TOMAC – REAL-TIME MESSAGE ORDERING IN WIRELESS SENSOR NETWORKS USING THE MAC LAYER

Albert Krohn, Michael Beigl, Christian Decker and Tobias Zimmer
{krohn, michael, cdecker, zimmer}@teco.edu
Telecooperation Office (TecO)
Universität Karlsruhe

ABSTRACT

The ordering of messages according to a given time measure is a largely discussed topic. Nevertheless, wireless or wired systems considering the order of messages do not use the access layer to accomplish this task. In this paper, we discuss the use of non-destructive bit-wise arbitration on the MAC layer for real-time message ordering in wireless sensor networks. With this access method, it is possible to use the measure of elapsed waiting time of a message as an input parameter to the channel access. Through the use of adaptive priorities, messages carrying older information get higher priorities and can therefore gather access first. This hard real-time message ordering mechanism realizes message ordering for one-hop distance mesh topologies. Additionally it is an ideal basis for message-ordering transport protocols. We discuss the system in the parameter context of WiFi (IEEE 802.11a) and Zigbee (IEEE 802.15.4) and successfully implemented the access method on a wireless sensor network.

1 INTRODUCTION

In research on distributed systems and sensor networks we find two different notions of time: *physical* and *logical* time. The use of physical time implies a common physical time base. In contrast to logical time it allows to establish a total ordering of events. Every event in the system is labeled with a globally synchronized time stamp t_j. According to this time stamp events (e.g. j, k) can be order if $t_j \neq t_k$. If $t_j = t_k$ the events happened synchronously on basis of the global clock's granularity. Additionally $|t_k - t_j|$ is a measure for the real time elapsed between the occurrences of the events. In contrast to this, logical time implies only a partial order like *a happened before b* on the set of events. In this partial order if neither *a happened before b* nor *b happened before a* holds true we cannot assume a and b to have happened synchronously. In this case we only know that a and b are unrelated.

Methods for ordering messages temporally using a physical time need a globally synchronized clock running on each device to produce the time stamp. Delay mechanisms or heartbeat protocols are commonly used.

Delay mechanisms like [1] and [2] assume that the network has an maximum delay Δt for the transfer time of a message from producer to receiver. To process all messages in the correct order a receiver has to delay the processing of a message with time stamp t_1 until $t_1 + \Delta t$ to ensure that no message with $t_2 < t_1$ will arrive after processing the message labeled t_1. This is problematic as there is generally no small up-

per bound for the network delay in sensor networks [3] making this technique unsuitable for many applications.

Heartbeat protocols like the Message Bus [4] assume the communication channel between two nodes to have a FIFO property. Each node maintains an ordered list of messages. A message $m_{t_0,i}$ from node i carrying an event occurred at time t_0 is held back until all other nodes in the network have sent messages with time stamps later than t_0. This mechanism guarantees, that there are no more messages waiting at remote nodes with time stamps earlier than t_0. To prevent the network from delaying message delivery to the application infinitely each node has to send so called heartbeat messages after a predefined maximum delay Δt. This introduces the same delay related problems as with the delay mechanisms described above, but additionally introduces a significant message overhead preventing energy efficient operation of wireless sensor networks.

Kay Römer introduced with TMOS a message ordering schema based on physical time that is optimized for the special requirements of wireless sensor networks [3]. The delay for processing of received messages is reduced by organizing the nodes in the network as a logical ring. When a sensor event generates a message, two copies of the message (m, m') are sent onto the ring; one clockwise the other counter clockwise. Again, the link between pairs of the nodes is assumed to have FIFO property. A receiving node places messages according to their time stamps in an ordered list. The message with the smallest time stamp in the list $m_{t_{min}}$ can be delivered to the application when its copy $m'_{t_{min}}$ is received as the FIFO property of the ring network ensures that all messages with $t < t_{min}$ have also been received at this time. TMOS can reduce the total delay until a message can be processed. But it introduces significant overhead by sending each message twice and maintaining a ring topology that is not ideal for fast message distribution.

