A MAC protocol for micro flying robots coordination

D. Piguet, J.-D. Decotignie, J. Rousselot
CSEM, Centre Suisse d'Electronique et de Microtechnique
{dpi, jdd, jrt}@csem.ch

Abstract

Unmanned micro-aerial vehicles (UAVs) flying autonomously in a coordinated formation are expected to be able to achieve missions that would be too expensive or even impossible to fulfil with conventional means. Among them typical examples are surveillance and mapping of areas that are either large or/and difficult to reach.

This paper presents the Medium Access Control (MAC) communication protocol design work conducted within the sFly European project[1] in response to the needs of communication between the members of a coordinated swarm of micro-helicopters.

The main design goals are seamless neighbour discovery, reduction of power consumption and hardware size and weight as well as low end-to-end communication delay.

The solution exposed in this paper involves an adaptation of WideMAC [11], a protocol originally developed for IEEE 802.15.4a ultra-wideband wireless sensor networks. Theoretical analysis and simulation results show that this protocol outperforms standard solutions such as IEEE 802.15.4 beacon-enabled regarding neighbour discovery and latency criteria while providing a similar energy conservation performance.

1. Introduction

Communication between flying robots has been studied in a number of papers. Our main contribution is in the power consumption reduction and, to a smaller extend, in size and weight diminution. For that, we borrow some of the concepts from the wireless sensor network field.

We focus on the Medium Access Control (MAC) layer because time constrained communication are local and only imply mobiles that are in direct visibility from each other. In addition, this layer has a dominant role in fulfilling the requirements in terms of critical metrics such as packet arrival success rate, power consumption and latency. In large formations, the network layer (routing) is clearly involved. However, the sFly scenarios do not foresee such large formations. This layer is hence not covered in the paper.

Our objective is to support regular communications between the robots in a swarm. This objective has two facets, regular communication and also discovery of neighbourhood. As exchanging traffic is the main source of power consumption, our objective is to combine neighbourhood discovery with regular information exchange in a single solution that does not generate more traffic than regular information exchange.

This paper is organized as follows. Section 2 details the requirements in terms of information exchange. Then section 3 summarises the state of the art for MAC protocols in the areas that are relevant to the problem. Section 4 gives a list of the protocols that have been selected and gives a first, qualitative, comparison. It is followed by the simulations results of the various solutions in section 5. Then the choice of the proposed solution is discussed in section 6. Finally, the paper is concluded.

2. Information exchange for formation flying

Flying in formation relies on regular information exchange between the UAVs. For dependable operation, the choice was made to avoid relying on a special role such as "master UAV". All UAVs are supposed equal at least with regard to communication. The usage assumptions of EU project sFly define the requirements for this study in terms of network size, traffic pattern and power consumption.

Swarms are expected to be made of three to five micro-helicopters. To allow some margin, the system is designed for 10 nodes. All the nodes join the swarm at the beginning of the mission prior to take-off. Connection establishment overhead must be null or very small. Once the swarm has been formed and the mission has started, no other flying robot can join anymore. However, due to battery exhaustion or any other unexpected incident, a swarm may accidentally loose a member before the end of the mission. All UAV's that are flying in the same area belong to a single swarm. The traffic pattern is made of local broadcasts sent every T_A seconds, with $T_A \in [0.5, 1.0]$. The payload size of each packet is 36 bytes. The application supports a certain degree of irregularity in packet reception.

[1] *The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 231855 (sFly)*

The performance of the proposed MAC protocols will be assessed using the following metrics: application packet arrival success rate, power consumption and end-to-end latency.

3. Previous work

A swarm of micro flying vehicles forms a Mobile Ad-Hoc Network (MANET): the network is self-organised, communications are peer-to-peer, there is no infrastructure and mobility is present. Hence, we focus on MAC protocols for MANET with special focus on robots communications, Vehicular Ad-Hoc Networks (VANET) and Wireless Sensor Networks (WSN).

3.1. Robot communications

Communications between robots has been the object of a number of papers for more than two decades. In [12], the authors describe a protocol based on random access using a form of CSMA in which each node that transmits is forced to use a different packet length. If two of more node transmissions collide as carrier sense is used before sending that means they started nearly at the same time. When a node finishes transmitting a packet, it checks if the medium is still busy. If this is the case, a collision occurred. In such a case, the node waits for a random time before trying again.

Arai and his co-authors [18] use a TDMA approach in which a central leader is selected which is in charge of creating and managing the TDMA rounds. The round start with a slot reserved to the leader in which the leader indicates the reservations. The following slots are used to indicate requests to send or leave. It is followed by a slot in which newly arrived nodes contend for inclusion. The remaining slots are used to send node traffic. Despite its flexibility, this solution is limited in the number of nodes and highly sensitive to collisions caused by foreign traffic.

A better solution is presented in [16]. The approach is still TDMA based in that there is a round in which each robot transmits in its slot. However, all nodes cooperate to establish the round and IEEE 802.11 is used as access protocol which means that in case of foreign traffic, the slot traffic is just shifted later. New nodes may be added up to a maximum. Insertion and removal of nodes is managed in a distributed manner. The main drawback of the approach is the absence of energy saving mechanisms.

In [15], medium access is based on a token passing over a logical ring in which nodes transmit one after. The ring is started by a single master node which is in charge of managing insertion and deletion of nodes. This techniques suffers from the relatively high bit error rate of wireless communications which cause frequent token losses and the long dead times in case of token loss.

SensorFlock [14] is closer to our approach. It is based on wireless sensor network technology and uses IEEE 802.15.4 without beacons (CSMA/CA). The paper does not address energy conservation and is centered on measuring the relationship between received signal strength and distance.

In summary, most of the inter-robot communication protocols address discovery and communication. To our knowledge, none of protocol presented to date explicitly tackle energy conservation.

3.2. VANETs

Vehicular Ad-Hoc Networks are mainly used to improve the safety of road traffic. Such applications share common requirements with micro-flying robots: robust communications and rapid interaction for both connection establishment and data transmission. However, in VANETs, power is not restricted.

Menouar et al. wrote a survey of existing MAC protocols for VANETs in [1]. Among others, reliable broadcasting of time sensitive Life-Safety messages in VANETs is proposed in [3], but its design is closely related to the intended road-safety application by prioritising the messages sent by the vehicle that is mostly in danger. This contradicts the assumption of periodic, equal importance communication between UAV's. ADHOC MAC [4] was conceived for inter-vehicles communication within the European project CarTALK2000 [6]. A powerful feature of ADHOC MAC is its dynamic, distributed TDMA slot allocation mechanism based on an extension of the Reservation ALOHA (R-ALOHA [7]) named RR-ALOHA.

TDMA protocols require the nodes need to be accurately synchronised. In VANETs, this is possible thanks to the presence of a GPS receiver, but this is not the case for the lightweight UAV's considered.

3.3. Wireless Sensor Networks

This work is expected to benefit from the high energy-efficiency, low footprint and connectionless capability of Wireless Sensor Networks (WSN).

A survey of MAC protocols for Wireless Sensor Networks has been conducted by Demirkol et al. in [2] and in the EU WASP project. As in the previous section, TDMA-based protocols are ruled out because of the synchronisation problem. Among the CSMA based protocols, S-MAC [8] is designed so that the nodes belonging to a same cluster share a common, regular sleep-listen schedule and contend for medium access during their active period. More efficient is WiseMAC [9], in which nodes regularly wake-up to sample the air interface for incoming data all with the same period, but at different instants. A sender appends a wake-up preamble in front of its data to make sure that its destination remains awake when it samples the medium. The first time, the wake-up preamble is long enough so that it covers a whole sampling period. Upon acknowledging the packet, the destination informs the source of its sampling instant so

that the source uses a short preamble the next time it sends something. Compared to S-MAC, WiseMAC distributes the traffic of a neighbourhood uniformly over the sampling period and reduces overhearing thanks to the different sampling instants of the nodes. However, preamble length reduction is not possible in the case of broadcast traffic.

Though they are not mentioned in the survey [2], the two MAC options of the IEEE 802.15.4 standard for Wireless Sensor Networks [10] are widely used in existing applications. The simple variant, named "non-beacon enabled" implements a classic CSMA/CA mechanism based on carrier-sensing with an exponential back-off algorithm. The beacon-enabled mode of IEEE 802.15.4 forms clusters made of one coordinator and of several client nodes which are all one hop-away from their coordinator. The coordinator periodically sends a beacon on which the other nodes synchronise. They wake-up to receive the beacon and contend for network access right after it using a slotted CSMA/CA mechanism. The time interval between two beacons is divided into two parts: the active period which follows the beacon and the inactive period during which all the nodes sleep.

WideMAC [11], a MAC protocol designed by Rousselot et al. for ultra-wideband sensor networks is more compatible with broadcast traffic. In WideMAC, the basic principle of WiseMAC is reversed: instead of listening to the medium at their sampling instant, the nodes transmit a beacon and shortly listen for incoming data right after the end of the beacon transmission. Candidates for unicast transmission wake-up, wait for the beacon of the intended destination, then contend for medium access. Once they have caught the destination's beacon for the first time, they know its sampling schedule, thus they are able to save energy by waking-up only at a short time before the beacon is sent. Since Ultra-Wideband radio models do not support carrier sensing, the protocol manages channel access with ALOHA, which performs well in Ultra-Wideband thanks to the inherent robustness of such radios. This protocol looks promising because efficient broadcasting is made possible by appending the data to the beacons. However, the protocol needs to be adapted to the narrow-band radios used in the project. Particularly, CSMA should be used to prevent beacon collisions.

4. Candidate protocols

In this section, the choice of MAC protocols candidates for simulations and comparison is justified and their modifications related to the design goal are explained.

As a convention, let T_A represent the period between two application packets created by the same node.

TDMA protocols are not considered in this study for two main reasons. First, they assume that some form of clock synchronisation exists between nodes. This can be provided by direct communication or by some external means such as GPS. GPS contradicts our power, weight and size requirements. Using local communications to synchronize is both complex and power consuming. The second reason for rejection is the extreme vulnerability of TDMA to interferences from other systems in the same band.

Low-latency and broadcast compatibility requirements make CSMA/CA a natural candidate. Nodes listen all the time, making it highly suitable to broadcast traffic. Latency is kept to a minimum because a sender does not have to wait for a sampling instant or an active period and the rather light traffic that is expected should ensure that the senders do not spend a lot of time in back-off state. The nodes being active all the time, CSMA will give a higher bound of the power consumption and a lower bound of the average end-to-end delay. The CSMA flavour that is chosen is the Non-beacon enabled mode of IEEE 802.15.4, a widely used solution in WSNs.

The interest in reducing power consumption directs toward protocols that allow the participants to the network to be partly inactive during the operation. By combining the advantages of CSMA and of a reduced duty-cycle, the beacon-enabled mode of IEEE 802.15.4 [10] naturally emerges as an interesting candidate. It departs from our wish to avoid having a special node but its popularity makes it a solution to consider. To obtain an efficient solution, we slightly modified the protocol. The node designated as the PAN coordinator appends the data to be broadcasted to its beacon. The beacon period is set to the application period T_A. The other nodes contend for access and listen during the active period. We assume that all traffic from non-coordinator nodes is sent in broadcast. Whether direct communication (i.e. not via the coordinator) is allowed or not is ambiguous in the standard [10], but these modifications are necessary to support efficient broadcasting.

Our third choice is WideMAC [11]. It has been initially developed as a MAC for Ultra Wide Band transmission such as IEEE 802.15.4a. It makes all nodes periodically (period T_W) and asynchronously wake up, transmit a beacon message announcing their presence and listen for transmissions attempts during a contention window T_{CW}. Figure 1 illustrates a single beacon transmission. It starts with a known and detectable synchronization preamble colored in grey and it is followed by a white colored data sequence which announces the node ID. A time-slotted contention window follows, during which the node stays in reception mode and allows it to receive messages.

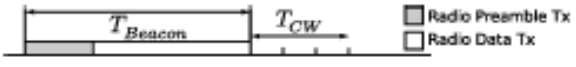

Figure 1. WideMAC beacon

WideMAC [13] is also slightly modified for our purpose. First, the interval between two beacons sending T_w is set such that $T_A = k\ T_w$ with the integer $k > 0$. When the upper layer submits the periodic application packet for

transmission, it is recorded by the MAC layer to be appended to each subsequently transmitted beacon. Every T_A, each node wakes-up and listens for an entire T_w period so that it can receive all the beacons emitted in its range. A node may leave this state temporarily to transmit its own beacon.

Carrier sensing is used prior to beacon transmission to prevent collisions. When the outcome of the CCA is a busy channel, the beacon is rescheduled with a delay chosen randomly in the interval *[0, T_d] with* $T_d \leq T_w$.

In an attempt to reduce the average end-to-end latency, the protocol learns the application schedule and moves its beacon sending instant so that it occurs right after receiving the periodic application packet from the upper layer.

In the rest of this paper, the protocol names refer to their adapted versions.

5. Performance assessment

The protocols have been studied through simulations in MiXiM, a framework of the OMNeT++ network simulator.

For all simulated protocols, the basic simulation setup comprised a playground field of 50 x 50 meters on which the nodes were uniformly distributed. Their relative positions did not change during an entire run like in a stabilised formation flight. During a simulation run, the application of every node generated 300 broadcast packets of size 36 bytes at a period of T_A *= 1 second*. For each simulated value of a protocol parameter, networks of 3, 5 and 10 nodes have been simulated. Each simulation point was run 20 times.

The physical layer model was a ChipCon CC1100 [17] transceiver that operates at a carrier frequency of 898 MHz. The maximum bitrate supported by the physical and MAC models is 250 kbps.

Minimal and maximal backoff exponents of CSMA and IEEE 802.15.4 were set to the default values of 3 and 5 respectively. The beacon interval (BI) of IEEE 802.15.4 beacon-enabled was set to 1 s so that it is equal to T_A. At 250 kbps, this is obtained by choosing a beacon order (BO) of 6. The superframe size is controlled by the parameter SO (Superframe Order), a simulated protocol variable. Simulated values are *1, 2, 3, 4, 5 and 6* so that the corresponding Superframe Durations (SD) ranges between BI/32 and BI.

WideMAC used a uniform backoff with $T_d = T_w/4$, a value chosen arbitrarily (see section 4). Simulations have been run for different values of T_w chosen so that *k = 32, 16, 8, 4, 2 and 1*.

The above selection of variables for IEEE 802.15.4 beacon-enabled and WideMAC allows their comparison at identical duty cycles T_w/T_A and SD/BI. For simplicity, the notation T_w/T_A is used for both protocols from this point. Additionally, in the tables and figures, IEEE 802.15.4 beacon-enabled is shortened to 802.15.4.

5.1. Success rate

The success rate obtained for CSMA is 1.0 for all simulation points. The values for WideMAC and IEEE 802.15.4 beacon-enabled versus the duty cycle T_w/T_A is depicted in Figure 2 below.

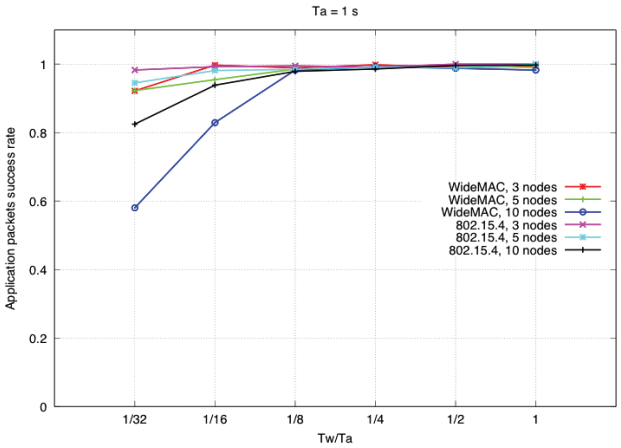

Figure 2. Application packet arrival success rate for WideMAC and IEEE 802.15.4 beacon-enabled. Number of packets per node: 300. Number of runs per point: 20. TA = 1s

For both protocols, the duty cycle T_w/T_A can be safely reduced to 1/8 before the arrival success rate starts to degrade because the medium becomes overloaded. For WideMAC, this corresponds to a value of 125 ms for T_w and to a superframe order SO = 3 for IEEE 802.15.4 beacon-enabled. At his point, a success rate of nearly 100% is reached including for networks of 10 nodes. This gives a safety margin regarding the requirement of supporting 5 nodes.

The curves are coherent in the sense that for small values of T_w/T_A, the performance decreases with the number of nodes. The adaptive exponential backoff mechanism of IEEE 802.15.4 beacon-enabled combined with the possibility to make several transmission attempts for the same packet is more aggressive than the uniform delay used in WideMAC. This increases the chances of IEEE 802.15.4 to "place" its packet on the channel at the expense of a longer delay, possibly higher than T_A if the packet is sent within a next superframe. If WideMAC struggles to send its beacon for more than T_A seconds, the currently waiting application packet is replaced by a subsequent one before the beacon is finally put on the line. The lack of retransmission is certainly a reason for the higher losses encountered by WideMAC. However, the WideMAC behaviour matches more closely the application needs. Indeed, due to the nature of the payload (swarm control algorithm update), a packet loses its validity as soon as the next packet, which contains newer information, is issued. Since the algorithm is tolerant to some update losses, timely delivery is preferred to the higher success rate.

5.2. Power consumption

Figure 3 gives the average node power consumption for the two protocols as a percentage of the consumption of the upper bound reference CSMA/CA: $P_{rx} = 54.124\ mW$.

