC-ACK* has not been received or the forwarding of the packet could not be overheard (*MAC-OVERHEARING*).
- The MAC protocol was not able to transmit the packet, because of e.g. channel was busy or some other reasons.

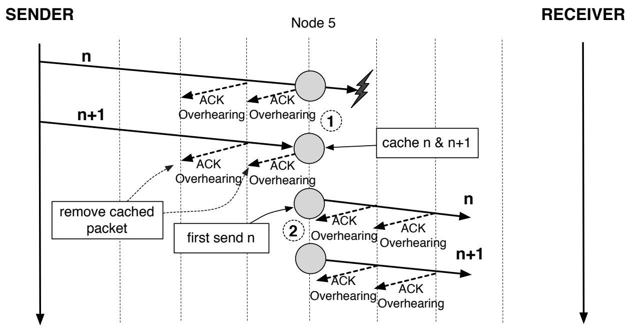


Fig. 10. Cross layer support of the MAC protocol.

According to these three states of the confirmation the node can retransmit the packet or not. In the first case the packet is normally deleted from the buffer. Thus, no retransmission is possible, even if there is no *TCP-ACK* within 1.5 * RTT. If the packet is still in the buffer (according to the basic idea of TSS shown in Figure 7), the packet is retransmitted after 1.5 * RTT. In the case of a negative confirmation (missed *MAC-ACK* or *MAC-OVERHEARING*), we can retransmit the packet immediately after the negative confirmation by the MAC protocol, or we have to wait for 1.5 * RTT and retransmit it then. In the third case, we can try immediately to retransmit the packet or to wait a short time until the channel is not busy anymore. As long as a cached packet has not been successfully delivered to the next hop, later received packets of the same connection are not forwarded. Thus, the order of the packets are not changed.

It happens that a packet has successfully been transmitted to the next hop, but the acknowledgment or the overhearing has not been discovered by the sender (shown in Figure 11). In such case the sender duplicates the packet and retransmits it. To avoid duplicated packets, every node holds a small history list with the identification numbers of received TCP packets and their acknowledgments. In case of a detected duplicate, it is dropped. The history list is the main element of the acknowledgment mechanism. The source and destination address, ports, sequence number, and whether the packet is acknowledged with a *MAC-ACK* or *TCP-ACK* are stored for each packet. With this information TSS can calculate the RTT, and assign *TCP-ACK*s to *TCP-DATA* packets. Further current *TCP-DATA* sequence numbers can be calculated and retransmissions can be recognized. The mechanism is shown in Figure 11 and works as follows:

1) The node caches and forwards *TCP-DATA n*. However, the MAC protocol confirms that it does transmit the packet but does not receive a *MAC-ACK*. Now the node waits 1.5 * RTT for a *MAC-ACK*.
2) The next node forwards the packet and the receiver acknowledges it. *TCP-ACK n+1* is delayed, e.g. because of much traffic in the neighborhood, and does not arrive at node 4 within 1.5 * RTT.
3) The RTT timer expires and the node retransmits *TCP-DATA n*. The next node recognizes with the history list, that it has already received the packet (and successfully forwarded). Thus, it does not forward it and deletes it from the buffer.

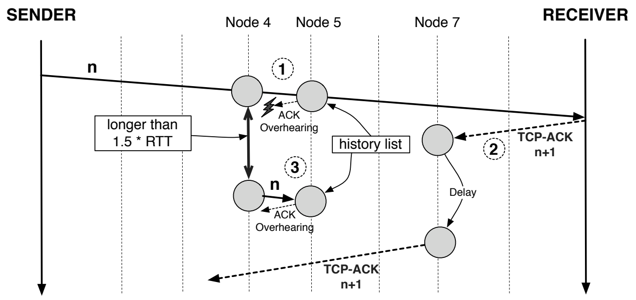


Fig. 11. History list to avoid packet duplications in the WSN

In TSS, the *TCP-ACK*s are very important to estimate the RTT, RTT timers, caching, etc. Experiments in [6] show that the loss of *TCP-ACK*s may have severe impact on the amount of *TCP-DATA* packet transmissions. Two mechanisms reduce the consequences of lost *TCP-ACK*s: local TCP acknowledgment regeneration and an aggressive TCP acknowledgment recovery (shown in Figure 12).