Ordering messages on basis of logical time is often called *causal ordering*. Methods for causal message ordering are discussed in [5, 6, 7]. For wireless sensor networks causal message ordering is often not suitable, because the elapsed real time between events is of major interest.

Contribution

In this paper, we focus on one issue in the problem area of message ordering. It is the problem of guaranteeing the FIFO property of the channel in a mesh network topology where all nodes are in one-hop distance to each other. We consider the following scenario:

A number of sensor nodes are placed in an area with single hop distance to each other. They share the same channel. In this area, a number of sensor events happen. We now want to ensure, that the messages carrying information on the events are transmitted in the order the events happened in the real world.

The messages carrying information on the events are queued in the MAC layers of the sensor nodes and are not transmitted immediately. There are mainly two reasons for this effect: First, the access method is time-slotted with a low duty cycle (which is the case for nearly all low-power protocols) and the message has to wait until the next possible access period. Then it will compete with all other event-messages that were generated in the time frame since the last possible channel access. The second case is relevant for burst traffic. Bursts of messages are a very common situation in systems where human interaction with an environment generates data traffic. If a lot of events – and consequently a lot of messages – compete for channel access at the same time (or in the same time slot for a slotted protocol), the order of messages cannot be guaranteed due to random access to the channel.

With the MAC layer solution presented in this paper, we propose a mechanism that can guarantee the time-order of message delivery in a single-hop mesh network topology. We guarantee that events that took place until the actual access phase are transmitted in the order of their occurrence in the real world. Additionally, we can easily realize very high time-resolutions (in the sub-μs area) and the mechanism does not demand hardware requirement beyond standard low-power, low-cost radio front ends. This MAC layer builds an ideal basis for real-time message ordering transport protocols.

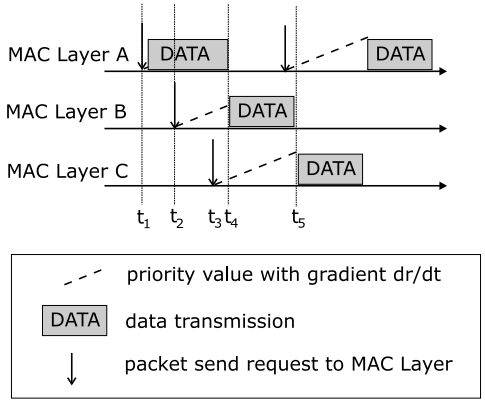

Figure 1: Increasing priorities for waiting packets

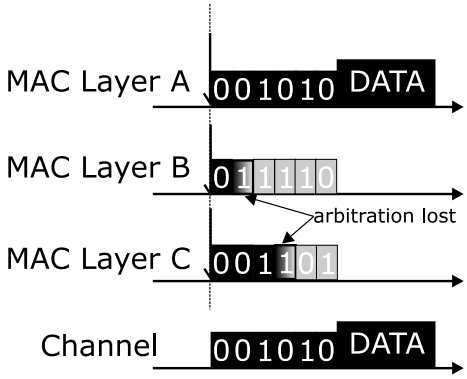

Figure 2: Non-destructive bit-wise arbitration

2 TOMAC Architecture

To realize the time-ordering of messages in the network, we intent to use message priorities r_i to organize the messages order. In this system, each messages is assigned with a priority value once it is produced. With more time elapsed until the actual delivery of the $i-th$ message, the priority r_i will increase. On generation of the packet on the MAC layer, the message starts with the lowest value of priority e.g. $r_i = r_{min} = 0$. Then, the priority is linear increasing with a given gradient e.g. $\frac{dr_i}{dt} = \frac{1}{\mu s}$ up to a maximum value r_{max}. For this example, the priority of the packet is increased by one unit for every elapsed μs that the packet is waiting for transmission in the MAC layer. In figure 1 the system flow for three nodes A,B and C is shown. At time t_1, the MAC layer of the node A receives a packet send request from a higher communication layer. As the medium is free, the packet can be send out immediately. At t_2 and t_3, the MAC layers of nodes B and C get also requests for sending packets. Now the priority system handles the message delivery. The priorities for the packets are linear increased until the channel is free again. At t_4, the channel is free again so either B or C can send their buffered packet. As the priority of the packet in the buffer of node B is higher, it therefore gains the access to the channel first. This system realizes a time-ordered delivery depending on the waiting time of packets. We will now discuss the necessary features of the MAC layer to realize the priority-access.