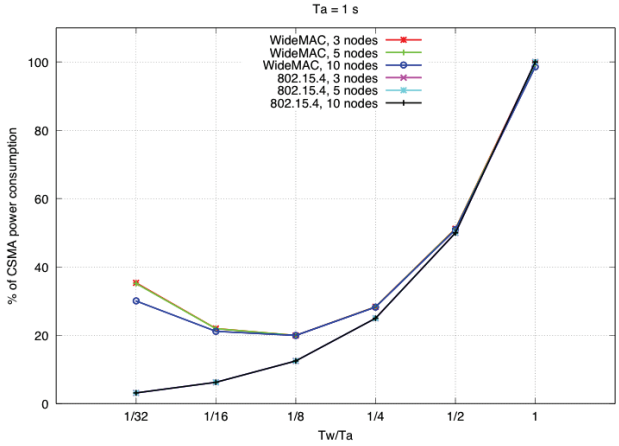

Figure 3. Average node power consumption (% of CSMA/CA) for WideMAC and IEEE 802.15.4 beacon-enabled. Number of packets per node: 300. Number of runs per point: 20. TA = 1s

In IEEE 802.15.4 beacon-enabled, the number of transmissions is nearly independent from the duty cycle. It may decrease a little for small values of T_W/T_A when packets start to be dropped, but this is imperceptible in the consumption simulation results because most of the active time of the nodes is spent in reception. In addition, the power consumption of the radio in receive and transmit differ only from 600 µW. Therefore, the power consumption of the protocol is roughly equivalent to $P_{rx} \times (T_w/T_A)$.

WideMAC differ with IEEE 802.15.4 beacon-enabled in the number of transmissions because in the first protocol, the nodes listen asynchronously once every T_A while they listen at the same period but simultaneously in the second one. This means that the nodes in WideMAC need to repeat their beacon every T_w to ensure that all their neighbours will catch it. The consequence is a higher number of transmissions, thus a higher power consumption. This is shown in Figure 3, in which WideMAC has clearly a lower energy efficiency. For high values of T_w, the higher number of transmissions incurred by WideMAC is not perceptible. When T_w/T_A decreases, its cost becomes visible: the curve diverge from that of IEEE 802.15.4. $T_w/T_A = 1/8$ is an energy efficiency and success rate optimum for WideMAC. At lower values, the cost of packet transmissions takes over that of listening and the success rate starts to degrade. In Figure 3, the curve for WideMAC with 10 nodes shows a consumption smaller than with 3 and 5 nodes for the smallest vales of T_w/T_A. This is likely a consequence of a higher number of delayed and finally dropped beacons due to the higher load on the channel.

5.3. Mean end-to-end latency

As seen in Figure 4, the average end-to-end latency of application packets in CSMA/CA is small when the channel occupancy is low. Backoffs are minimal and the packets do not accumulate and wait in the transmit queue. For the beacon-enabled mode of IEEE 802.15.4, packet submitted to the MAC layer will wait until the coordinator sends its beacon. Then the contention access period is open and the extra delay is similar to that of CSMA/CA as long as the traffic load and the superframe duration are such that the back-off algorithm does not delay some packets to the next active period. In that case, most of the delay is due to the time spent waiting for the beacon. The beacon interval being equal to T_A, the average waiting time should be around $T_A/2$. A similar estimate is reached for WideMAC: if T_W/T_A is small, the delay mainly depends on the listening instant of the receiver, which will take place after an average of $T_A/2$ seconds. If T_W is close to T_A, the destination is listening most of the time but the packets wait to be embedded in the next beacon for up to T_A seconds as well. In WideMAC however, the delay at transmission is reduced using the following optimisation: synchronising the beacon with the application broadcast instant. Note that this optimization could be used in IEEE 802.15.4 in which case the end-to-end latency would be close to the value obtained with CSMA/CA.

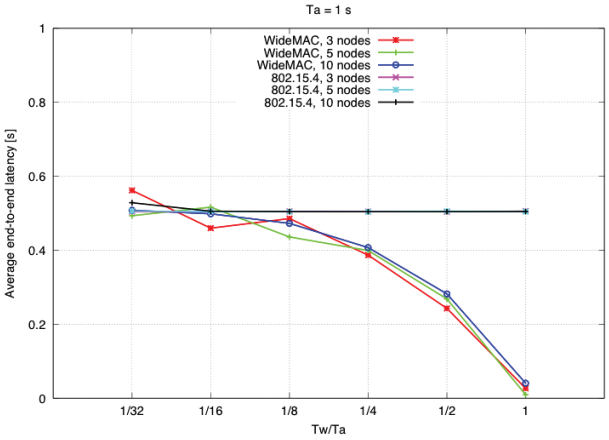

Figure 4. Mean application packets end-to-end latency for WideMAC and IEEE 802.15.4 beacon-enabled. Number of packets per node: 300. Number of runs per point: 20. TA = 1s

6. Discussion

Whilst neighbour discovery is provided seamlessly by WideMAC (periodic beacon) and CSMA/CA (regular application traffic), IEEE 802.15.4 beacon-enabled nodes need to associate to a coordinator before being able to infer the neighbourhood from the application traffic. Moreover, even though the network can be formed before take-off, during the flight, the formation may change and some node may lose contact with the coordinator. If this happens, the node is separated from the network even if it

is still in range of some of the nodes. To re-connect to the network, it will need to re-synchronise with the coordinator but it is unlikely that this operation can take place because the trajectories of the two devices may diverge due to the disconnection and subsequent degradation or loss of control of the device. If WideMAC is used, a node belongs to the network as long as it is visible from at least one other node. Temporary disconnections are less likely and can be fixed quickly and with a higher probability.

Down to a duty cycle of 1/8, all the protocols are able to deliver the data with a high success rate given the traffic pattern and network size foreseen for the sFly project with a sufficient margin. IEEE 802.15.4 is slightly more energy-efficient but it is outperformed by WideMAC in latency performance. Applications could possibly synchronize their packet submission to beacon reception thus reducing the end-to-end delay.

From the three protocols, WideMAC appears to be the most suitable to support the coordinated flight of micro-helicopters. First of all, it supports neighbour discovery by sending periodic beacons. Second, its energy-efficiency is close to that of IEEE 802.15.4 beacon-enabled and, if the optimisation is used, it is more reactive than 802.15.4. Third, its beacon period T_W is an interesting and simple design parameter: when UAVs size and weight reduction is important, a small value is chosen to optimise the protocol for energy-efficiency. Greater values can be selected when power consumption must be traded-off for lower latency or higher traffic or network size.

7. Conclusion

In this article, we presented an analysis of the possibilities of power consumption and hence size and weight reduction for the ad-hoc communication solution embedded on micro unmanned aerial vehicles dedicated to autonomous formation flying. Our main contribution is the adaptation of WideMAC [11], a MAC protocol originally developed for ultra-wideband networks. This solution ideally combines low power consumption, low latency, seamless neighbour discovery and connection.

As a future work suggestion, we mention the optimisation of WideMAC for even better energy performance. A possible direction would be to allow the protocol to sleep between two beacons during a listening phase. There the challenge is to keep the node loosely synchronised with its neighbours despite the delays that some beacons may encounter due to positive clear channel assessments.

References

[1] H. Menouar *et al.*, "A Survey and Qualitative Analysis of MAC Protocols for Vehicular Ad-Hoc Networks", *IEEE Wireless Comm.*, vol. 13, n°5, pp 30-35, Oct. 2006.

[2] I. Demirkol *et al.*, "MAC protocols for wireless sensor networks: a survey", *IEEE Comms. Mag.*, vol. 44, n°4, pp 115-121, April 2006.

[3] M. Taha, Y. Hasan, "VANET-DSRC Protocol for Reliable Broadcasting of Life Safety Messages", *2007 IEEE Int. Symp. on Sig. Proc. and Info. Techn.*, pp 104-109, Dec. 2007.

[4] F. Borgonovo *et al.*, "MAC for ad-hoc inter-vehicle network: services and performance", *IEEE Vehicular Technology Conference'03-Fall*, pp. 6-9, 2003.

[5] M. S. Gast, "802.11 Wireless Networks: The Definitive Guide", *O'Reilly*, 2002.

[6] http://www.cartalk2000.net/

[7] W. Crowther *et al.*, "A system for broadcast communication: Reservation-ALOHA", *6th Hawaii Int. Conference on System Science*, pp. 596-603, January 1973.

[8] W. Ye *et al.*, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks", *IEEE/ACM Trans. on Netw.*, vol. 12, n° 3, pp. 493-506, June 2004.

[9] A. El-Hoiydi *et al.*, "WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks", *Algorithmic Aspects of Wireless Sensor Networks, Springer Berlin*, vol. 3121, pp 18-31, June 2004.

[10] 802.15.4-2003 IEEE Standard for Information Technology-Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Net- works (LR-WPANS), 2003.

[11] J. Rousselot, "WideMac: a low power and routing friendly MAC protocol for Ultra Wideband sensor networks". *Proc. of the Int. Conf. on Ultra-Wideband*, pp. 105—108, 2008.

[12] J. Wang, S. Premvuti, "Resource sharing in distributed robotic systems based on a wireless medium access protocol (CSMA/CD-W)", *Robotics and autonomous systems*, vol. 19, No. 1, 1996, pp.33-56.

[13] M. Perez-Vernet, "Wireless communication data link protocol for cooperating and tele-operated mobile robots", *Proc. 11th Mediterranean Conference on Control & Automation*, Rhodos, Greece, 2003.

[14] J. Allred *et al.*, "SensorFlock: An Airborne Wireless Sensor Network of Micro-Air Vehicles", *ACM SenSys 2007*, pp. 117-129.

[15] P. Wilke *et al.*, "Flexible wireless communication network for mobile robot agents", *Industrial Robot*, Vol. 28, No. 3, pp. 220-232, 2001.

[16] F. Santos *et al.*, "Self-configuration of an adaptive TDMA wireless communication protocol for teams of mobile robots", *ETFA 2008*, September 15-18, pp. 1197-1204, 2008.

[17] CHIPCON AS, Oslo (Norway): CC1100 Data Sheet, 2005.

[18] J. Airai *et al.*, "A New MAC Protocol for Robot Communication and Its Performance Evaluation", 7th Int. Workshop on Multimedia Network Systems and Applications, Vol. 7 pp. 743-748, 2005.

On a Reliable Handoff Procedure for Supporting Mobility in Wireless Sensor Networks

Hossein Fotouhi[*], Mário Alves[*], Anis Koubâa[*#], Nouha Baccour[*+]
[*]*CISTER Research Unit, Polytechnic Institute of Porto (ISEP-IPP), Portugal*
[#]*COINS Research Group, Al-Imam Muhammad bin Saud University (CCIS-IMAMU), Riyadh, Saudi Arabia*
[+]*ReDCAD Research Unit, National School of Engineers of Sfax, Tunisia*
{mhfg,mjf,aska,nabr}@isep.ipp.pt

Abstract

Wireless sensor network (WSN) applications such as patients' health monitoring in hospitals, location-aware ambient intelligence, industrial monitoring /maintenance or homeland security require the support of mobile nodes or node groups. In many of these applications, the lack of network connectivity is not admissible or should at least be time bounded, i.e. mobile nodes cannot be disconnected from the rest of the WSN for an undefined period of time. In this context, we aim at reliable and real-time mobility support in WSNs, for which appropriate handoff and re-routing decisions are mandatory. This paper[1] drafts a mechanism and correspondent heuristics for taking reliable handoff decisions in WSNs. Fuzzy logic is used to incorporate the inherent imprecision and uncertainty of the physical quantities at stake.

1. Introduction

We aim at supporting reliable and real-time communications in Wireless Sensor Networks under nodes' mobility. Reliable and real-time mobility support can be associated to patients' health monitoring in a hospital, process/maintenance personnel in a factory floor, mobile robots or human surveillance in homeland security. This concerns both individual nodes and node groups (e.g. body sensor network) mobility – usually dubbed as "physical mobility".

The problem is that current WSN protocols do not permit to fulfil reliability and real-time requirements under physical mobility. Mobility support in WSNs is in its preliminary steps, since the majority of the current WSN applications assume nodes are static. In this line, most WSN protocols (e.g. ZigBee) just support joining/leaving of nodes, leading to unbounded network inaccessibility times, resulting in unacceptable message delays or losses.

Additionally, radio interference, environmental characteristics and nodes mobility turn radio links unpredictable, leading to message error/losses. This is more acute for low-cost low-power nodes operating in an increasingly crowded 2.4 GHz ISM (Industrial, Scientific, and Medical) band (e.g. WiFi, ZigBee, Bluetooth, cordless phones, microwave ovens or video transmitters).

In this context, we have been addressing the design of an optimal handoff procedure, building upon an accurate estimation of the radio link quality between the mobile node (MN) and the surrounding access points (APs, defined as connectivity points to the rest of the WSN, e.g. routers or cluster-heads) and several other important parameters (e.g. traffic load or energy level at the APs). Handoff refers to the process where a mobile node disconnects from one AP and connects to another AP.

The proposed handoff heuristic (Section 3) is based on Fuzzy Logic to combine all these "uncertain" metrics. Section 2 outlines some handoff models. We conclude the paper in Section 4.

2. Related Works on Handoff Models

The most widely used criteria for evaluating handoff in wireless networks are bit-error rate (BER) and received signal strength (RSS) as indicators for deciding whether to handoff to a new region. However, considering the RSS criterion individually can lead to inappropriate or unnecessary handoff decisions, particularly in WSN scenarios (harsh environmental conditions and strong resource constraints). For this reason, other parameters such as signal to interference-plus-noise ratio (SINR), distance, velocity, direction, transmit power and traffic load have also been considered.

The remainder of this section summarizes some of the most relevant methodologies that have been adopted for designing handoff mechanisms.

Basically, there are two major families of handoff decision. The most common models are the standard techniques, which are used in cellular, wireless mesh, WLAN, and 6LoWPAN networks [1,2,3,4]. These protocols build upon the mobile IPv6 mobility management mechanism. The handoff procedure in mobile IPv6 is initiated by predicting node mobility according to

[1] This work has been partially funded by the Portuguese Science Foundation under the CISTER Research Unit (FCT UI 608), by the REWIN FCT project, by the CONET European NoE and by the EMMON European project.

RSS information. The use of this technique in wireless sensor networks is not recommended, since nodes are usually deployed in a harsh environments and low cost radio transceivers and antennas are usually used, at least for large scale WSN scenarios, hence the received signal strength is not stable. Therefore, relying on only one (unreliable) metric may lead to a poor handoff decision.

Some adaptive and heuristic models have been proposed to handle the handoff procedure considering several input parameters. The classification of these models is illustrated in figure 1. Before a detailed description of our approach, we briefly present the following five heuristic models that have been adopted for designing handoff mechanisms.

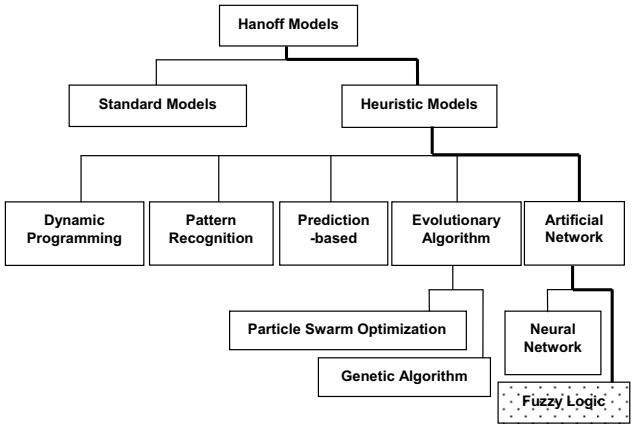

Figure 1. Classes of Heuristic Handoff Models

Dynamic programming allows a systematic approach to optimization. However, it is usually model-dependent and handoff is viewed as a cost optimization problem [5]. RSS samples at the MN are modelled as stochastic processes. The reward is a function of several characteristics such as signal strength, channel fading, shadowing, etc.

In [6], the handoff problem is formulated as a pattern recognition (PR) problem. This technique is based on the idea that the points that are close to each other in a mathematically defined feature space represent the same class of objects or variables. The PR method is an exhaustive method for finding the best possible handoff and is practical only for a canonical (Manhattan) topology but still involves huge computation when applied to generic network topologies.

A prediction based handoff algorithm has been proposed to estimate the future values of handoff criteria, such as RSS. It also shows a trade-off between the number of handoffs and overall signal quality [7].

Some handoff models are based on evolutionary algorithms such as genetic algorithms (GA) and particle swarm optimization (PSO) methods as their optimization technique is used to fine tune the decision parameters. The GA method is an efficient searching technique used for finding the exact or approximate optimization solutions. This method was used in [8] to minimize the sum of weighted distance costs whose complexity is NP-hard.

The other evolutionary algorithm, PSO, is used for handoff decision. It is initialised with a group of random particles (solutions) and looks for an optimum by updating generations. The optimal solutions are called particles which fly through the problem space by following the current optimum particles. In [9], the authors presented a technique for predicting the signal strength value, which aids in providing efficient handoffs in wireless networks and PSO was used to fine tune the weighting function of the handoff decision.

The use of artificial intelligence requires less computational time as compared to the aforementioned searching models, thus seem adequate for time-sensitive applications. Artificial neural networks are one example; they are made up of interconnecting artificial neurons that mimic the properties of biological neurons. These techniques used simplified simulation models (e.g. [10]).

Another example of artificial networks used in handoff is fuzzy logic, which is a multi-valued logic that has been derived from fuzzy set theory to deal with reasoning that is approximate rather than precise. In [11], a handoff decision for heterogeneous networks is identified as a fuzzy multiple attribute decision making problem, and fuzzy logic is applied to deal with the imprecise information.