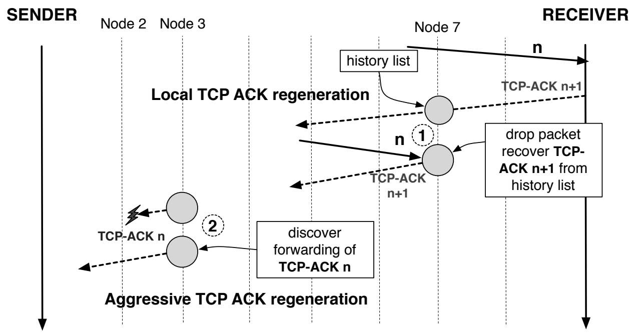

SENDER
Node 2
Node 3
Node 7
RECEIVER
n
history list
TCP-ACK n+1
Local TCP ACK regeneration
n
1
drop packet recover TCP-ACK n+1 from history list
TCP-ACK n+1
TCP-ACK n
2
discover forwarding of TCP-ACK n
Aggressive TCP ACK regeneration

Fig. 12. Two mechanisms to reduce the consequences of lost TCP acknowledgments

The local TCP acknowledgment regeneration is used to drop duplicated TCP data packets, which are already acknowledged by the receiver. Because of the history list, an intermediate node can discover an already acknowledged packet. The duplicated packet is dropped and a TCP acknowledgment with the highest acknowledgment number seen so far is regenerated and transmitted. A scenario is shown in 1) of Figure 12). The aggressive TCP acknowledgment recovery becomes active when a sensor node cannot ensure by *MAC-ACK* that the *TCP-ACK* has been successfully transmitted to the next hop. Using *MAC-ACK*, the retransmission can be enforced directly. It is shown in 2) of Figure 12.

VI. Simulation and Evaluation

We implemented TCP/IP, TSS presented in Section V, and the MAC protocol presented in Section IV in OMNeT++ [7] and run a number of simulations. We compared MAC-ACK and MAC-OVERHEARING as well as we evaluated TCP/IP with and without TSS. We use three different scenarios to evaluate our cross layer design communication architecture. First, we arrange the sensor nodes in a line (*line scenario*, shown in Figure 13) and establishs a TCP connection between node 0 and node 6 which transmits 20 bytes (according to a configuration task) or 1000 bytes (according to a code updating task). No energy-saving functions such as sleeping cycles are implemented, because the focus lies on the transmission performance. The configuration and updating task should be processed as fast as possible.

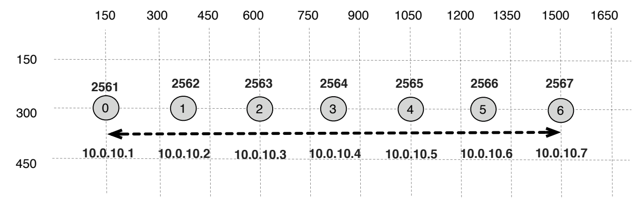
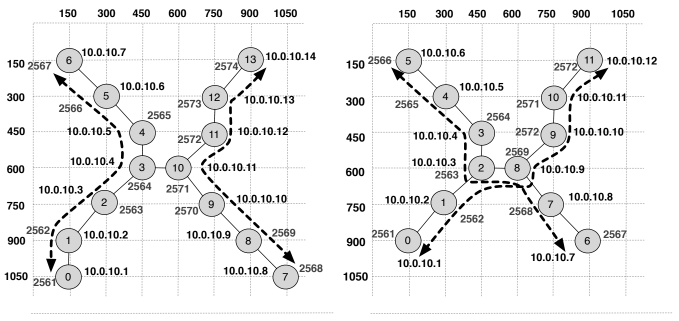


Fig. 13. Sensor nodes arranged in a line.