Time Reference

For the time ordered delivery, it must be assured that each packet (denoted with index i) carries its own priority r_i. This priority must be increased with the according $\frac{dr_i}{dt}$. For consistency, we need a common gradient for the whole priority system, such that $\frac{dr_i}{dt} = \frac{dr}{dt} = const.$ All participants in the network have to make sure that r_i of each waiting packet is increased correctly until the packets are sent out.
The starting time of the priority counter r_i could also be before the actual internal hand-over to the according MAC layer. If e.g. notifications of sensor events should be transported to a destination in a time-ordered manner, the local priority counter would start counting at the moment when the sensor event is recognized not when the message is handed over to the MAC layer. So, the MAC layer would accept a packet that already carries a certain priority offset and then continue as usual by increasing the r_i with the $\frac{dr}{dt}$. As the priority r_i is also a measure for the total elapsed time since the occurrence of the event, it can be used to determine the absolute time of the event in the past.

Granularity and Total Delay

The granularity of the message ordering using priorities is one of the most important aspects for the design and implementation of the priority system. For high time-resolution of events, the gradient of priorities $\frac{dr_i}{dt}$ must be relatively high. Especially for sensory systems, the time resolution is an important aspect. Assuming a time-granularity of e.g. $10\mu s$, (meaning $\frac{dr_i}{dt} = \frac{0.1}{\mu s}$) the priority r_i of a waiting packet would be 50000 after only half a second wait time in the MAC layer. For a low duty cycle power saving TDMA system such as IEEE802.15.4 [8], the TDMA duty cycle period can be as high as 500s. For such a system, we would have to be able to distinguish between $5 \cdot 10^6$ priority steps. The maximum possible value of a priority that the system must be able to handle is denoted with r_{max}. It is also a measure for the scaling requirements to the *TOMAC* system. If r_{max} is very high, a very high number of priority steps have to be distinguished. The r_{max} of a given setting is determined through two variables: First, the time granularity (reflected in $\frac{dr}{dt}$) and second, the total possible delay (Δt). Δt describes the maximum total possible delay until the transmission of a waiting packet. We obtain:

$$r_{max} = \Delta t \cdot \frac{dr}{dt} \; [steps]$$

CSMA as Existing Solution

Projecting the time-ordering of messages directly into the MAC layer is a novel approach that requires additional functionality on the MAC layer. In wireless sensor networks (also for Zigbee or WiFi), the most common access control is CSMA. In CSMA, the access priority is realized through a random wait time. After the termination of the last message on the medium or the beginning of a new access period in low duty cycles protocols, the nodes with pending packets wait a random time and then send their packet. The node with the smallest wait time wins and can send its packet. The possible number of wait times in 802.11a [9] is limited to 1024 slots; in 802.15.4 it is 31 slots. Therefore, priorities can only be distinguished in 1024 or 31 steps. Additionally, the wait time for all prioritized packets would always be very long (802.11a: 1024 slots). This is very inefficient and artificially produces unnecessary high delays. For IEEE802.15.4 with a slot time of $320\mu s$, each message would be artificially delayed by $\approx 10ms$.