The use of fuzzy logic is a suitable method for the decision process, because it describes a system intuitively using linguistic variables. In contrast, mathematical optimization approaches typically are not able to cope with diffuse sets, whereas neural networks are highly complex and may have problems with variations and non-deterministic communication characteristics. Moreover, by considering the inherent constraints of wireless sensor networks like limited battery power and the imprecise characteristics of the radio link, the use of fuzzy logic rules seems to be the most efficient heuristic model [12].

3. Proposed Handoff Mechanism

This Section presents the WSN models, a snapshot of the handoff procedure and an insight of the use of fuzzy logic in the handoff heuristics. Then, the two phases of the proposed handoff procedure are described.

3.1. WSN Model

Handoff decision can be made in a distributed (managed at the mobile nodes) or centralized (managed by a single node, e.g. the sink) way. The centralized approach may become less effective for large scale WSNs, as the communication burden between mobile nodes and the central node may lead to unacceptable message delays (for an effective real-time handoff), extra traffic load and energy consumption. For this reason, we opted for distributed handoff management – the mobile nodes take the responsibility of managing handoff, just interacting with the neighbor access points. Figure 2 illustrates our

generic WSN model in which nodes may be static (SN) or mobile (MN) and are somehow associated to access points (APs) that enable WSN nodes connectivity with the rest of the WSN.

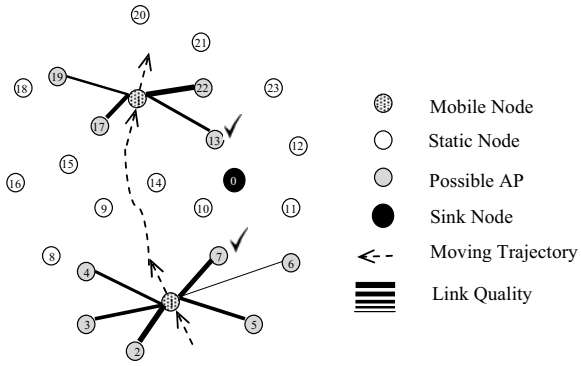

Figure 2. Network Model

We assume different types of time or frequency scheduling between groups/clusters of APs and associated WSN nodes, as to avoid message collisions between adjacent groups/clusters. In this paper we only focus on the handoff heuristic, thus other specifics such as groups/clusters scheduling, (re)routing/(re)addressing and mobility patterns are not under scope of this work. When a mobile node (MN) is moving from the coverage area of an access point (AP_{old}) to the coverage area of another access point (AP_{new}) with a certain speed, the mobile node may learn about the possibility of changing into another cell by a degradation of the signal quality in terms of received signal strength of AP_{old}, and so triggering the handoff mechanism. Depending on the WSN model under consideration using frequency division or time division multiple access between adjacent groups/clusters, probe requests should be sent in different frequency channels or in specific time slots in such a way to guarantee that a MN assesses all the neighboring APs. This handoff model is proposed for a generic network model and it does not focus on a specific model such as FDMA or TDMA. In case of having an FDMA-based model, each node transmits data on a locally unique frequency channel, and in TDMA-based model, nodes communicate using specific time slots.

3.2. Overview of the Proposed Handoff Procedure

As already referred, in most wireless network protocols handoff is based just on the RSS value. In the proposed approach, handoff is based on RSS level, velocity of mobile node, AP depth level (number of hops to sink node), and some other metrics such as traffic load, energy level and link quality value. Any link quality estimation mechanism can be utilized, but the F-LQE (Fuzzy Link Quality Estimator) [13] has been selected, because it has shown a better performance compared to other LQEs as it inherently combines several link quality metrics.

The proposed handoff procedure is composed of two phases: 1) initial assessment of the need for handoff; 2) handoff.

The first phase (described in Section 3.3) aims at deciding whether to do handoff or not, trying to avoid unnecessary handoffs. A MN sends periodic probe messages to its current AP, expecting some acknowledgement message (ACK). It then infers the need for handoff from the RSS average of the acknowledgement messages and from the speed of the MN, if available. If the decision is to handoff, the MN moves to the second phase of the handoff procedure.

In the second phase, the quality of the radio link between the MN and the available neighbouring APs is assessed using the F-LQE link quality estimator [13]. Additionally, the handoff heuristic is enriched by taking into consideration other characteristics of the APs, such as their energy level, traffic load, and depth level.

Figure 2 illustrates a mobile node (1) in two different times - t_0 and t_n. Our example is not concerned with two consecutive handoff procedures other than that it shows two distinct handoff decisions. The link quality is represented by a solid line (the thicker, the better). At time t_0, the mobile node detects six alternative SNs that can be chosen as its next AP. In this case, node 7 and node 2 have more chances to be selected as the next AP, since they have the highest link quality. There are more decision factors in the proposed handoff algorithm such as energy level and traffic load, as it will be discussed in the following sections. For instance, since node 7 is only one hop away from the sink node, it is more likely to be selected as the AP. Now consider time t_n in which the mobile node detects four alternative APs. As it is shown in the figure, node 17 and node 22 have the highest link quality but their location may affect their chances of being selected as the next AP. In contrast node 13 with medium link quality which is closer to the sink has higher priority of being the next AP. These two examples show the importance of different input parameters in various situations. It can be concluded that there is a trade-off between different input parameters and a node with the strongest link quality or smaller number of hops to the sink is not always the best choice.

3.3. On the Use of Fuzzy Logic

Fuzzy Logic is an alternative methodology which can be used in the design of both linear and non-linear systems for embedded control. Fuzzy logic provides a rigorous algebra for dealing with uncertainty. It is expressed in a mathematical discipline invented to express human reasoning with mathematical notations. By this approach the two cases of true and false in conventional algebra are converted to more relaxed conditions, which can help to combine different objectives to achieve an optimal solution. This technique seems to be an efficient alternative for handoff decision making in wireless sensor networks.

A comprehensive theory of fuzzy logic can be found in [14]. The general concept of fuzzy logic is introduced next. By definition, let U be a collection of objects and be called the universe of discourse. A fuzzy set $F \in U$ is characterized by a membership function $\mu_F(u): U \rightarrow [0,1]$ where $\mu_F(u)$ represents the degree (or grade) of membership of $u \in U$ in the fuzzy set F. Therefore, the variables that are used as input parameters are defined by a membership value. This mechanism is used in both phases of the proposed handoff procedure when using fuzzy logic.

3.4. Handoff Mechanism (Phase One)

We define some notations with reference to Figure 3, which shows a handoff from the current AP, referred as AP_{old}, to the future AP, referred as AP_{new}.

The S_{th} level is the threshold value of the RSS to initiate the handoff process. Therefore, when the RSS level of AP_{old}, referred to as RSS_{old} drops below S_{th}, the handoff is triggered. The $S_{\min}$, indicates the minimum value of RSS required for successful communication between a MN and the AP_{old} with a certain probability (let us say 95%). The maximum transmission range of each AP is denoted by a. Hence, as the figure illustrates, the handoff mechanism must be completed before the RSS of AP_{old} drops below $S_{\min}$, i.e., before the MN moves beyond the coverage area of AP_{old}.

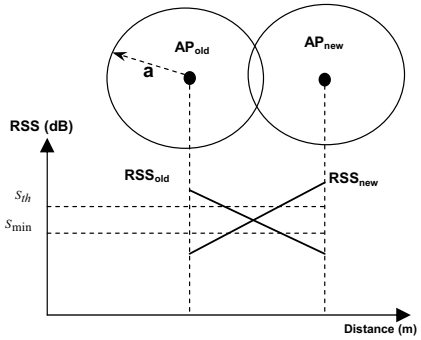

Figure 3. Analysis of Handoff in Phase One

The first phase of the handoff procedure is illustrated in the algorithm of Figure 4. The connectivity between MN and the current AP is assessed by averaging the RSS value from probe acknowledgement messages.

A MN sends periodic (T_{probe}) probe requests which are to be acknowledged by the current AP. Upon the reception of the probe acknowledgements, the MN computes the average of the last θ RSS values ($\overline{RSS}$). Parameter θ should be set low enough to enable a quick assessment of the radio link (the higher θ, the longer it takes) and high enough to attenuate (by averaging) too sudden fluctuations of the RSS. We use a short window to calculate the mean RSS (e.g. θ = 5). The computation of T_{probe}, which is a function of the mobile node's velocity and the radio coverage/overlapping of the APs, is left out of this paper.

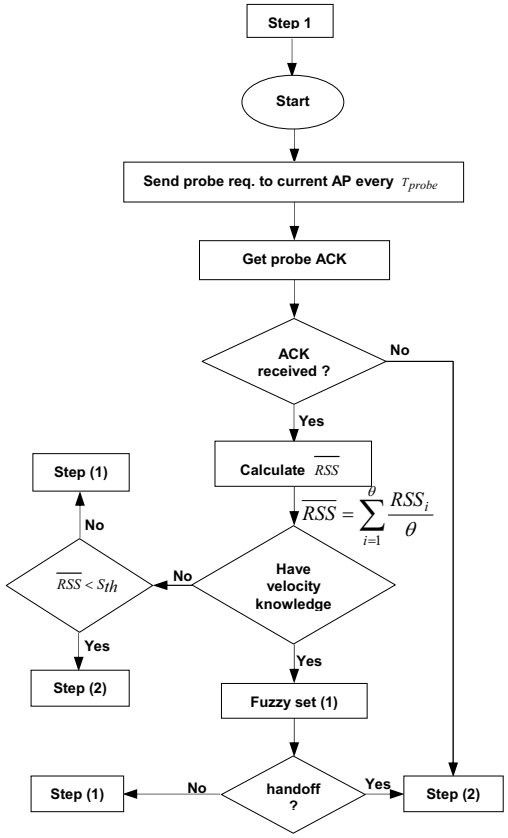

Figure 4. Handoff Mechanism (Phase One)

If there is no information on MN's speed, the $\overline{RSS}$ is compared with S_{th}. If the mean received signal strength has dropped below this threshold then the handoff should be performed, otherwise it continues sending probe requests. In case the MN knows its velocity, either predefined or estimated, a fuzzy logic set getting both $\overline{RSS}$ and velocity values is computed. If the result of this rule indicates to (try to) associate to another AP, then the MN should pass to the second phase of the algorithm.

The basic configuration of the fuzzy logic system is shown in Figure 5 and consists of four principal elements: fuzzifier, fuzzy handoff rule, fuzzy interface engine (handoff decision making unit), and defuzzifier.

The fuzzifier performs a mapping from the observed crisp input space, e.g. the measured RSS, to the membership of the fuzzy set, e.g. high RSS, where a fuzzy set is characterized by a membership function. The handoff fuzzy rule consists of a set of linguistic rules in the form of "IF a set of conditions are satisfied, THEN a set of sequences are inferred". The fuzzy inference engine is a decision making logic which employs fuzzy rules from the handoff fuzzy rules unit to map the fuzzy sets in the input space. Finally, the defuzzifier performs a mapping from

the fuzzy sets to crisp points. The output of the defuzzifier is generally a crisp value, calculated by using fuzzy logic operators.

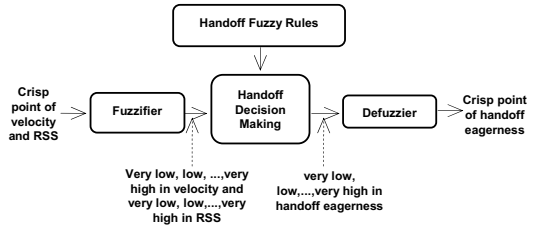

Figure 5. Fuzzy Logic System Unit in Details

Table 1. Fuzzy Rule

Velocity	RSS	Handoff Eagerness
Very low	Very low	Very high
Very low	Low	High
Very low	Medium	Medium
Very low	High	Very low
Very low	Very high	Very low
low	Very low	Very high
low	Low	High
low	Medium	Medium
low	High	Low
low	Very high	Very low
Medium	Very low	Very high
Medium	Low	High
Medium	Medium	Medium
Medium	High	Low
Medium	Very high	Very low
High	Very low	Very high
High	Low	Very high
High	Medium	High
High	High	High
High	Very high	High
Very high	-	Very high

The input fuzzy variable of speed and RSS are assigned to one of the five fuzzy sets, "*very low*", *"low"*, *"medium"*, *"high"* or *"very high"*, which are optionally classified into five levels. This grouping strategy gives more clues on the weakness and strength of input variables and helps generating more accurate output data. Table 1 illustrates the eagerness of performing handoff depending on the velocity and RSS levels. For example, when the value of velocity is "*very high*" and the value of RSS is "*very low*"; this condition indicates that handoff should be encouraged immediately or the handoff eagerness is "*very high*". We define the handoff in cases of having "*high*" or "*very high*" eagerness in output.

3.5. Handoff Mechanism (Phase Two)

By getting handoff permission in the first phase, the MN moves to second phase of the handoff procedure. The handoff decision will be based on a more accurate estimation of the radio link quality (using F-LQE, rather than just RSS) between an MN and all AP in its vicinity, and on AP-specific parameters such as the traffic load, depth and energy level.

Similarly to the first phase, as it is illustrated in Figure 6 and explained previously, probe requests are periodically sent every T_{probe} on available channels or time slots (according to FDMA or TDMA schemes). By receiving probe acknowledgements from neighbouring APs, the algorithm enters the decision making phase.

The process of choosing the best AP between several alternatives can impact WSN performance. Hence, it is important to obtain reliable and accurate link quality estimation in a short time. Link quality estimators (LQEs) have been proved to provide a more accurate and stable information on link quality than just RSS [15]. We opted for F-LQE [13], since it has recently been shown to perform better than existing LQEs. It advocates combining several important link properties to get a holistic characterization of the link. It uses fuzzy logic to estimate the link quality. Therefore, by defining link properties in linguistic terms and performing the fuzzy logic rule, it results the degree of membership of the link in the fuzzy subset of good quality links.

In this design, four link quality metrics of packet delivery, asymmetry, stability and channel quality are considered. The goodness or high quality of a link is characterized by the following rule: "**IF** the link has high packet delivery AND low asymmetry AND high stability AND high channel quality **THEN** it has high quality." By use of and-like compensatory operator of [13], the following equation stands for link i with high quality:

$$\mu(i) = \beta.\min(\mu_{SPRR}(i), \mu_{ASL}(i), \mu_{SF}(i), \mu_{ASNR}(i)) + (1-\beta).mean(\mu_{SPRR}(i), \mu_{ASL}(i), \mu_{SF}(i), \mu_{ASNR}(i)) \quad (1)$$

The membership function $\mu(i)$ in equation (1) represents the membership to the fuzzy set of high quality links and the others like $\mu_{SPRR}(i), \mu_{ASL}(i), \mu_{SF}(i)$ and $\mu_{ASNR}(i)$ indicate the membership functions in the fuzzy subsets of high packet delivery, low asymmetry, low stability, and high channel quality respectively. The parameter β is a constant value in range [0,1]. By considering $LQ(w) = 100.\mu(i)$, the link score range changes to [0..100], where 100 denotes the best link quality and 0 shows the worst. Equation (2) shows the F-LQE value after performing EWMA filter for smoothing:

$$FLQE(\alpha, w) = \alpha.FLQE + (1-\alpha).LQ \quad (2)$$

Where α = 0.9 to provide a stable link estimate, and w is the estimation window, meaning that a node estimates link quality based on each w received packets.

In order to choose the appropriate AP, we consider other criteria apart from link quality estimation. These criteria are energy level (EL), traffic load (TL), and depth level (DL). Each criterion is considered as a fuzzy variable and is supposed to be embedded in the payload of the probe acknowledgement messages. The following

equation shows the membership function of fuzzy handoff (FHO) for mobile node n :

$$\mu_{FHO}(n) = \gamma.\min(\mu_{FLQE}, \mu_{TL}, \mu_{EL}, \mu_{DL}) + (1-\gamma).mean(\mu_{FLQE}, \mu_{TL}, \mu_{EL}, \mu_{DL}) \quad (3)$$

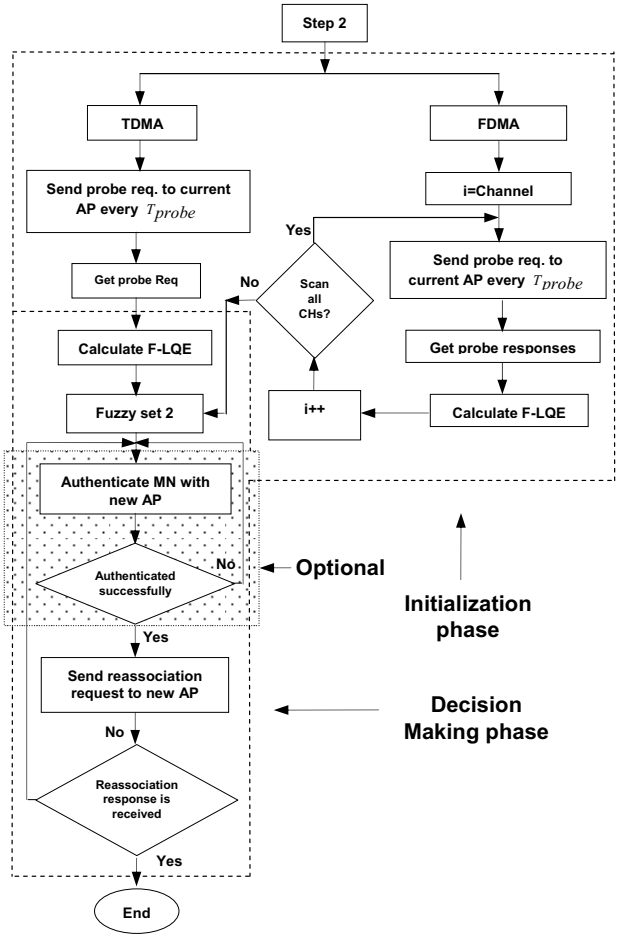

Figure 6. Handoff Mechanism (Phase Two)

The constant value of γ should be defined/tuned according to future simulation/experimental results. Afterwards, an optional authentication phase is performed by sending an authentication request by MN and getting the response from the AP. Finally, the mobile node sends a reassociation request to the new AP. The handoff mechanism ends when the MN receives the association ACK message.