Second, we arrange the sensor nodes in a cross grid and establish 2 TCP connections shown in Figure 14. In the *parallel scenario* one connection goes from node 0 to node 6 and one connection from node 7 to node 13. The *cross scenario* is also based on the cross grid and establish one TCP connection node 0 to node 11 and one connection from node 5 to node 6. Also in these scenarios 20 bytes and 1000 bytes are transmitted over a connection.

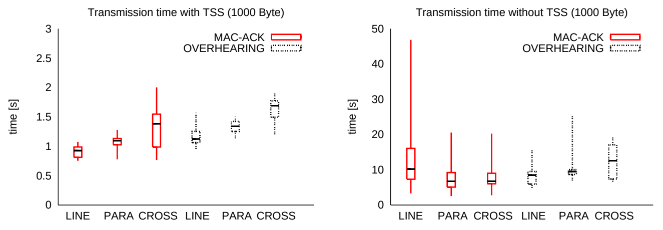


(a) Parallel Scenario (b) Cross Scenario

Fig. 14. Sensor nodes arranged in a cross grid.

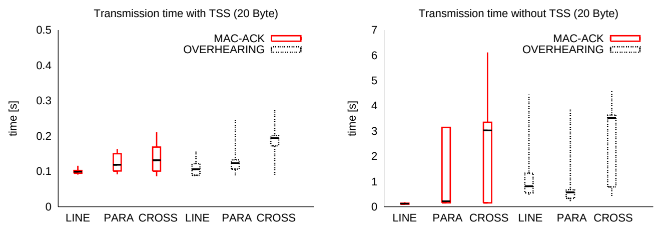
We evaluate the TSS protocol compared with a pure TCP implementation the in line, parallel, and cross scenario using the MAC-ACK mode or the MAC-OVERHEARING mode. We measure the transmission time, the number of the total transmitted bytes, and the number of total transmissions in the network. The transmission time is the duration for transmitting 20 bytes or 1000 bytes over the TCP connection (time between establishing and closing the connection). The number of total transmissions gives the number of sent frames. It means not the number of successfully transmitted frames. The parameters for the MAC-ACK mode are: TCP window size 780, RTT coefficient 1.5 (cf. [6]), TSS buffer size 4, MAC buffer size 2, random backoff time 200 μs, and the number of retransmission attempts 1. The parameters for the MAC-OVERHEARING mode are the same as for the overhearing period of 15ms. The size of the MAC frame is 128 bytes (802.15.4 conform). The average error rate between two neighbor nodes is approximately 15% to 20%, if there are no interferences by other nodes.

Figure 15 shows the transmission time of 1000 bytes (typical code updating task) in the three scenarios with TSS and without TSS using MAC-ACK (3 left points) or MAC-OVERHEARING (3 right points). Transmission time with TSS is 10% of pure TCP in case of the line scenario and 20% in case of the cross scenario. Using MAC-ACK is faster than using MAC-OVERHEARING. Figure 16 shows the same scenarios transmitting 20 bytes (typical configuration task). Here the effect of TSS is even bigger (30 times better performance in case of cross scenario in the MAC-ACK mode), because there are just one TCP-DATA packet transmitted. In the 1000 bytes scenario 129 data packets are needed.



(a) with TSS (b) without TSS

Fig. 15. Transmission time (1000 byte scenario).

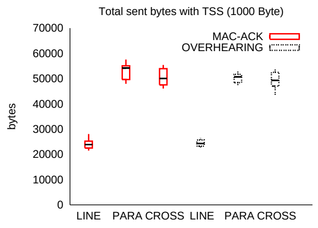


(a) with TSS (b) without TSS

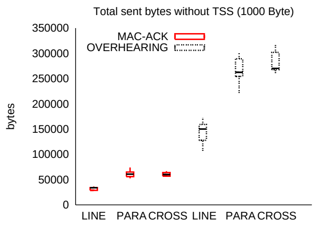
Fig. 16. Transmission time (20 byte scenario).