Non-Destructive Bit-Wise Arbitration

To handle fine grained priorities in an access protocol, we need a way to reflect the priority in a parameter of the access control itself. A well known standard for priority control on the MAC layer of a single channel environment is the access protocol of ISO 11898 [10] also known as *controller area network* (CAN). The CAN-bus is a very common standard in the automotive area. In the MAC layer of CAN, a *non-destructive bit-wise arbitration* is implemented. The messages have assigned a message ID. Then, all concurrently sending station use this ID to solve the access by selecting the message with the lowest ID number. This is done in a synchronized binary search tree. The shared medium – in this case a 2-wire bus – has two different states: dominant if a zero-bit is sent to the channel and recessive if a one-bit is sent. The bits of the message-ID are interpreted as dominant (0) and recessive (1) and are sent into the medium by the attached nodes. If a node sends a recessive bit and another node sends a dominant bit at the same time, the station sending a recessive bit loses the channel access and has to perform a packet send retry. In figure 2 we see three stations with a waiting packet on the MAC layer. The message IDs are 001010, 011110 and 001101. After the start of the arbitration, the nodes send their IDs bit-wise on the channel. At the second bit-position, node B loses the arbitration as it notices a dominant bus state where it's own bit was recessive. Station B exits the arbitration and will retry after the remote data transmission when the channel is free again. At bit-position four, station A wins over station C as it sends a dominant bit over the recessive one of C. For CAN, the dominant and recessive states are normally realized by a pull-down bus system. If a recessive bit has to be transmitted, the bus is left open. A dominant bit would pull the bus down.

NDBA for Time-Ordering of Messages in WSN

The main advantage of non-destructive bit-wise arbitration (NDBA) compared to usual CSMA is the scaling. With the binary search tree mechanism in NDBA, the distinguishable amount of numbers (or IDs) is much higher than with CSMA. We simply replace the ID with the binary complement of the priority and then use the NDBA to solve the concurrent access for the node with the highest priority. Assuming e.g. 100 slots for CSMA, the distinguishable number of values for CSMA is $r_{max} = 100$. For NDBA, r_{max} is $2^{100} \approx 10^{30}$. But, for a wireless medium, the NDBA must be technically adopted. In contrast to CSMA, the stations would have to change between rx and tx in every slot – active tx is interpreted as dominant, rx is interpreted as recessive. Additionally, the round trip time must be considered: Each bit must reach each arbitrating node before the next bit follows. This limits the speed of the bits in a CAN-system due to the round-trip time. For the wireless medium, this means that the slot time will be longer than for pure CSMA. Taking the parameters of 802.11a, the slot time for NDBA is $18\mu s$ (see [11] with the same technical requirements) which is the double slot time of $9\mu s$ for CSMA. For other physical layer specifications the slot value of NDBA will typically be ten or more times higher compared to CSMA. This is due to the slow rx-tx and tx-rx turnaround times of cheap radio front ends. Taking the example of 802.11a with $r_{max} = 1024$, the total arbitration time would be $9\mu s \cdot 1024 = 9.2ms$. Assuming now a slot time that is 20 times longer for NDBA, we get $9.2ms/(20 \cdot 9\mu s) \approx 51$ slots of NDBA. But the exponential nature of NDBA can easily compensate with

$$r_{max,NBDA} = 2^{51} >> 1024 = r_{max,CSMA}$$

So even with very slow radio front ends, the NDBA will always scale appropriate. NDBA can handle very high values of r_{max} in reasonable processing time and is therefore very suitable for the time-ordering scheme on the MAC layer.

System Summary

We presented a way to realize message-ordering with the help of the MAC layer. For this purpose, every event-message is assigned a priority r_i that increases with $\frac{dr}{dt}$. If a message transmission is not possible due to a sleep time in a low duty cycle protocol or due to a busy channel, the priority will increase. Then, if different messages compete for the channel access at one time, the one with the highest priority will gain access. We use the binary-search mechanism of NDBA as an underlying method to handle very high numbers of priority steps (typically $> 10^6$). High numbers of priority step arise from low duty cycle TDMA protocols as well as from precise time-resolution of events. To cope with the problem of events that occur simultaneous in the given time-resolution we provide three additional random bits. Those random bits are rehashed for each channel access attempt to avoid dead-locks between messages that carry the same priority due to the identical time elapsed.