4. Final Remarks

This paper outlines a reliable handoff procedure for supporting mobility in WSNs. A two-phase procedure is proposed that performs handoff decision according to several important metrics, combining them using fuzzy logic.

Next step is to implement, test and validate the proposed handoff mechanism via simulation and experimental models. This will enable to tune the different parameters of the handoff heuristics for an optimal handoff.

We are planning to implement and integrate the proposed handoff mechanism in standard WSN protocols such as ZigBee and 6LoWPAN, to demonstrate its feasibility and efficiency.

5. References

[1] R. Ramjee, K. Varadhan, L. Salgarelli, S. R. Thuel, S. Y. Wang, and T. L. Porta, "HAWAII: a Domain-Based Approach for Supporting Mobility in Wide-Area Wireless Networks," *IEEE/ACM Trans. on Networking*, vol. 10, No. 3, pp. 396-410, June 2002.

[2] S. Pack*, et al.*, "Fast Handoff Support in IEEE 802.11 Wireless Networks," *Communications Surveys & Tutorials, IEEE,* vol. 9, pp. 2-12, 2007.

[3] L. Bo*, et al.*, "A Survey on Mobile WiMAX [Wireless Broadband Access]," *Communications Magazine, IEEE,* vol. 45, pp. 70-75, 2007.

[4] Kim JH, Hong CS, Shon T. A Lightweight NEMO protocol to support 6LoWPAN. *ETRI Journal* 2008; 30(5): 685–695.

[5] A.Z. Melikov, A.T. Babayev, "Refined approximations for performance analysis and optimization of queueing model with guard channels for handovers in cellular networks", *12th IEEE International Conference on Network* 2004, Vol. 29, Issue 9, Pages 1386-1392, May 2006.

[6] Wong, K. D. and D. C. Cox, "A Pattern Recognition System for Handoff Algorithms", *IEEE Journal on Selected Areas in Communication*, Vol. 18, No. 7, July 2000.

[7] P. Bellavista, A. Corradi, C. Giannelli, Adaptive Buffering based on Handoff Prediction for Wireless Internet Continuous Services, *Int. Conf. High Performance Computing and Communications (HPCC)*, LNCS 3726, pp. 1021-1032, 2005.

[8] Chan, T. M., Kwong, S., Man, K. F. & Tang, K. S., Hard handoff minimization using genetic algorithms, Signal Processing, Vol. 82, No. 8, pp. 1047-1058, 2002.

[9] S. Venkatachalaiah, R. J. Harris, and J. Murphy, "Improving handoff in wireless networks using Grey and particle swarm optimisation," *in CCCT*, vol. 5, 2004, pp. 368–373.

[10] Joe Capka and Raouf Boutaba. Mobility Prediction in Wireless Networks using Neural Networks. "*7th IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS 2004)"*, San Diego, CA, USA, October 2004.

[11] W. Zhang, "Handover Decision Using Fuzzy MADM in Heterogeneous Networks," in *Proc. of IEEE WCNC'04*, Atlanta, GA, March 2004.

[12] D. S. Dewey, Fuzzy Logic, 2010. http://www.omega.com/techref/fuzzylogic.html

[13] N. Baccour, A. Koubâa, H. Youssef, M. Ben Jamâa, D. do Rosario, M. Alves and L. Becker, F-LQE: A Fuzzy Link Quality Estimator for Wireless Sensor Networks**,** "*The 7th European Conference on Wireless Sensor Networks (EWSN 2010)*", Coimbra, Portugal, September 17-19, 2010.

[14] L. X. Wang, Adaptive Fuzzy Systems and Control. Englewood Cli_s, New Jersey: PTR Prentice Hall, 1994.

[15] C. Çeken, S. Yarkan, H. Arslan, "Interference aware vertical handoff decision algorithm for quality of service support in wireless heterogeneous networks",in *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Volume 54 , Issue 5 (April 2010), pp. 726-740.

Robust Data and Event Services in Real-Time Embedded Sensor Network Systems*

Krasimira Kapitanova
University of Virginia, USA
krasi@virginia.edu

Sang H. Son
University of Virginia, USA
son@virginia.edu

Seog Park
Sogang University, Korea
spark@dblab.sogang.ac.kr

Abstract

The majority of event detection in real-time embedded sensor network systems is based on data fusion that uses noisy sensor data collected from complicated real-world environments. Current research has produced several excellent low-level mechanisms to collect sensor data and perform aggregation. However, solutions that enable these systems to provide real-time data processing using readings from heterogeneous sensors and subsequently detect complex events of interest in real-time fashion need further research. We are developing real-time event detection approaches which allow light-weight data fusion and do not require significant computing resources. Underlying the event detection framework is a collection of real-time monitoring and fusion mechanisms that are invoked upon the arrival of sensor data. The combination of these mechanisms and the framework has the potential to significantly improve the timeliness and reduce the resource requirements of embedded sensor networks.

1. Introduction

With the continued miniaturization and growing computation power of wireless sensors, the deployment of real-time embedded systems using such devices has significantly increased. These systems use sensors to monitor the physical world and provide appropriate reaction and control over it. The scale of such interactions is very wide, ranging from resource-constrained sensor devices to global-scale networked monitoring systems, and these systems have the potential to transform the way people interact with and control the physical world.

Real-time embedded sensor network systems are used for variety of applications such as infrastructure monitoring, medical systems and smart healthcare facilities, surveillance applications, and environmental monitoring and control. Regardless of the specific application, these systems should be able to detect when particular events of interest occur. In embedded sensor networks, most events are not binary. Instead, they are based on sensor data fusion using noisy sensors deployed in complicated real-world environments. Several excellent low-level mechanisms and protocols to collect, transport, and perform data aggregation on the raw sensor data have been developed. However, systematic solutions that allow these sensor network systems to provide real-time data processing using data fusion from heterogeneous sensors and subsequently detect complex events are still lacking. Recent work [8] addresses confident event detection in sensor networks. The proposed detection mechanism is centralized and nodes are trained offline to recognize specific events. However, such training is often not feasible, e.g. when the detected events present danger (explosions, fires, collapsing of buildings), or cannot be easily reproduced (earthquakes, volcano eruptions).

There are a number of requirements that an event detection service for embedded sensor networks must satisfy. It has to support event specification, real-time data fusion, and real-time stream data management. Since sensor devices typically have limited resources, this service should be light-weight. It must also provide high confidence event detection and minimize false alarms. All these features make building robust event services for embedded sensor network systems very challenging.

The main contribution of our work is that it provides key building blocks for robust real-time data and event services to be used by event detection applications. We have designed novel approaches that allow light-weight data fusion for real-time event detection and do not require significant computing resources. We are developing an event detection framework built around a collection of event specification, transformation, real-time monitoring, and fusion mechanisms. We expect that this framework together with the underlying mechanisms will significantly improve the accuracy and timeliness of event detection, as well as help reduce the resource requirements of embedded sensor network applications.

2. Complex event detection

2.1 Event description

Petri nets are well accepted as a model to describe systems with distributed, concurrent, asynchronous, and

*This work was supported by KOSEF WCU Project R33-2009-000-10110-0

non-deterministic features [1]. A basic Petri net consists of places (circles), transitions (rectangles or bars), directed arcs, and tokens (dots inside places). Arcs represent changes between states and the way in which tokens are created or destroyed. Places represent the states in which the application can be, and transitions are used to model various kinds of actions. Each place can contain zero or more number of tokens. A transition of a Petri net can fire whenever there is a suitable token in each of the input places for that transition. When the transition fires, it consumes these tokens, and injects tokens in its output places. A marking of a Petri net represents the status of the Petri net, i.e. a specific distribution of the tokens.

One of our objectives is to develop an effective specification language for event detection based on Petri nets. This language should address features specific to applications relying on the analysis of data from streaming networks (sensors, video cameras, etc). The foundation of our work is a coMpact Event Description and Analysis Language (MEDAL) [2]. As a formal method, MEDAL is based on Petri nets, which allows it to rigorously and unambiguously specify complex events. MEDAL attempts to address key aspects of event detection networks such as temporal control, spatial constraints, heterogeneity, and probability issues.

The MEDAL description of a sensor network event system can be given as a 7-tuple structure: F = (P, T, A, λ, β, H, L), where P is the set of places, T is the set of transitions, A is the set of arcs, λ is a probability/weight function for the arcs, β is a temporal guard function, H is a threshold function for places, and L is a spatial guard function for transitions.

Figure 1 shows the MEDAL model of a complex event detection application. The token at *Temperature event 1* represents the detection of temperature value *v* at time *t* by a sensor at location *(x, y)* with sensing range *r*. The event detected by this application, event *E*, is characterized by specific simple events in temperature (e.g., detection of high temperature at a specific location), pressure (e.g., detection of potentially dangerous levels of pressure), and friction (e.g., occurrence of high friction at the joint point). The occurrence of each of these simple events is represented by a token in the corresponding places. When all three tokens are present, transition *T4* fires and the application reports the detection of the complex event *E*. Complex sensor network events often require processing of stream data. For those events, tokens can be generated using the query results from stream data processing, as discussed in Section 3.

Complex events are a function of when and where they occur. We address the need of event detection applications for spatial and temporal semantics by incorporating spatial and temporal logic into MEDAL.

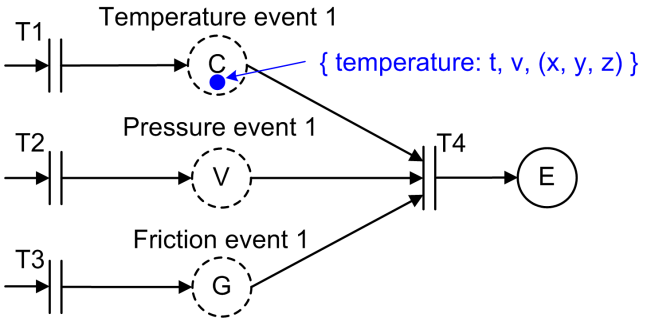

Figure 1: MEDAL model of an event detection system

2.1.1 Temporal logic: Temporal logic refers to the temporal guard function β. It helps specify the temporal concepts "when" and "how long" in MEDAL Petri nets. β guards the transitions to ensure they fire only during the specified temporal intervals. Introducing β in MEDAL has practical importance because some events can occur only during a particular temporal interval. For example, clustering to form a group can only happen if intensive communication occurs within a relatively short interval. In addition, β can help specify conditions such as "a transition will fire only if the input tokens have been generated within a predefined time interval". In Figure 1, for example, if the generation times of the tokens entering transition *T4* are more than 30 seconds apart from each other, it is more likely that the events were unrelated or they belong to different groups rather than indicating the occurrence of event E. In cases like this, the network should not report the occurrence of an event even if the necessary input tokens are present.

2.1.2 Spatial logic The geographic semantics of the application are enforced by the spatial function *L*. As a guard function for a transition *T*, *L* ensures that the tokens carried by *T*'s incoming arcs satisfy the spatial locality conditions. If *L(T) = R*, the effective radius of the higher-level event recognized by *T* should be equal to or smaller than R. In other words, there should be a circle of radius *R* encompassing all tokens' locations in order to consider the readings to be caused by one particular event. In Figure 1, for example, if the tokens in the input places of transition *T4* are generated at a distance larger that R from each other, it is likely that the events are not related. *L* could help detect such situations and thus decrease the number of false positives in event detection.

2.2 Introducing probabilities

Most previous work on event description in embedded sensor networks uses precise, also called "crisp", values to specify the parameters that characterize an event. For example, we might want to know if the temperature drops below 5°C. However, sensor readings are not always precise. In addition, different sensors, even if located close

to each other, often vary in the values they register. Consider an example scenario where we want the cooling to be turned on if the temperature goes above 5°C. Two sensors, *A* and *B*, measure the temperature in the room and the average of the values they report is used to determine if an action should be taken. At some point, sensor *A* reports 5.1°C and sensor *B* reports 4.8°C. The average, 4.95°C, is below the predefined threshold and the cooling remains off. However, if sensor *B*'s measurement is imprecise and therefore lower than the actual temperature, we have made the wrong decision. The situation becomes even more convoluted when more than two sensor measurements are involved. This makes determining the precise event thresholds an extremely hard task which has led us to believe that using crisp values might not be the best approach. Fuzzy logic, on the other hand, has a number of properties that make it suitable for describing events in sensor networks:

- It can tolerate the unreliable and imprecise sensor readings;
- It is much closer to our way of thinking than crisp logic. For example, we think of fire as an event described by high temperature and the presence smoke rather than an event characterized by temperature above 55°C and smoke obscuration level above 15%;
- Compared to other classification algorithms based on probability theory, fuzzy logic is much more intuitive and easier to use.

The structure of a general fuzzy logic system (FLS) is shown in Figure 2. First, the fuzzifier converts the crisp input variables $x \in X$, where X is the set of possible input variables, to fuzzy *linguistic variables* by applying the corresponding membership functions. Linguistic variables are "variables whose values are not numbers but words or sentences in a natural or artificial language" [3]. An input variable can be associated with one or more fuzzy sets depending on the calculated membership degrees. For example, a temperature value can be classified as both Cold and Lukewarm. Second, the fuzzified values are processed by *if-then* linguistic statements, called rules, derived from domain knowledge provided by experts. These rules are of the form:

IF premise, THEN consequent

where the *premise* is composed of fuzzy input variables connected by logical functions (e.g. AND, OR, NOT) and the *consequent* is a fuzzy output variable. The inference scheme maps input fuzzy sets to output fuzzy sets. Finally, the defuzzifier uses the output fuzzy sets to compute a crisp output. The crisp output value determines the control actions that need to be taken.

As previously mentioned, sensor readings are generally believed to be unreliable and imprecise. Therefore, to increase our confidence in the presence of an event somewhere in the monitored area, we often need readings from multiple sensors and/or readings over some period of time. To address this, we instrument the event detection process with spatial and temporal semantics. We believe that including temporal and spatial linguistic variables in the rule-base can significantly improve the detection accuracy. It can also allow us to describe and detect more complex events. To the best of our knowledge, no previous work on applying fuzzy logic to event detection has considered the effects of temporal and spatial semantics on the accuracy of event detection.

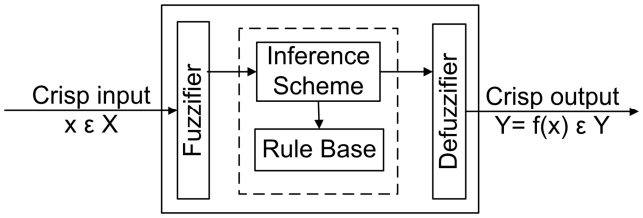

Figure 2: The structure of a fuzzy logic system

2.2.1 Spatial semantics: It is important to understand how including spatial guards affects the accuracy of event detection. Spatial guard variables will allow us to express any spatial requirements the detection application might have. An example requirement could be: "if the readings of two sensor nodes indicate the presence of event X, we believe that this event has occurred only if the two sensor nodes are located close to each other". A disadvantage of including spatial semantics is that sometimes events might not be detected if the spatial guard is too strict. However, we believe that using fuzzy logic will help us alleviate this problem.

2.2.2 Temporal semantics: To further decrease the number of false alarms we also need to take into account the temporal properties of the monitored events. One approach is to include linguistic variables in the rule-base that can act as temporal guards. Adding temporal semantics is especially important for embedded sensor networks because of the nature of sensor communication. It is very possible for messages in a wireless sensor network to be delayed because of network congestions or bad routing. Consequently, a reliable event detection rule-base should take into consideration the generation times of the participating sensor readings.

2.2.3 Decreasing the rule-base: A disadvantage of using fuzzy logic is that storing the rule-base might require a significant amount of memory. The number of rules grows exponentially to the number of variables. With *n* variables each of which can take *m* values, the number of rules in the rule-base is m^n. For example, if there are four linguistic variables each of which can be associated with one of five values, the rule-base will contain 625 rules.

Adding spatial and temporal linguistic variables to the rule-base further increases the number of rules. Since sensor nodes have limited memory, storing a full rule-base on every node would be a waste of valuable resources. In addition, constantly traversing a large rule-base might considerably slow down the detection process. Further, the intensified computation caused by frequently going through a larger rule-base will also lead to increase in the power consumption. To address this issue, we have designed a set of rule-base reduction techniques to decrease the size of the rule-base. An important property of these techniques is that they do not negatively affect the accuracy and promptness of event detection.

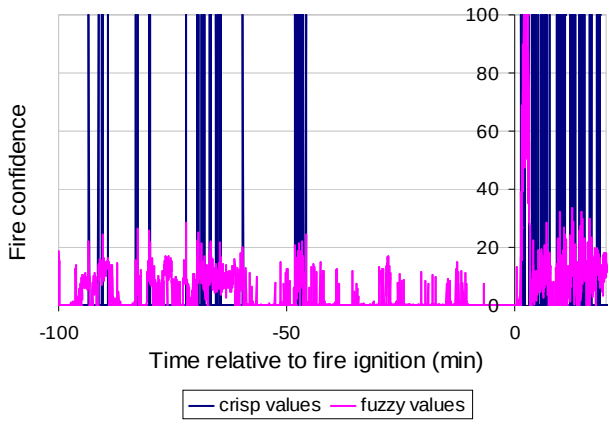

Figure 3: Fire simulation: burning mattress

2.2.4. Experimental results: We have performed preliminary simulation experiments on real fire data publicly available on the National Institute of Standards and Technology (NIST) website [4]. The goal of these simulations was to evaluate how using fuzzy logic affects the accuracy of event detection. The experiments were run with both crisp and fuzzy values. The temperature and smoke obscuration thresholds we used in the crisp logic experiments are threshold values used in commercial smoke and heat detectors [5, 6].