Figure 17 shows the number of real transmitted bytes comparing TSS with pure TCP using MAC-ACK or MAC-OVERHEARING in the 1000 byte scenario. TSS reduces the number of total transmitted bytes by reducing the of retransmissions significantly. The MAC-OVERHEARING mode needs less transmitted bytes than the MAC-ACK mode using TSS. In case of using pure TCP, MAC-ACK needs less transmitted bytes than MAC-OVERHEARING. The same effect can be seen in the 20 byte scenario shown in Figure 18.

Figure 19 shows the number of transmissions comparing TSS with pure TCP using MAC-ACK or MAC-OVERHEARING in the 1000 byte scenario. Figure 20 shows the results in the 20 byte scenario. The number of total transmissions increases, if TSS is not used. In case of the 20 bytes scenario the effect is stronger, because of overhead of TCP connection establishment.

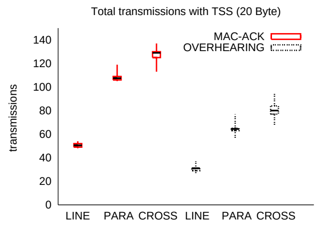


(a) with TSS

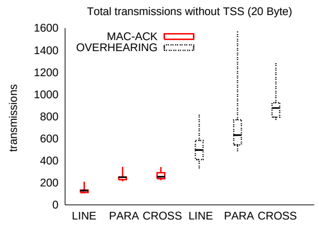


(b) without TSS

Fig. 17. Total sent bytes (1000 byte scenario).

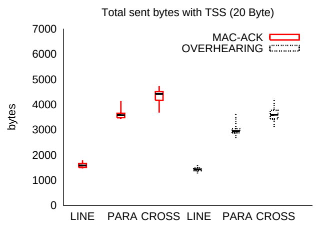


(a) with TSS

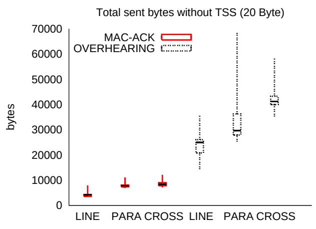


(b) without TSS

Fig. 20. Total number of transmissions (20 byte scenario).



(a) with TSS



(b) without TSS

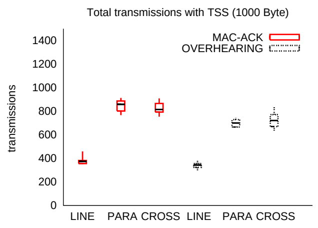
Fig. 18. Total sent bytes (20 byte scenario).

Without the TSS protocol, only about 60% of 1000 byte scenarios, and about 85% of the 20 scenarios are successful. The rest of the transmissions are aborted by TCP timeout after reaching the retransmission limit.

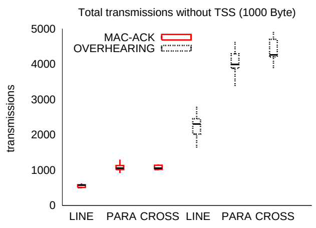
VII. Conclusion and Outlook

We presented a communication architecture based on cross layer design containing a IEEE 802.15.4 MAC protocol in the nonbeacon-enabled mode for peer-to-peer topologies, TCP/IP, and TCP Support for Sensor Nodes (TSS). Reliability mechanisms such as local caching and retransmission of TCP data packets, and acknowledgment mechanisms in the MAC protocol and the TSS protocol are described. We showed that using the additional TSS protocol increase the performance of TCP in WSNs dramatically. The total transmission time of pure TCP is reduced to 10%. Also the the transmitted bytes and transmission count is considerably reduced by TSS.

The next steps including the extension of the protocol with a congestion and flow control algorithm and the design of energy-saving functions. Further the communication architecture should be integrated in the μIP stack of the Contiki operating system to build a real world scenario of a heterogeneous WSN.