3 Implementation and Application

We implemented the idea of dominant and recessive bits (NDBA) as the general medium access method on the wireless sensor nodes *Particle Computer* [12].

We considered to use *TOMAC* to support a sound location system. Multiple receivers with known positions detect an audio signal and then send out messages to inform on the detection of this trigger signal. Assuming an environment with myriad sensor nodes this requires a broad and fast communication channel. In the context of this application message ordering is helpful to improve the performance of such a system and also to reduce the demand for the network.

The principle behind the discussed location system is the use of time-of-flight for the location determination of the sound source. The audio signal detection is more reliable at sensor nodes close to the sound source as the attenuation during the air-propagation of audio reduces the signal quality. Therefore, a good

and fast sound location estimation can be performed when using the information from the nodes near to the audio sound source and ignoring the others. But when using standard random medium access control we are not able to guarantee that the messages from the sensor nodes nearby the sound source are sent first over the medium and have to expect a mean delay of $\frac{number\ of\ nodes}{2} \cdot (average\ message\ length)[s]$.

To overcome this problem we use message ordering to implement a robust position detection method for low bandwidth networking devices. Message ordering ensures that information from those devices that receive the sound signal first are put first on the network communication channel. After a minimum of messages is found for computing the position of the sending source, all other nodes stop the transmission of sound trigger information on the communication channel because they would be superfluous. This method ensures that only the best information are sent and the network is not flooded with unnecessary information which may overload the network for a longer period of time.

In the test implementation for this application, the time granularity was $50\mu s \hat{=} 1.6cm$ and the maximum delay was fixed to $13ms \hat{=} 4.3m$. We dealt with 256 priority levels. We could show that the messages from nodes closer to the audio signal were transmitted first over the wireless channel.

References

[1] Y. C. Shim and C. V. Ramamoorthy. Monitoring and Control of Distributed Systems. In *In First Intl. Conference of Systems Integration*, Morristown, USA, 1990.

[2] M. Mansouri-Samani and M. Sloman. Gem – a generalised event monitoring language for distributed systems. *Distributed Systems Engineering Journal*, 4 (25), 1997.

[3] Y. C. Shim and C. V. Ramamoorthy. Monitoring and control of distributed systems. In *In First Intl. Conference of Systems Integration*, Morristown, USA, 1990.

[4] Jörg Ott, Colin Perkins, and Dirk Kutscher. The Message Bus: A Platform for Component-based Conferencing Applications. In *Proceedings of CBG2000*, Philadelphia, USA, 2000.

[5] Sridhar Alagar and S. Venkatesan. Causal Ordering in Distributed Mobile Systems. *IEEE Transactions on Computers*, 46 (3):353–361, 1997.

[6] R. Prakash, M. Raynal, and M. Singhal. An efficient causal ordering algorithm for mobile computing environments. In *Proceedings of 16th International Conference on Distributed Computing Systems (ICDCS '96)*, Hong Kong, 1996.

[7] P. Chandra and A.D. Kshemkalyani. Causal Multicast in Mobile Networks. In *Proceedings of 12th IEEE/ACM Symposium on Modelling, Analysis, and Simulation of Computer and Communication Systems (MASCOTS)*, Volendam, Netherlands, 2004.

[8] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Standard 802.15.4-2003.

[9] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Standard 802.11a-1999.

[10] ISO 11898-1. Road vehicles – controller area network (can) – part 1: Data link layer and physical signalling. 2003.

[11] Albert Krohn, Michael Beigl, and Sabin Wendhack. SDJS: Efficient statistics for wireless networks. In *Proceedings of the 12th IEEE International Conference on Network Protocols*, Berlin, Germany, 2004.

[12] C. Decker, A. Krohn, M. Beigl, and T. Zimmer. The Particle Computer System. In *IPSN Track on Sensor Platform, Tools and Design Methods for Networked Embedded Systems (SPOTS). Proceedings of the ACM/IEEE Fourth International Conference on Information Processing in Sensor Networks*, Los Angeles, USA, 2005. to appear.