Figure 3 shows the results from one of the crisp-value experiments. The origin of the coordinate system represents the time of fire ignition. As we can see from the figure, using crisp values results in a very large number of false fire detections. In the period prior to fire ignition, there were 40 false fire detections which constitutes about 1.3% of the readings. Such a considerable number or false positives significantly affects the efficiency and fidelity of an event detection system. Our results show that fuzzy logic is more suitable for event detection in sensor networks. However, we need to perform additional experiments to determine if there are applications for which this does not hold true. It is also important to compare the resource requirements of applications using fuzzy and crisp values.

We used the same scenario to evaluate the efficiency of the rule-base reduction techniques. The rule-base initially had 81 rules. Applying two of our reduction techniques, which take advantage of the similarity between rules as well as the significance of different rules, helped decrease the size of the rule-base by more than 70% without compromising the event detection accuracy.

2.3 Event transformation

Transforming the formal model into code that can be executed on the sensor nodes is the next step an event service needs to perform. Since the process of recognizing the specified events is similar to a DNA transcript procedure, we call the event recognition code generated from the MEDAL model and stored on the sensor nodes, event-DNA. Each event-DNA is an encoded representation of a MEDAL model and just like a MEDAL model, the event-DNA might represent the description of simple or complex events. The sensor nodes have an event detection middleware stored in their memory which can read different event-DNAs and act accordingly.

2.4 Event service framework

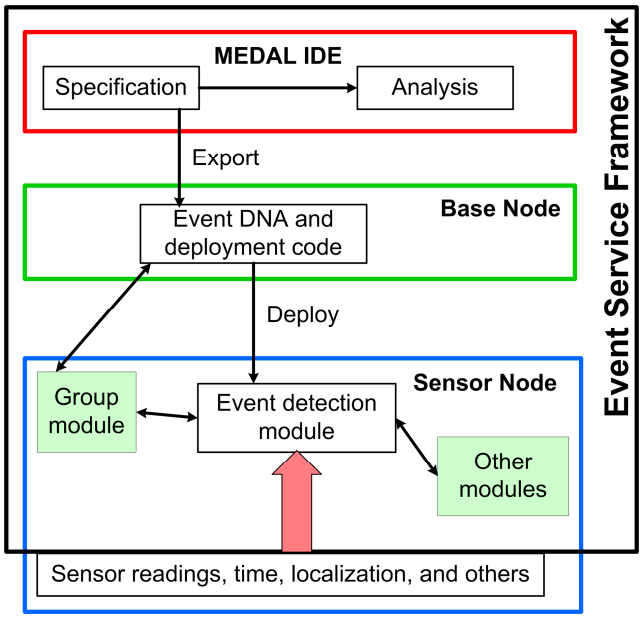

Figure 4: Event service framework

The overall event service architecture we are developing is shown in Figure 4. In the figure, the MEDAL IDE (Integrated Development Environment) is an offline package which resides on a PC and from which specified event semantics can be encoded and exported to a base node. The base node installs a deployment module that deploys all event-DNAs onto their corresponding motes. For example, if an event-DNA represents a mote-

level event, it will be deployed onto every mote, while if an event-DNA represents a group-level event, only a group leader will have a copy. For dynamic leader schemes, the deployment module needs to interact with the group module to determine what to deploy. Then inside each mote, the imported code generated from the event-DNA is stored in program memory. The detection module achieves its goal by calling other lower primitives in the program section such as reading sensor values, obtaining time, location information, and other data from other modules in the program section.

The framework contains the following modules:

- **MEDAL environment**, which includes both a specification module and an analysis module. It also encodes event specifications into event-DNAs and exports them to the base node.
- **The Base node** contains encoded event-DNAs and has a deployment module in charge of transferring event-DNAs to nodes in the field. The deployment module provides communication, in which the messages are event-DNAs.
- At the **Sensor node** level, the event detection module resides in program memory. The event detection module uses a token vector to communicate with other nodes for collaborative detection of higher level events.

Once the designers have the MEDAL model that describes their application, the next step is to write the code to be run on the nodes. A weakness of this step, however, is that manually translating the formal model into code might introduce bugs as well as lead to divergence of the code from the model. An advantage MEDAL has over other currently used event description approaches is that due to its formal structure and lack of ambiguity, it can be directly and automatically translated into code that can be executed on the sensor nodes. We have been developing a tool to automatically generate the event-DNA code based on the MEDAL model of the application. Currently, we are generating TinyOS code using the formal application model [7]. However, our approach is adaptable so that code in languages other than nesC can be generated as well. This approach will significantly reduce the effort of writing TinyOS code and improve the correctness of the code.

3. Real-time stream data

3.1 Overview

For timely detection of complex events, real-time embedded sensor network systems have to operate on continuous unbounded data streams from multiple sources. The streaming data may come from different types of sensors which need to be integrated and fused with each other as well as with the data already stored in the system. Stream data takes the form of continuous, ordered, potentially infinite data streams, as opposed to finite, statically stored data sets. In embedded sensor network applications, there exist significant volumes of dynamically changing data in the form of data streams, such as data associated with the current locations and movements, data in the form of simple events detected by other subsystems, and data from other sources reflecting the dynamic and volatile situations. Analysis of stream data poses great challenges to real-time data management, due to the unique features of data streams, such as huge (and possibly infinite) volume, unpredictable changing patterns, and flowing in-and-out in a dynamic order. Due to the high volume of data streams and the timing constraints of the applications, it is often assumed that it is not possible to store a stream in its entirety, nor is it feasible to query the whole stream history. Typically, the queries are executed on a *window* of data. A window on a data stream is a segment of the data stream that is considered for the current query. A lot of stream data resides at the primitive abstraction level in the form of raw sensor data. It is necessary to perform *aggregation and generation of derived data* from the raw data to find interesting patterns or outliers at appropriate levels of abstraction and with appropriate dimension combinations.

3.2 Quality management

To process real-time stream data in embedded sensor networks, the system should support long-running and persistent queries. When a query arrives in the system, it is registered and its instances are executed periodically. All queries are pre-registered in the system and converted to query plans (containing operators, queues, and synopses) before the system starts executing. Queues in a query plan model the incoming data streams and the intermediate results between the operators. A synopsis could be related to a specific operator and it stores state that may be needed for future evaluation of the system. For example, a join operator may be associated with a synopsis for each of its inputs to store some of the items. These items can be probed later by the operator as needed.

Let us consider the following data streams and query associated with it. The query result can be used to detect unusually speeding trucks.

Stream: Speed (int lane, float value, char[8] type);
Relation: Lanes (int ID);
Query: SELECT avg (Speed.value) FROM Speed [range 1 minute], Lanes WHERE Speed.lane = Lanes.ID AND Speed.type = Truck;
Period 10 seconds
Deadline 5 seconds

The query above operates on data streams generated by speed sensors and calculates the average speed of trucks in

particular lanes in the last 1 minute. The query needs to be executed every 10 seconds and the deadline is 5 seconds after the release time of every periodic query instance. The generated query plan is shown in Figure 5. This query plan consists of three query operators (range window operator, join operator, and aggregate operator) and two buffers (one for storing the range window output and the other for storing the output of the join operator).

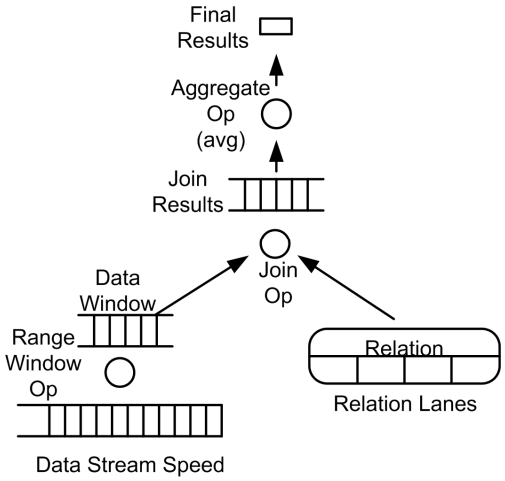

Figure 5: Query plan

One of the main challenges for data stream management with real-time constraints, as in embedded sensor networks, is the unpredictability of the data streams themselves. It is possible that the system may get overloaded as the arrival rates and contents of the incoming data streams change. The system must be able to handle these workload fluctuations. Otherwise some of the queries may miss their deadline. This QoS management can be performed at two levels. The *inter-query QoS management* can allocate resources to different queries in the case of system overload so that they have similar quality in query results. The *intra-query QoS management* is then used to allocate available resources to different operators within the query plan so as to maximize the query quality. For inter-query QoS management, the system needs to estimate the query execution time corresponding to different input data sizes. For intra-query QoS management, the system needs to know the selectivity of different operators corresponding to the current data input and their estimated execution time. Since our main objective is to ensure the timeliness of query results, the query execution time estimation is the key to QoS management. In order to estimate the query execution time, we need three parameters for each query, namely, the input data stream volume, the operator selectivity, and the execution time per data tuple for each operator. In this work, we consider queries that are ready to be executed when performing the QoS management routine. Therefore, the input data volumes for these queries are known since the incoming data stream segments are already present in the system.

It is beneficial to estimate the execution time of queries which do not have their complete input yet and use these estimations in the QoS management process, since the current ready-to-go queries may overlap with these future queries in their life span. This is a challenging problem as it involves designing effective algorithms to monitor and estimate the volume and contents of data streams.

4. Summary

This work provides the basis for developing robust real-time data and event services that allow sensor network applications to monitor, detect, and react to sophisticated events. We are designing an event service framework which combines several techniques that will help provide light-weight and timely event detection in embedded sensor networks. In addition, in order to improve the performance of these sensor systems, we are developing approaches to manage the QoS. We believe that combining the event service framework and the QoS management mechanisms will significantly improve the timeliness and reduce the resource requirements of event detection in embedded sensor network systems.

5. References

[1] C. Girault, R. Valk. “Petri Nets for System Engineering: A Guide to Modeling, Verification, and Applications” Springer-Verlag New York, Inc. (2001).

[2] K. Kapitanova, S. H. Son. “MEDAL: A coMpact Event Description and Analysis Language for Wireless Sensor Networks”, *International Conference on Networked Sensing Systems*, 2009

[3] L. Zadeh. “Outline of a New Approach to the Analysis of Complex Systems and Decision Processes”, *IEEE Transactions on Systems, Man, and Cybernetics*, 1973

[4] Building and fire research laboratory. http://smokealarm.nist.gov/.

[5] J. Geiman and D. Gottuk, “Alarm Thresholds for Smoke Detector Modeling,” *Fire Safety Science – Proceedings of the 7th International Symposium* 2002

[6] WS4916 Series Wireless Smoke Detector. http://www.alarmsuperstore.com/dsc/ws4916installation.pdf.

[7] Y. Wu, K. Kapitanova, J. Li, J Stankovic, S. Son, K. Whitehouse, “Run Time Assurance of Application-Level Requirements in Wireless Sensor Networks”, *IPSN 2010*

[8] M. Keally, G. Zhou, G. Xing, “Watchdog: Confident Event Detection in Heterogeneous Sensor Networks”, *RTAS 2010*

Elements of Scalable Data Processing

Björn Andersson*, Paulo Gandra de Sousa*, Filipe Pacheco*, Panayiotis Andreou†,
Pedro José Marrón‡, Umer Iqbal‡ and Vinny Reynolds§
*CISTER Research unit, Polytechnic Institute of Porto, bandersson@dei.isep.ipp.pt
*CISTER Research unit, Polytechnic Institute of Porto, pag@isep.ipp.pt
*CISTER Research unit, Polytechnic Institute of Porto, ffp@isep.ipp.pt
†University of Cyprus, panic@cs.ucy.ac.cy
‡University of Duisburg-Essen, pjmarron@uni-due.de
‡University of Duisburg-Essen, umer.iqbal@uni-due.de
§The National University of Ireland, Galway, vinny.reynolds@deri.org

***Abstract*—Cooperating objects (COs) is a recently coined term used to signify the convergence of classical embedded computer systems, wireless sensor networks and robotics and control. We present essential elements of a reference architecture for scalable data processing for the CO paradigm.**

I. Introduction

As embedded computer systems increase in importance and become networked, new concepts emerge to emphasize different aspects of them. The concepts of pervasive and ubiquitous computing emphasize seamless deployment in everyday things and environments. The concept of wireless sensor networks (WSNs) emphasizes sensing and networking aspects, whilst the concept of cyber-physical systems (CPSs) emphasizes the interaction between information processing ("Cyber") and the physical world. The concept of cooperating objects (COs) [1] has recently gained acceptance in signifying the convergence of classical embedded computer systems, wireless sensor networks and robotics and control, whilst emphasizing ad-hoc, opportunistic cooperation between (often) autonomous entities. Furthermore, it is currently seen by the European Commission as the main research direction in networked embedded computer systems.

Formally speaking, a cooperating object (CO) [2] is a single entity or a collection of entities consisting of (i) sensors, (ii) actuators, (iii) information processors or (iv) cooperating objects. Since the definition of a CO is recursive (a CO may be a collection of COs), this definition makes it possible to form arbitrarily complex structures. The concept of a CO is quite new however and therefore, no general theory for COs is known and no model or architecture for data processing concerning COs is known either.

It is possible to co-locate a single sensor, a single information processor and a single actuator to a single computer node and let the information processor perform processing based on readings from the single sensor and let the actuator enact commands computed by the single information processor. Harnessing the power of cooperating objects requires however that information processors perform processing of sensor readings originating from different computer nodes. Sometimes this is strictly necessary. For example, most methods for estimating the most likely geographical location of a physical object depend on sensor readings originating at different computer nodes. In other cases, an information processor can obtain a much better image of the physical world, using better resolution and greater coverage, if the information processor takes as input sensor readings originating from different computer nodes. Clearly, many COs need to perform data processing based on sensor readings originating from different computer nodes.

Since COs often interact with their physical environment by giving commands to actuators (these commands are computed based on sensor readings), it is essential that the *response time* of data processing from sensing to actuation is small. A straightforward approach to perform data processing in COs is that each actuator has an associated information processor and all sensors that provide sensor readings that are needed for the data processing on this information processor send the sensor readings to this information processor. This approach is applicable in COs composed of a relatively small number of sensors. But the trend [3], [4], [5] in COs is towards a larger number of sensors because (as already stated) this provides a better image of the physical world. Unfortunately, performing data processing with the straightforward approach leads to long response times because of the large amounts of packets that must be sent from each sensor. This is particularly problematic when all sensor nodes are in a single broadcast domain because then at most one sensor can transmit at a time. Clearly, better methods for data processing are needed; data processing based on sensor readings originating from different computer nodes should be performed in a scalable manner; that is, the response time should grow slowly (or not at all) with the number of sensor nodes/readings.

Many COs are comprised of sensor nodes belonging to different organizations with no superior/subordinate relationship between these organizations. Yet, the functioning of the CO requires that data processing is performed based on these sensor readings. And this brings an additional challenge to data processing in COs. Therefore, a core problem in the design of COs is to perform data processing based on sensor readings originating from different computer nodes potentially in different organizations and do so in a scalable manner; we refer to this as the *Scalable Data Processing* problem.

In this paper, we present essential elements of a reference architecture for the Scalable Data Processing problem[1]. We believe that developing this reference architecture is significant because (i) COs with a large number of sensor nodes and which operates across a large number of organizations will be built in the future and (ii) there is (as far as we know) no reference architecture available for the Scalable Data Processing problem. Since COs and CPSs have many commonalities, we also expect our future work on this reference architecture to be useful also for CPSs.

The remainder of this document is organized as follows. Section II presents a usage scenario to motivate the need for query processing. Section III presents a system overview. Section IV presents the architecture. Section V presents related work. Finally, Section VI presents ongoing work.

II. A USAGE SCENARIO

In order to understand the scalable data processing problem, let us consider a usage scenario.

Europe food safety legislation [6] specifies that when certain foods are being transported, they must be kept within certain temperature thresholds for the safe preservation of the food's quality. For example, it is necessary to be able to verify that frozen foods must be kept frozen for the entire duration of the journey. From a European health and safety perspective, food that defrosts and is re-frozen is considered unsafe and this event must be detected and reported. In order to accomplish this, many refrigerated vehicles, "reefers", now embed wireless sensor networks in vehicles and in load containers. Coupled with on board telematics units, this set up allows the monitoring of these temperature thresholds during the entire food transportation cycle.

A. Description

Typically, a fleet management company provides clients (food companies in this scenario) with a fleet administration tool which allows clients to make a range of queries against vehicles transporting their load. Each vehicle, shown in Figure 1, may be connected to the fleet administration tool via a 3G internet connection, and each vehicle administers its own wireless sensor network, which is monitoring the vehicle's load. For larger companies, they may have several hundreds or even thousands (in the case of supermarket chains) WSN-enabled vehicles transporting loads across Europe at any one time. A simple geospatial query requesting the location of all vehicles at a single point in time, would result in a very large and costly query being disseminated to potentially all of these vehicles. In this scenario, minimizing the financial cost of these queries is very important to the fleet administrators, and doing this in the presence of large scale systems is a challenge.

[1]Due to space limitations, we do not present the entire architecture — only the essential elements of it.