(a) with TSS



(b) without TSS

Fig. 19. Total number of transmissions (1000 byte scenario).

References

[1] G. Wagenknecht, M. Anwander, T. Braun, T. Staub, J. Matheka, S. Morgenthaler: MARWIS: A Management Architecture for Heterogeneous Wireless Sensor Networks, *WWIC08, Tampere, Finland, May'08.*

[2] A. Dunkels, T. Voigt, J. Alonso, H. Ritter, J. Schiller: Connecting Wireless Sensornets with TCP/IP Networks, *WWIC04, Frankfurt/Oder, Germany, Feb'04*

[3] M. Zorzi and R. R. Raot: Is TCP energy efficient?, *MoMuC99, San Diego, CA, USA, Nov 1999.*

[4] A. Dunkels, T. Voigt, J. Alonso, H. Ritter: Distributed TCP Caching for Wireless Sensor Networks, *MedHocNet04, Bodrum, Turkey, Jun'04.*

[5] IEEE 802.15.4: Wireless Medium Access control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), *IEEE Computer Society, Feb'06.*

[6] T. Braun, T. Voigt, A. Dunkels: TCP Support for Sensor Networks, *WONS07, Obergurgl, Austria, Jan'07.*

[7] OMNeT++: Discrete Event Simulation System, *Website: http://www.omnetpp.org.*

[8] F. Stann, J. Heidemann: RMST: Reliable Data Transport in Sensor Networks, *SNPA03, Anchorage, AK, USA, May'03.*

[9] C. Y. Wan, A. T. Campbell, L. Krishnamurthy: PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks, *WSNA02, Atlanta, GA, USA, Sep'02.*

[10] C. Y. Wan, S. B. Eisenman, A. T. Campbell: CODA: Congestion Detection and Avoidance in Sensor Networks, *SenSys03, Los Angeles, CA, USA, Nov'03.*

[11] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva: Directed Diffusion for Wireless Sensor Networking, *Transaction on Networking, 11(1):2-16, Feb'02.*

[12] Y. Sankarasubramaniam, Ö. B. Akan, I. F. Akyildiz: ESRT: Event-to-sink Reliable Transport in Wireless Sensor Networks, *MobiHoc03, Annapolis, MD, USA, Jun'03.*

[13] H. Balakrishnan, S. Seshan, E. Amir, R. H. Katz: Improving TCP/IP Performance over Wireless Networks, *Mobicom95, Berkeley, CA, USA, Nov'95.*

[14] C. Barakat, E. Altman: Bandwidth Tradeoff between TCP and Link-level FEC, *Computer Networks, 39(2):133-150, Jun'01.*

[15] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, M. Gerla: The impact of Multihop Wireless Channel on TCP Throughput and Loss, *INFOCOM03, San Francisco, CA, USA, Apr'03.*

[16] L. Donckers, P. J. M. Havinga, G. J. M. Smit, L. T. Smit: Energy Efficient TCP, *AMOC02, Langkawi Island, Malaysia, May'02.*

[17] L. B. Ruiz, J. Nogueira, A. Loureiro: Manna: A Management Architecture for Wireless Sensor Networks. *IEEE Communications, 41(2):116-125, Feb'03.*

[18] P. J. Marrón, A. Lachenmann, D. Minder, J. Hähner, M. Gauger, O. Saukh, K. Rothermel: TinyCubus: A Flexible and Adaptive Framework for Sensor Networks. *EWSN'05, Istanbul, Turkey, Jan'05.*

[19] T. Stathopoulos, J. Heidemann, D. Estrin: A Remote Code Update Mechanism for Wireless Sensor Networks. *Technical Report CENS-TR-30, University of California, Los Angeles, USA, Nov'03.*

[20] CC2420: Datasheet for the Chipcon CC2420 2.4 GHz IEEE 802.15.4 compliant RF Transceiver, *Online, Mar'07.*