Fig. 1. A typical query in the fleet management scenario

B. Scenario Characteristics

This scenario has several characteristics that are noteworthy with the scalable data processing problem. These include:

- Dynamic scale. The total number of sensors (gateways + wireless sensors) within the entire query system includes the amount of wireless sensor nodes on each individual vehicle in addition to the mobile gateway on each vehicle. Dynamic aspects of typical queries such as geospatial queries, can result in a very large number of nodes having to be queried.
- Mobility. Gateway nodes (on the vehicle) are mobile as are the wireless sensor networks themselves, but not with respect to the gateway node. The gateway node is attached to a mobile telematics unit, (equivalent to a embedded mobile phone), which can both process one shot, continuous and event triggered queries. The mobile telematics unit is equipped with a GPS, so accurate location data is usually provided.
- Expensive communication costs. The cost of an individual communication between the client and a vehicle is small but not zero, as the communication is performed over a 3G internet connection. For example, continuous queries that push data constantly upwards are cost prohibitive and may only be executed once an hour, or even less frequently. Effective management of queries should help to depress the overall cost of the communication within the system, which is an important characteristic of this scenario.

C. Queries

A client sends his query from a non-self administered website, a portal. Two typical queries would be as follows:

1) Query Q is "Display the location of all of my vehicles". Without any prior information, an uninformed query planner would have to disseminate this one shot query to all telematics devices. They would respond with their location.
2) Query Q2 is a continuous query. "Raise an event when the temperature within a vehicle is greater than -2 degrees." In this query, periodic single shot queries are ex-

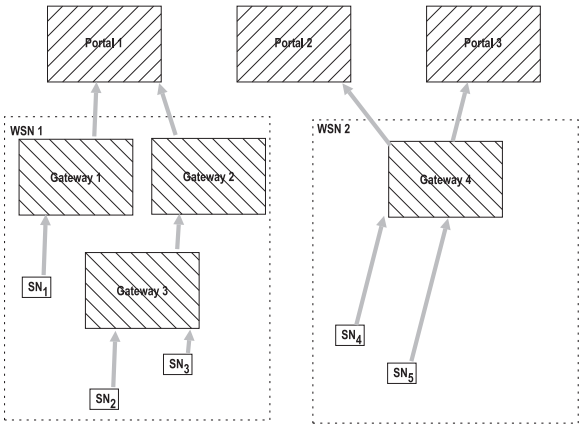

Fig. 2. An example of a CO that performs scalable data processing. There are three users (not shown). One user asks a query to Portal 1; another user asks a query to Portal 2 and a third user asks a query to Portal 3. The former query requires execution of sensor reading originating in WSN_1. The two latter queries require execution based on sensor readings in the same WSN, WSN_2. The gray arrows show the flow of data when executing the queries.

pensive and inefficient. Events that are triggered locally by the vehicle's onboard sensors push the data/event towards interested parties (the client).

III. System overview

In this section we will formalize our system model and the basic terminology which will be used in the subsequent sections.

Let $\{SN_1, SN_2, \ldots, SN_n\}$ denote a set of individual sensing devices with the ability to communicate in an adhoc manner wirelessly. These sensing devices have the ability to acquire physical attributes at discrete time instances t and then propagate them using a multihop communication mechanism to some gateway $Gateway_i$. A gateway has the purpose of retrieving critical data from individual sensor nodes and has a larger energy capacity. This setup is typically referred to as a Wireless Sensor Network (WSN) which is normally owned by a single organization. Usually, a WSN consists of a single gateway but there are often cases where more than one gateway exists to better support the operation of the WSN. For example, in Figure 2, WSN_1 consists of three gateways.

A user/application can specify a query Q directly to a WSN (i.e., through a gateway) or through a portal. In the former case, the user must first identify the proper gateway and then follow specific access routines and protocols to issue Q. On the other hand, portals offer a transparent way of accessing a WSN (or multiple WSNs) by providing a set of abstract interfaces that allow the user to query one or more WSNs easily in an identical manner. To accomplish this, portals employ: (i) WSN registries that allow applications to discover WSNs, and (ii) data transformation mechanisms that enable communication to (i.e., query) and from (i.e., data) the WSN.

In this work, queries are represented in a declarative SQL-like syntax. For instance, the following query declares that each sensing device should recursively collect the node identifier and the temperature from its children every 31 seconds and communicate the results to the gateway. `SELECT nodeid, temp FROM sensors EPOCH DURATION 31 seconds`

IV. Architecture

The reference architecture describes the main components that have to be provided to meet the requirements for scalable data processing across heterogeneous sensor networks. As an *in-node architecture*, the reference architecture omits the description of specific interfaces and implementations, but instead presents generic roles that should be met by software components on each node, regardless of whether the node is a sensor node, gateway or portal. Within these nodes, different software and hardware constraints are enforced requiring the reference architecture to be implemented to meet these constraints. Our scalable data processing architecture, shown in Figure 3, presents six main components. Due to the brevity of the paper, only those components that relate to the key elements of scalable data processing are described, and as such the descriptions of the API layer, and the system management component are omitted.

A. Query Manager

The Query manager is responsible for the execution of queries in our framework. Queries are posted to the query manager via the Query API. There exists three different groups of queries in our framework categorized according to the layer at which they are going to be executed. These are the (i) Portal, (ii) Gateway and (iii) Broadcast domain levels. Our framework will support different instantiations of the same components in order to cope with this diversity.

The Query Manager consists of the following components:

- **Parser:** Queries executed within a wireless sensor network can be classified using a variety of metrics, e.g. number of returned results, aggregation mechanisms used, execution frequency, etc. The Parser component is responsible for translating queries from a predefined format to a data structure. In order to do so, the Parser incorporates a grammar that checks the query syntax and looks for specific tokens that exist in the grammar. This process enables the parser to determine the query semantics and build the appropriate data structure that will be used by other components for the execution of the query. Our framework supports three query types: (i) Selection Queries, (ii) Storage/View Queries and (iii) Event based Queries.
- **Planner:** The Planner component handles the planning of the queries. Specifically, it considers all possible query plans for a query Q, calculates a "cost" for each query plan and then opts for the one with the lower cost. Typically, "cost" is calculated with regards to response time performance (I/O operations, memory usage, etc.). However, in WSNs, "cost" is evaluated with different metrics that take into account the peculiarities and limitations of WSNs (e.g., limited battery).
- **Query Optimizer:** The Query Optimizer sub-component considers all possible query plans and chooses the

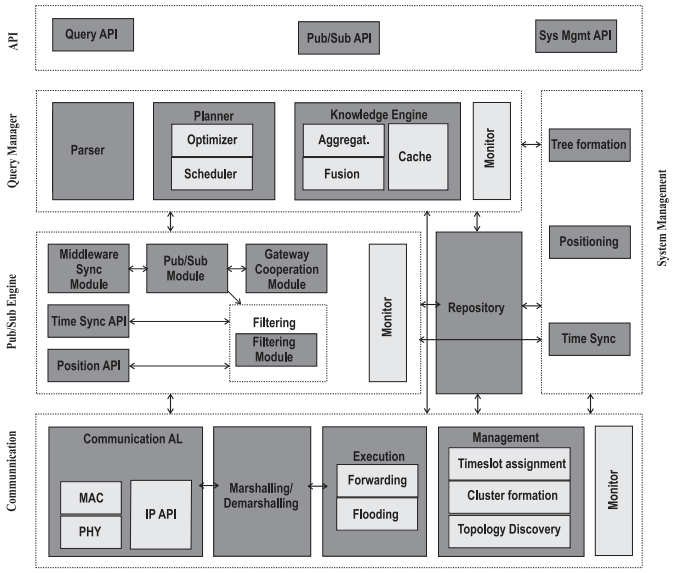

Fig. 3. The Scalable Data Processing reference architecture

most cost-efficient, whilst the scheduling sub-component schedules the execution of the query depending on the query type.

- **Knowledge Engine:** The knowledge engine is responsible for advanced topics in the query execution process like aggregation. Additionally, it utilizes a caching component that enables query instantiations to cache results locally in order to speed-up operations.

B. Publish/Subscribe

Unlike in traditional networks where the communicating entities are interested in the actual sender and receiver of data, data centric networks are less worried about this information and more concerned with the data transmitted between those entities. This means that in such networks the entities register for a particular data type in which they are interested in and not with particular nodes offering or looking for data. This is also true for the majority of wireless sensor network applications where the applications are interested in the data gathered by the sensor nodes and not in the physical sensor nodes themselves.

The Publish-Subscribe (Pub/Sub) paradigm is one such approach in which entities associate or register themselves to a type of data. It is an asynchronous messaging paradigm which provides a decoupling between the sender and receivers of data and provides one of the better abstractions for communication in mobile networks or applications. The Pub/Sub paradigm is based on the roles of publisher and subscriber, either or both of which can be assumed by a network entity. Entities which assumes the role of publishers publish data and entities subscribed to that type of data receive it. The pub/sub interaction can be either centralized or have some distributed mechanism. In the centralized approach, all publishers notify a central entity or a broker about the data which they would like to publish and all subscribers make their subscription to that central entity based on their requirements. The central entity on having a condition where a subscription request can be fulfilled by a publisher, forwards the data to the subscriber. In a distributed approach, every node can assume a role of a publisher or subscriber or both. In this case, the publish and subscription requests are sent over the network and are maintained by every node, which it can use knowledge of what types of data are available in the network and whether it is interested in it. Conversely, the type of data published locally is also known and a list of subscribers to that data within the network is also maintained.

Usually, publishing and subscription requests are additionally filtered on topics and/or content. In the former, the association is made on the topic e.g. if there is a temperature data available for the building then pass this information to the subscriber, whereas in the latter the approach, association is based on the contents i.e. if the temperature in the building is more than 25° C then pass this information to the subscriber.

In the proposed reference architecture, publish subscribe mechanism can be used for both or either of the actual sensor data and metadata regarding the sensor network. Examples of actual sensor data include temperature, humidity, light etc. whereas metadata can include network performance parameters such as number of subscriptions, position of sensor nodes, number of active nodes etc. The choice of using publish subscribe mechanism for a particular type of data depends upon the usage scenario. Depending upon usage scenario, it is also possible to use publish subscribe mechanism for regular sensor data at one part of the network and for transferring metadata for some other part of same network. An example could be a usage scenario mentioned in Section II, where publish subscribe mechanism can be used for transferring metadata information between different containers and also for transferring regular sensor data within a network in a single container.

C. Communication

In order to achieve scalable data processing, it is clearly necessary for nodes to communicate. The need for communication is however a major source of energy consumption (because transceivers are left on) and delay (because of contention for the medium) and therefore it is crucial that the architecture makes efficient use of the communication system.

Our use of the communication system is quite different from that of many typical address centric networks. For example, the pub/sub component receives a packet and its filtering module may inspect it and decide whether the packet should be forwarded. Execution of queries typically requires that in-network processing is used; typically many packets are received from child-nodes and a computation based on their data payload is performed and the result is transmitted. And query processing in a single broadcast domain may need the communication system to simply perform contention for the medium. For example MAX and MIN of sensor readings in a single broadcast domain can be computed efficiently using a prioritized medium access control protocol. Because of these reasons, the communication component must expose details rather than hiding them.

Our architecture follows these principles in the following way. The communication abstraction layer exposes primitives for other parts of the architecture to use in innovative ways in order to achieve scalable data processing. And the component "execution" provides address-centric services.

We note that the components Timeslot assignment and cluster formation configure the communication system. These can be used to optimize the execution of an ongoing continuous query, for example, if node N_1 and node N_2 transmits packets to node N_3 and N_3 aggregates this information and forwards this aggregate to N_4 then it is desirable (from the perspective of attaining low latency) that the transmissions from N_1 and N_2 has a lower timeslot id than the transmission from N_3. The timeslot assignment can also consolidate timeslots so that one node has a long wakeup time instead of two short wakeup times in a TDMA cycle. The functionality of these components can also be used by the query optimizer to setup the timeslot assignment and cluster formation when a new query (especially for a continuous query) is setup.

D. Repositories

The Cross-Layer Repository (XLR) is the location for the storage of meta data generated by local instantiations of the reference architecture and also meta data generated by remote instantiations. The purpose of maintaining this meta data is that it may be used by several other components within the architecture in order to achieve optimizations and scalability that would otherwise not be possible. For example, knowledge of the physical location of leaf instantiations of the reference architecture may be used to optimally direct certain spatial queries to some leaf nodes, and not others. It cannot be assumed that this information is known a-priori or that this information is even static. For example, node mobility may influence the outcome of certain types of queries, and being able to maintain certain types of meta-data at parent instantiations of the reference architecture may make better optimizations possible.

Query based meta-data In order to be able to plan and execute queries dynamically, it is necessary to know and store what types of queries can be executed at other instantiations of the reference architecture. To achieve effective query planning, it is also required that some additional information such as the cost of the query, be maintained, and that cost information may be dependent on how the query is implemented at that node. For example, a certain query might have one cost when the overall query is executed in such a way as to ensure a very timely response, but may have a different cost when the query is executed in such a way to minimize the amount of messages required to execute that query, i.e. optimize based on energy consumption. The cross layer repository is responsible for the consistent storage of meta-data that captures this information.

One proposed solution is that the XLR maintains a tuple set (N_i, Q_i, C_i, O_i) capturing this information, where N_i is the node where the query is to be executed at, Q_i is that query, C_i is the cost of that query given that it is optimized according to O_i.

Sensor node meta-data Additional information pertaining to the nodes themselves may also be maintained as meta-data. This would include information such as routing tables and also the physical topology of the network, for example the location of nodes.

V. Related Work

There is currently no reference architecture specifically focusing on scalable data processing. There are however related architectures.

RM-ODP [7] is a Reference Model for Open Distributed Processing. It shares our goal of addressing heterogeneity and interaction between objects in different organizations. But it does not address scalability in terms of data processing.

Ultra-Large Scale Systems (ULS) [8] is a study made by Software Engineering Institute at Carnegie-Mellon University for the US Department of Defense. The study discusses how to design software systems comprising more than one billion lines of code and where these software systems are highly distributed and deployed across different organizations. The report about ULS differs from our paper in that ULS provides a roadmap whereas we provide a reference architecture. Also, ULS focuses on interoperability and negotiation between different systems and suggests the use of mechanism design. These issues are not in the scope of our paper.

Scalable Querying of Sensor Networks from Mobile Platforms Using Tracking style Queries (SENSTRAC) proposed in [9] is an approach with some similarities to the our proposed architecture. SENSTRAC aims at querying sensors through mobile nodes (mobile phones, PDAs) using publish subscribe mechanisms. SENSTRAC assumes resource rich sensor nodes integrated in to the infrastructure. Due to the mobility aspects of applications, SENSTRAC instead of in-network aggregation of query results, suggests transformation of queries into the

subscriptions to the topics published by the sensor nodes to incorporate increasing number of querying nodes to meet the scalability requirements. This aggregation at the subscription level is proposed by a separate algorithm periodically computing sensors of interest. SENSTRAC uses leased based subscriptions in which subscription is valid for a limited time period and does not require unsubscription procedure mainly because of the mobility aspect. SENSTRAC has two types of mapping, one from query to the topics and one from topics to the sensor nodes. The sensor network architecture proposed by SENSTRAC is a broker based architecture in which subscriptions and publish offers are sent to the broker nodes in a static grid-cell network structure. In addition to intra-cell communications, the broker nodes are also used for inter-cell communications to avoid redundancy of sensed data.

Scalability, in terms of the number of nodes within a Cooperating Objects Network, has also been studied in the context of the IPAC project [10]. IPAC aims at delivering a middleware and a service creation environment for developing embedded, collaborative and context-aware services in mobile nodes equipped with sensing devices. IPAC relies on short range communications for the ad hoc realization of dialogs among collaborating nodes. The networking capabilities of IPAC are based on rumour spreading techniques, a stateless and resilient approach, and information dissemination among embedded nodes. One of the key advantages of this scalable ad-hoc network construction/communication mechanism is the ability to integrate new mobile nodes and new sensing elements in an efficient manner. However, in contrast to the reference architecture proposed in this paper, scalability with regards to multiple gateways has not been considered.

TAG [11] is a query processor for WSN. It allows the execution of restricted SQL queries and it uses in-network aggregation by creating an aggregation tree (called routing tree in [11]) to improve scalability. Our architecture differs from the one in TAG in that (i) our architecture may take advantage of query processing in a broadcast domain using the MAC to improve scalability, (ii) we consider queries that may span multiple WSNs and (iii) our architecture considers query optimization (for example by taking knowledge of the network topology into account).

VI. Ongoing work

The next steps of our ongoing efforts are now to focus on the development of a prototype implementation of the complete architecture, i.e. the query manager, publish subscribe engine, communication layer, system management layer, cross layer repositories, etc. As part of the requirements specification phase, different real world scenarios were identified and for the first prototype, we will implement one of them. The first challenge in this regard is the selection of an appropriate simulation platform which can provide the required functionalities for implementing the proposed architectural components. The simulation platforms that we have investigated so far do not completely fulfill our requirements. Therefore, additional features will be added to the existing platform(s) for developing the prototype. For example, the prototype implementation will have to be executable at the different levels of the network namely the portal, gateway and broadcast domain levels. This is not supported by any existing simulator. In parallel, we will be refining the architecture with more low level details on architectural components, interaction between the different layers and addressing open issues identified in the design phase such as data models for the cross layer repository and syntax/semantics rules of query language(s). In addition to the prototype, an evaluation methodology for the reference architecture will be devised to validate that the proposed architecture achieves its goals. The validation process will ensure the conformance of the specified requirements with the prototype.

Acknowledgements

This work was partially supported by CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053, ARTISTDesign Network of Excellence on Embedded Systems Design, funded by the European Commission under FP7 with contract number ICT-NoE-214373 and the Portuguese Science and Technology Foundation (Fundação para Ciência e Tecnologia - FCT) and SmartSkin project supported by ISEP and by the European Union under the project IPAC (#224395) and the Cyprus national project MELCO (#TIIE/OPIZO/0308/(BIE)/14) and by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

References

[1] P. Marrón, D. Minder, and The Embedded WiSeNts Consortium, "Embedded WiSeNts research roadmap," IST/FP6 (IST-004400), 2006.

[2] S. Karnouskos, "Research roadmap on cooperating objects, section 2.1: Definitions, available at http://www.cooperating-objects.eu/roadmap," 2009.

[3] A. Arora, R. Ramnath, E. Ertin, P. Sinha, S. Bapat, V. Naik, V. Kulathumani, H. Zhang, H. Cao, M. Sridharan, N. Seddon, C. Anderson, T. Herman, N. Trivedi, C. Zhang, R. Shah, S. Kulkarni, M. Aramugam, and L. Wang, "Exscal: Elements of an extreme scale wireless sensor network," in *RTCSA '05: Proc. of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2005, pp. 102–108.

[4] A. Rowe, M. Berges, G. Bhatia, E. Goldman, R. Rajkumar, L. Soibelman, J. Garrett, and J. M. F. Moura, "Sensor Andrew: Large-scale campus-wide sensing and actuation," Carnegie-Mellon University, Pittsburgh, PA, Tech. Rep., 2008, available at project website: http://www.ices.cmu.edu/censcir/resources/SensorAndrew-Tech-Report.pdf.

[5] "ARTEMIS EMMON project web page, http://www.artemis-emmon.eu/objectives.html," 2009.

[6] "Article I of Regulation (EC) No 852/2004 of 29 April 2004 on the hygiene of foodstuffs, http://www.fsai.ie/uploadedfiles/consol_reg852_2004.pdf."

[7] "RM-ODP, http://www.rm-odp.net/," 2010.

[8] P. Feiler, R. P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, L. Northrop, D. Schmidt, K. Sullivan, K. Wallnau, and B. Pollak, *Ultra-Large-Scale Systems: The Software Challenge of the Future*, 2006.

[9] S. Pleisch and K. P. Birman, "SENSTRAC: scalable querying of SENSor networks from mobile platforms using TRACking-style queries," *Int. J. Sen. Netw.*, vol. 3, no. 4, pp. 266–280, 2008.

[10] "IPAC integrate platform for autonomic computing, http://ipac.di.uoa.gr/," 2009.

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation service for ad-hoc sensor networks," in *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*. New York, NY, USA: ACM, 2002, pp. 131–146.

The Reality of Wireless Real-Time:
Wireless Networks in Industrial Automation

BSc Sander Rotmensen
Siemens AG, Industrial Automation

Speaking about wireless technologies it is important to first have a look on the different categories of available technologies in terms of their fields of application. Within these categories several standardized technologies and quite a range of proprietary systems are available. But not all of them qualify for every application in demand in the field of industrial automation. One has especially to differentiate between factory and process automation and the differing real-time requirements in these two worlds.

The most real-time requirements come from the area of factory automation where often reaction times in the range of a few hundred milliseconds can be realized by means of standardized technologies. But in some typical use cases vendors need to deviate from standards and implement proprietary features to fulfil customer requirements. These use cases, their specific demands and the optimized technologies used to realize them will be discussed within this speech.

Timeliness in Wireless Sensor Networks: Common Misconceptions

Ramon Serna Oliver and Gerhard Fohler

Chair of Real-Time Systems
Technische Universtät Kaiserslautern, Germany
{*serna_oliver, fohler*}*@eit.uni-kl.de*

Abstract

Existing real-time methods enable wireless sensor networks (WSN) to achieve –in principle– different levels of timeliness guarantees. However, the design and evaluation processes of these methods are often grounded on naive assumptions that constrain their usability in real-world deployments.

In this paper, we analyze from a timeliness perspective a number of implicit and explicit assumptions common in existing methods and discuss their impact in real deployments. We base our arguments on gained experience from simulations under realistic assumptions in WSN as well as well-known research literature. Based on these arguments, we provide a list of considerations to mitigate the effects of misleading assumptions and achieve timeliness solutions consistent with the particularities of WSN.

1. Introduction

The rapid expansion of wireless sensor networks (WSN) [1] in an increasing number of application domains contributes to a growing demand of network services with thorough performance requirements. With the support of a large theoretical and practical background, existing timeliness solutions [2] aim at enabling WSN to operate with real-time guarantees. However, the design and evaluation of these methods are often based on naive assumptions that constrain their applicability in real-world deployments.

The definition of ambitious goals –which cannot be satisfied unless severe assumptions are granted– is one of the major drawbacks of current real-time methods. This overestimation of capacity entails important simplifications during the evaluation process. Among others, common practices include the definition of misleading evaluative criteria and the loss of generality due to ad-hoc test-beds. Hence, the quality assessment from a timeliness perspective becomes unclear because the methods are evaluated against unrealistic models.

This work is partially financed by the European Commission under the Framework 6 IST Project "Wirelessly Accessible Sensor Populations (WASP)".

In this paper, we analyze –from a timeliness perspective– three main aspects of existing real-time solutions for WSN. First, we overview the different goals of existing real-time methods to deem their suitability if applied to realistic WSN. Secondly, we evaluate the impact of a number of implicit and explicit assumptions taken in the design of these methods. In most cases, these assumptions have a significant impact on the real-time performance and constrain the applicability in real-world deployments. Lastly, we examine different evaluation criteria and identify common misconceptions of the evaluative process that can lead to misguided conclusions.

Based on this analysis, we infer a number elementary considerations, which allow mitigating the impact of unrealistic assumptions and facilitate meaningful evaluation tests that increase the confidence of these methods.

The reminder of this paper is organized as follows: section 2 overviews some of the most representative methods in the current state-of-the-art; the following sections present a discussion about common misconceptions and misleading assumptions about real-time objectives (section 3), networking protocols (section 4), and evaluation criteria (section 5); based on this analysis, section 6 presents a series of considerations with the aim of mitigating the inclusion of these misconceptions in future developments; finally, section 7 concludes the paper.

2. Overview of Real-Time in WSN

Applications of WSN can be divided into two main areas: monitoring and tracking [3]. The former includes examples such as monitoring of health parameters in a medical context, environmental control, and structural monitoring of buildings. The latter, includes object tracking in multiple contexts as well as intrusion surveillance of restricted areas. Both domains exhibit inherent demands for real-time guarantees that existing methods try to satisfy at different levels.

Real-time MAC protocols aim at bounded data link transmission times, which are necessary to guarantee forwarding delays in single hop scenarios. Common ways to achieve this include traffic regulation mechanisms [4], scheduling

of message transmissions [5], as well as structured network topologies [6], [7], and prioritized schemes [8].

Routing techniques pursue bounded end-to-end delays for multi-hop scenarios in a broad number of possible ways [9]. Examples include geographic packet forwarding [10], multi-path routing [11], and prioritized queuing models [12]. Multi-layer approaches try to embed the functionalities of different layers into a complete real-time framework with delay guarantees [13].

The analysis of end-to-end latency [14], [15] aims at providing a better understanding of timeliness capacities of WSN. This information enables adaptive methods to adjust their behavior to the current network conditions. In addition, low latency [16], robustness [17] and specially low energy consumption [18], [19], are properties inherently present within the goals of most real-time solutions.

3. Misleading Real-Time Objectives

The solid background of real-time systems is in many aspects a source of inspirations to provide real-time support in new research areas. However, the inherent capacity of satisfying real-time constraints may be significantly different from one domain to another. Overseeing the fundamental incompatibilities between both domains develop in some of the most common misuses of real-time methods applied to WSN, which may lead to unfeasible goals and unrealistic scenarios.

Assumption 1. *The goal of a real-time method is to provide hard real-time guarantees for each transmitted message.*

The notion of hard real-time systems [20], in which each event is associated with a strict deadline, does not match with the general architecture of WSN. Messages are transmitted via hop-by-hop forwarding through unreliable links; the end-to-end delivery ratio is typically low; and the low-energy profile of most communication stacks increases the probability of expiring the maximum retransmission attempts without success. A consequence of these facts is that any individual transmission is susceptible to fail.

Guaranteeing strict deadlines requires excessive resources and complex algorithms for which WSN are not designed. A more elaborated notion of timeliness and the definition of adequate metrics to evaluate the quality of service (QoS), accommodate to a larger extend with the inherent properties of WSN. For example, in [21] the authors present a notion of timeliness which based on the current real-time performance of the network extracts the probability of messages being transmitted within bounded intervals.

Efficient real-time methods should encourage the analysis and exploitation of network trade-offs, adapting their timeliness performance according to the suitability of expending resources.

4. Common Protocols Assumptions

Existing protocols are not free of assumptions. In this section we enumerate a number of misleading assumptions in existing protocols and their implications in realistic scenarios.

Assumption 2. *Availability of resources.*

In a number of existing protocols, it is common practice to base the methods on the assumption of specific hardware resources. Although it is possible to conceive a plausible scenario to justify these assumptions, they are not valid for the general case. For example, GPS devices are mentioned by [22], [19], and [11]; [8] assumes multiple radio transceivers; and [23] and [24] provide solution based on unconstrained nodes acting as access points.

Assumptions on such equipment imply the loss of generality and restrain the applicability of these methods to particular cases. Mitigating the implications of such assumptions by alternative methods strengthens both the validity and applicability of the method. However, the consequences of such substitutions may introduce inaccuracies with respect to the dedicated hardware that must be taken into account.

4.1. Data Link Level

Precise models of radio transceivers and the propagation of waves through the air are inherently complex due to the interaction of a considerably large number of physical laws. However, their accuracy may determine the validity of real-time models built on top of them. The trade-offs between accuracy and simplicity are not straight forward, and lead to different levels of precision. The following aspects have a significant impact on the data link models.

Assumption 3. *Radio links are symmetric and stable over time. Transmission range follows a radial pattern equal to the interference range.*

This set of assumptions has been widely discussed and refuted. Radio transmissions are neither symmetric nor stable over time as shown in [25], [26]. Both studies conclude that the transmission range of omnidirectional antennas is not regular for all directions and varies over time even in static set-ups. In [27], the authors experiment with the vertical placement of nodes and conclude that nodes placed a distance above the ground achieve a significant larger transmission and reception range.

From a timeliness perspective, the implications of unrealistic radio models introduce a number of important drawbacks. In the first place, in real-world scenarios the delivery ratio drops due to radio anomalies [28]. Hence, the necessary mechanisms to ensure successful transmission within strict deadlines must be reenforced. Moreover, further nodes are typically preferred by message forwarders, as they

offer a shorter hop distance till the sink. However, these nodes may be located within the boundaries of the effective transmission range, where links suffer from a high bit error rate (BER). In [29], the authors explore the use of different metrics other than the *distance-to-sink* in order to determine the quality of paths. Their study reveals that the elaboration of a path metric is not straightforward and may require the combination of different indicators.

Broadcast messages, which are often used to build network trees, also suffer from similar effects. For example, a node closer to the sink will broadcast "HELLO" messages to its neighboring nodes, which will then register the source as the forwarding preference for their traffic. However, some of these child nodes may not be able to send their messages back, either due to the non-symmetric range of the radio devices or because of temporal instability. In [30], the authors explore further this effect and propose a simple method to determine stable links based on the consecutive reception of enumerated broadcast messages.

Assumption 4. *A radio transceiver is either in transmitting or receiving mode, or turned off.*

The common assumption with respect to the radio transceiver is that at any time, it is either turned off or in one of two possible states: receiving (Rx) or transmitting (Tx). However, the transition between these two modes produces a third state in which the transceiver is neither listening nor sending out any signal. This, in general, is widely neglected in simulation models, despite accounting for a large number of collisions. In real-world scenarios, it introduces a large enough interval of time –$192\mu s$ in a TI CC2420 [31]– between sensing the channel and being able to start transmitting. During this gap of time, other nodes sensing the medium may also start transmitting, which may lead to collisions if both nodes are within their interference ranges.

From a timeliness perspective, the most relevant impact of this effect is again a notable decrease of the effective delivery ratio, which indirectly affects the performance figures of real-time protocols validated against simplistic models.

Assumption 5. *The received signal strength (RSSI) is proportional to the distance between sender and receiver.*

The relation between RSSI and the distance between the communicating parts is not as straight forward as often assumed. In [32], the authors analyze the signal strength measured at increasing distances and conclude that although *the average* signal strength shows a correlated trend with respect to the distance, this cannot be extrapolated to individual measurements. This conclusion is shared in [33], which additionally explores the correlation between signal strength and packet loss. They found out that typically, high signal strength produces low packet loss, although surprisingly, the opposite statement does not necessarily hold.

4.2. MAC Protocols

Real-time MAC protocols try to guarantee bounded transmission delays between neighbor hops. Their success depend in great measure on carefully defining their operational boundaries. Certain assumptions, as the following, may lead to unsatisfactory results.

Assumption 6. *If no other node in the network is trying to access the medium, the medium is free.*

The assumption of complete isolation with respect the the wireless medium is not safe. Some existing methods (e.g. [34], [35], [36]) and most TDMA scheduling policies (e.g. [37], [38]) are designed under the assumption of having a constant amount of network capacity at their disposal.

Nevertheless, communications may still suffer from external interferences and reduced connectivity due to weak link. As a consequence, messages may result corrupted or not transmitted, despite theoretical guarantee of conflict-free communications provided by the protocol.

Protocol designers must take into account that RF communications are prone to uncontrollable interferences that may enter in conflict with TDMA schedules as well contention-free intervals. The assumption of a a completely isolated environment could be a valid claim for testing purposes. However, the calculation of real-time delay bounds based on this principle is not accurate.

Assumption 7. *WSN can be organized in fixed topologies which remain stable for the entire network life-time.*

The restriction to a particular network topology is common in some real-time protocols (e.g. [7]). In spite of being a legitimate requisite for characteristic scenarios, the implications of such assumptions are questionable in real deployments. In fact, provided that factors such as the radio anomalies discussed in assumption 3 are taken into account, the relation between the physical placement of nodes and their connectivity over time with neighbor nodes is not constant.

4.3. Routing Protocols

Routing protocols are not exempt of misleading assumptions which cannot be always taken for granted.

Assumption 8. *Location-awareness.*

Equipping each sensor node with a GPS device is out of budget for most WSN deployments. Realistic assumptions should be made also with respect to the availability of resources. Nevertheless, multiple location algorithms are available and can be combined with real-time methods. However, it is important to consider the unavoidable error of these algorithms in finding the exact position of a node. For example, the performance of routing protocols based on

geographic forwarding (e.g. [10]) may be directly affected or seriously jeopardized if these errors occur.

Assumption 9. *The maximum length of any routing path is bounded. Hop distance is proportional to physical distance.*

Assuming upper bounds on the number of hops necessary to reach the sink from a given source node (e.g. [18]) is a very practical but unrealistic restriction. The elaboration of routing protocols that define the trajectory of messages towards the sink following the "shortest path" may result in low throughput. In [29], the authors analyze this effect and provide a number of alternative metrics.

Establishing a realistic upper bound requires strong assumptions on the network dynamics which are often out of control. Nevertheless, the establishment of a bound for the"longest possible path" introduces an implicit constrain in the protocol scalability.

Assumption 10. *Messages that cannot satisfy their deadlines are dropped.*

This case may not be considered an assumption but rather a common behavior of real-time routing protocols as a consequence of aiming at strict deadlines (see assumption 1). In most WSN scenarios with timeliness requirements, there is an added value to the *freshness* of data. Following this principle, old messages are often discarded at intermediate hops if the algorithm estimates that their end-to-end deadline cannot be fulfilled.

However, guaranteeing end-to-end delays is not effective if the protocol itself contemplates the possibility of dropping unsuccessful messages based on estimates. In some cases, receiving old data may produce better results than receiving no data at all. Alternative approaches may consider adaptive methods with the ability of defining flexible deadlines.

5. Imprecise Evaluation Criteria

Choosing meaningful evaluation criteria has a great impact on the performance figures and the quality of the evaluation procedure. In this section, we discuss some important misconceptions affecting the generalization of evaluation analysis in realistic scenarios.

5.1. Misleading Theoretical Proofs

Assumption 11. *Everything can be turned into an analytical expression.*

With the use of properly validated models, meaningful bounds for the network latency or other performance metrics can be inferred. However, in many cases the necessary level of abstraction introduces serious simplifications of complex systems; for example: assumptions about traffic pattern distributions, service times, or the minimum network density.

Analysis of *average-case scenarios* provide theoretical bounds for the figures of interest. However, introducing all possible factors that could interfere in the *worst-case scenario* is practically unfeasible in analytical expressions.

Assumption 12. *The distribution of average (service time/transmission latency/queue size) is constant during the entire network life-time.*

This assumption is correlated with the previous and reflects the unfeasibility of analytical expressions to capture the dynamic behavior of a WSN.

5.2. Simplistic Simulations Models

Assumption 13. *"Our model reflects accurately the physical properties of ..."*.

Due to the complexities of physic laws and the propagation of waves, a realistic radio model including all possible anomalies is practicably unfeasible. Channel access, environment, and interferences are as important to model as the method being evaluated. Simplistic models may hide design flaws or applicability limitations that appear in real-world deployments.

Experiments such as [33], [28], and [39] show that the deviation between simulation results and real test-beds are not negligible. However, the additional level of complexity involving a real test-bed is not always affordable.

Nevertheless, an appropriate validation process can lead to sufficient levels of accuracy for the most significant figures. For example, in [40], the authors profile the necessary steps to achieve accurate evaluations of timeliness protocols with properly tuned simulations.

6. Considerations

Designing and implementing timeliness methods for WSN without relying on misleading assumptions is a challenge that still needs further attention. The definition of appropriate objectives and a careful validation of models are crucial to achieve high quality methods.

The following list of considerations summarize the main problems of existing methods, and may help overcome a number of popular misconceptions constraining the quality of timeliness solutions:

- Hard real-time solutions require strict deterministic models that are not compatible with WSN. Adaptive methods and a proper definition of QoS trade-offs may reduce the number of necessary restrictive assumptions.
- Realistic radio models are difficult to achieve, yet crucial in the evaluation of timeliness models. The careful validation of data link models plays a significant role in the elaboration of satisfactory methods.
- RF communications in WSN are typically exposed to many sources of interferences. MAC protocols have to

be robust enough to deal with unstable channels and weak links.

- Effective routing protocols should be able to support timeliness without requiring restrictive resources. Scalability and adaptiveness are also important figures to evaluate.
- Validation criteria must be consistent with the scenarios for which the evaluated methods are designed. Simplistic models may lead to optimistic figures that do not match the real performance.

7. Conclusion

In this paper, we enumerated a number of misleading assumptions that are found in many existing real-time methods for wireless sensor networks. Our analysis is conducted from a timeliness perspective with the goal of identifying the source of common misconceptions with a negative impact on the real-time performance.

Based on existing literature and gained experience of simulations under realistic assumptions, we presented argumentation against misleading evaluation and validation criteria leading to imprecise conclusions.

We completed our analysis with a series of considerations that may help mitigate the effects of misleading assumptions.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] Y. Li, C. S. Chen, Y.-Q. Song, and Z. Wang, "Real-time qos support in wireless sensor networks: a survey," in *Proceedings of the 7th IFAC, International Conference on Fieldbuses and Networks in Industrial and Embedded Systems - FeT'2007*, 2007.

[3] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[4] K. Karenos and V. Kalogeraki, "Real-time traffic management in sensor networks," in *Proceedings of the 27th IEEE Real-Time Systems Symposium (RTSS)*, 2006.

[5] O. Chipara, C. Lu, and G.-C. Roman, "Real-time query scheduling for wireless sensor networks," in *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, Dec. 2007, pp. 389–399.

[6] T. Watteyne, I. Augé-Blum, and S. Ubéda, "Dual-mode real-time mac protocol for wireless sensor networks: a validation/simulation approach," in *InterSense '06: Proceedings of the first international conference on Integrated internet ad hoc and sensor networks*. New York, NY, USA: ACM, 2006, p. 2.

[7] K. Shashi Prabh and T. Abdelzaher, "On scheduling and real-time capacity of hexagonal wireless sensor networks," in *Proceedings of the 19th Euromicro Conference on Real-Time Systems, ECRTS07*, July 2007, pp. 136–145.

[8] M. Caccamo, L. Zhang, L. Sha, and G. Buttazzo, "An implicit prioritized access protocol for wireless sensor networks," in *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS)*, 2002.

[9] J. Al-Karaki and A. Kamal, "Routing techniques in wireless sensor networks: a survey," *Wireless Communications, IEEE*, vol. 11, no. 6, pp. 6–28, Dec. 2004.

[10] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "Speed: A stateless protocol for real-time communication in sensor networks," in *Proceedings of ICDCS'03*, May 2003.

[11] E. Felemban, C.-G. Lee, and E. Ekici, "Mmspeed: multipath multi-speed protocol for qos guarantee of reliability and timeliness in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 738–754, June 2006.

[12] K. Akkaya and M. Younis, "An energy-aware qos routing protocol for wireless sensor networks," in *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, May 2003, pp. 710 – 715.

[13] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He, "Rap: A real-time communication architecture for large-scale wireless sensor networks," in *Proceedings of the IEEE RTAS 2002*, 2002.

[14] R. Serna Oliver and G. Fohler, "Probabilistic estimation of end-to-end path latency in wireless sensor networks," in *Proceedings of the Sixth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS09)*, Macau SAR, P.R.C, October 2009.

[15] Y. Wang, M. Vuran, and S. Goddard, "Cross-layer analysis of the end-to-end delay distribution in wireless sensor networks," in *Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE*, Dec. 2009, pp. 138–147.

[16] S. Gobriel, R. Cleric, and D. Mosse, "Adaptations of tdma scheduling for wireless sensor networks," in *Proceedings of the 7th International Workshop on Real-Time Networks (RTN)*, 2008.

[17] T. He, B. M. Blum, Q. Cao, J. A. Stankovic, S. H. Son, and T. F. Abdelzaher, "Robust and timely communication over highly dynamic sensor networks," *Real-Time Syst.*, vol. 37, no. 3, pp. 261–289, 2007.

[18] S. C. Ergen and P. Varaiya, "Energy efficient routing with delay guarantee for sensor networks," *Wireless Networking*, vol. 13, no. 5, pp. 679–690, 2007.

[19] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher, "Real-time power-aware routing in sensor networks," in *Proceedings of the 14th IEEE International Workshop on Quality of Service (IWQoS).*, June 2006.

[20] G. C. Buttazzo, *Hard Real-Time Systems: Predictable Scheduling Algorithms and Applications*, 2nd ed. Springer, 2005.

[21] R. Serna Oliver and G. Fohler, "A proposal for a notion of timeliness in wireless sensor networks," in *Proceedings of the 8th International Workshop on Real-Time Networks RTN'09*, Dublin, Ireland, June 2009.

[22] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher, "A spatiotemporal communication protocol for wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 10, pp. 995–1006, 2005.

[23] S. C. Ergen and P. Varaiya, "Pedamacs: Power efficient and delay aware medium access protocol for sensor networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 7, pp. 920–930, 2006.

[24] P. Pothuri, V. Sarangan, and J. Thomas, "Delay-constrained, energy-efficient routing in wireless sensor networks through topology control," in *Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, 2006. ICNSC '06.*, 0-0 2006, pp. 35 –41.

[25] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *In Proceedings ACM SenSys '03*. ACM Press, 2003, pp. 14–27.

[26] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of radio irregularity on wireless sensor networks," in *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM, 2004, pp. 125–138.

[27] M. Holland, R. Aures, and W. Heinzelman, "Experimental investigation of radio performance in wireless sensor networks," in *Wireless Mesh Networks, 2006. WiMesh 2006. 2nd IEEE Workshop on*, Sept. 2006, pp. 140 –150.

[28] V. Turau, M. Witt, and C. Weyer, "Analysis of a real multi-hop sensor network deployment: The heathland experiment," in *in Proc. Third International Conference on Networked Sensing Systems (INSS 2006)*, 2006.

[29] D. S. J. De Couto, D. Aguayo, B. A. Chambers, and R. Morris, "Performance of multihop wireless networks: shortest path is not enough," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 83–88, 2003.

[30] A. Herms, G. Lukas, and S. Ivanov, "Realism in design and evaluation of wireless routing protocols," in *Proceedings of First international Workshop on Mobile Services and Personalized Environments (MSPE'06)*, 2006.

[31] *CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*. Chipcon / Texas Instruiments, 2008.

[32] C. Newport, D. Kotz, Y. Yuan, R. S. Gray, J. Liu, and C. Elliott, "Experimental evaluation of wireless simulation assumptions," *Simulation*, vol. 83, no. 9, pp. 643–661, 2007.

[33] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems. SenSys03*. New York, NY, USA: ACM, 2003, pp. 1–13.

[34] K. Mizanian, R. Hajisheykhi, M. Baharloo, and A. Jahangir, "Race: A real-time scheduling policy and communication architecture for large-scale wireless sensor networks," in *Communication Networks and Services Research Conference, 2009. CNSR '09. Seventh Annual*, May 2009, pp. 458–460.

[35] T. Abdelzaher, S. Prabh, and R. Kiran, "On real-time capacity limits of multihop wireless sensor networks," in *Proceedings of the IEEE International Real-Time Symposium (RTSS)*, 2004.

[36] T. Watteyne and I. Auge-Blum, "Proposition of a hard real-time mac protocol for wireless sensor networks," in *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE Computer Society, 2005.

[37] A. Sahoo and P. Baronia, "An energy efficient mac in wireless sensor networks to provide delay guarantee," in *Proceedings of the 15th IEEE Workshop on Local and Metropolitan Area Networks. LANMAN*, June 2007.

[38] C. X. Mavromoustakis and H. D. Karatza, "Real-time performance evaluation of asynchronous time division traffic-aware and delay-tolerant scheme in ad hoc sensor networks," *International Journal of Communication Systems*, vol. 23, no. 2, pp. 167 – 186, Sep 2009.

[39] M. D. Yarvis, W. S. Conner, L. Krishnamurthy, J. Chhabra, and B. Elliott, "Real-world experiences with an interactive ad hoc sensor network," in *ICPPW '02: Proceedings of the 2002 International Conference on Parallel Processing Workshops*. Washington, DC, USA: IEEE Computer Society, 2002, p. 143.

[40] J. Rousselot, J.-D. Decotignie, M. Aoun, v. d. S. Peter, R. Serna Oliver, and G. Fohler, "Accurate timeliness simulations for real-time wireless sensor networks," in *European Modelling Symposium 2009*, Athens, Greece, November 2009.

Bandwidth Isolation for Composability in Fixed Priority Real-Time Networks

Daniel Sangorrín
Nagoya University, Japan
dsl@ertl.jp

Michael González Harbour
University of Cantabria, Spain
mgh@unican.es

Abstract

The increasing complexity of real-time systems has motivated the application of component-based software engineering principles during the last few years. Temporal encapsulation is key to smoothing the integration stage of software components in complex distributed hard real-time systems. This paper presents a network scheduling server algorithm to guarantee and at the same time limit the network bandwidth assigned to streams of messages with different real-time requirements in a fixed priority network. The algorithm is based on a recently corrected version of the POSIX sporadic server whose rules, originally intended for scheduling tasks, have been adapted and optimized for the special case of fixed-priority networks. The algorithm is able to provide bounded response times that can be analyzed with off-the-shelf real-time analysis tools and can be used for both synchronous and asynchronous messages. The proposed approach has been implemented and evaluated on real hardware, using the CAN bus. The performance evaluation results show that bandwidth isolation can be achieved with rather low overhead both on the processor and the network resources.

1. Introduction

The complexity of developing large real-time applications can be handled by independently developing components that are later integrated into a physical platform. The success of the integration depends on the ability of the platform to provide the required resource usage guarantees to every component while protecting each component from timing faults in the others. Some compositional frameworks [5] have an integrated view of the different resources involved in a distributed application. In particular, for network resources, application components are able to specify their bandwidth requirements, so that the implementation can make the corresponding guarantees or reservations.

In real-time distributed systems, the communication paradigm (event- or time-triggered [10, 23]) plays an important role in the composability, flexibility and responsiveness of the system. In the *time-triggered* paradigm, messages are sent at predefined time windows according to a global schedule. This approach is well-suited for periodic activities that require very low jitter. Furthermore, it enables composability regarding to the temporal behavior because the access to the bus is predefined and decoupled from the actual network load. The major drawbacks of this approach are the lack of flexibility in the design process, the need of global synchronization between the nodes and its poor support for aperiodic messages. In the *event-triggered* paradigm, messages are sent as a response to the occurrence of an event. This approach is generally more flexible and better suited to support asynchronous traffic together with critical activities that require very short response times. The major disadvantages of this approach are the increased message jitter and the lack of temporal isolation.

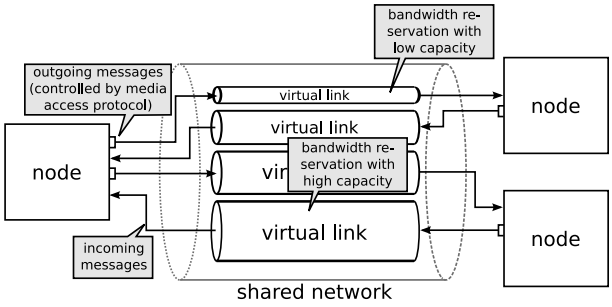

Figure 1: Bandwidth reservations

Since both paradigms have strong and weak points, several protocols that combine support for both event- and time-triggered traffic have been proposed. In some protocols (e.g., FlexRay [3] or FTT-CAN [15]), temporal isolation between both types of traffic is enforced by implementing a cyclic sequence that alternates between them. However, the arbitration of the event-triggered phase of these protocols, implemented with different approaches such as fixed priorities (FTT-CAN) or TDMA with minislots (FlexRay), has the same drawbacks that were mentioned before, high jitter and lack of temporal isolation. For instance, in case of a software babbling idiot failure [13], a misbehaving task may affect the bandwidth preallocated to other tasks that are working correctly by transmitting excessive messages at a higher priority.

This paper presents a network scheduling algorithm that follows the event-triggered paradigm and is able to satisfy the requirements for the integration of independently developed components. The algorithm is based on a recently corrected version [25] of the POSIX sporadic server whose rules, originally intended for scheduling tasks, have been adapted and optimized for the special case of fixed-priority networks. The algorithm is able to control the jitter caused by aperiodic messages, provide bounded response times

that can be analyzed with off-the-shelf real-time analysis tools and it can be used for both synchronous and asynchronous messages. It enables the creation of bandwidth reservations, which can be thought as unidirectional virtual links between two nodes providing a guaranteed service, as shown Fig. 1.

The paper is organized as follows. After an introduction to server-based scheduling for networks in Sec. 2, Sec. 3 proposes an algorithm for an optimized version of the sporadic server policy for fixed-priority networks. Sec. 4 gives details about the implementation of the algorithm on real hardware, whose overhead is evaluated in Sec. 5. Sec. 6 compares the proposal with previous work and Sec. 7 closes the paper with conclusions.

2. Server-based scheduling in networks

Server-based scheduling techniques have been used for a long time to limit the processor time assigned to a particular computation or set of computations while also guaranteeing some minimum level of service. Servers such as the periodic server [14], the sporadic server [24], or the constant bandwidth server [9] are a few examples.

The concept of server is also applicable to the outgoing direction of a network stack to limit the bandwidth used by message streams. However, scheduling in the networks is somehow different than in the processors. When a server is used to schedule a network the concept of execution time must be mapped into transmission time. In most networks, messages are fragmented in units called packets which are usually non preemptible. Therefore, an easy way to specify budgets in a network server is to measure them in terms of number of packets. The maximum packet size is usually limited by the network, but a smaller limit can also be imposed by the implementation as necessary, for instance as an application-defined parameter. Of course, if the message stream mixes very short messages with longer messages that fit into the maximum packet size, the bandwidth available to the message stream may be suboptimal, since each message consumes one unit of budget regardless of its size. However, it is easy to design a solution to this problem by creating several sporadic servers with different maximum packet sizes, and submitting the messages to the appropriate server based on their size. The non-preemptability of the network packets has bounded delay effects that can be easily modeled through a blocking time term. Other more complex analysis models can also be used to better estimate response times [12].

2.1. Sporadic server

The sporadic server is a bandwidth preserving scheduling algorithm designed for processing aperiodic events in hard real-time systems [24, 18]. It allocates a specific bandwidth for processing aperiodic requests at a given priority level (the normal priority). This bandwidth is provided by allocating a certain execution time capacity for each interval of time called the replenishment period. The scheduling algorithm is defined through a set of rules for consuming this execution capacity when the sporadic server runs, and

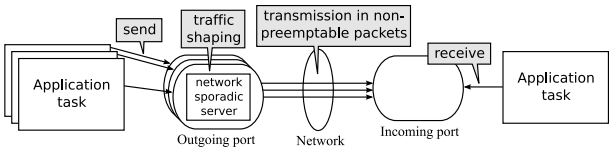

Figure 2: Communication elements

for later replenishing this capacity. When the capacity is consumed, the sporadic server may still do useful work at a background priority level, to make full use of the resource.

When the sporadic server was standardized in the additional real-time extensions [6] of the POSIX standard for portable operating system interfaces (later included in the unified version of the standard [7]), it was defined with a set of consumption and replenishment rules, intended to allow for a feasible implementation in the context of a real-time operating system (RTOS). Unfortunately, except for some specific cases, the new rules had the same problem of the original sporadic server definition that could cause preemptions to occur too early [18]. Recently, in [25], a new set of rules has been proposed in order to fix the original POSIX sporadic server problem while maintaining its main value, the simplicity of its implementation.

In the next section, the sporadic server proposed in [25] is adapted and optimized for the case of fixed priority networks. To simplify the implementation, the presented approach takes advantage of the discrete nature of the network packets and considers that the capacity chunks used in the sporadic server are always of size one. This allows to create a capacity queue of fixed size, equal to the number of packets represented in the budget of the sporadic server. Each packet in the capacity queue is annotated with its replenishment time. This simplifies the budget arithmetics and eliminates the need to introduce optimizations to limit the fragmentation of the capacity.

3. Network Sporadic Server algorithm

The proposed network sporadic server policy is based primarily on two parameters: the replenishment period and the initial transmission capacity. The replenishment period is called *repl_period* and is measured as an absolute time. The initial transmission capacity is called the *init_budget* and is an integer number of network packets of bounded size. As shown in Fig. 2, the network sporadic server policy is used to schedule a stream of messages that are sent from a specific sender node in the system, through the network. The destination node of these messages is any node that is reachable in a single hop. Messages to be sent are submitted by the application and stored in a transmission queue until they are sent. Messages in this queue fit into one packet, but a fragmentation layer is provided outside the sporadic server implementation if larger messages are required.

Fig. 3 shows the architecture of the network sporadic servers. For each sporadic server the system maintains in the sender node a capacity queue, with transmission capacity chunks. The size of the queue is equal to *init_budget*. Each chunk represents a transmission capacity of one

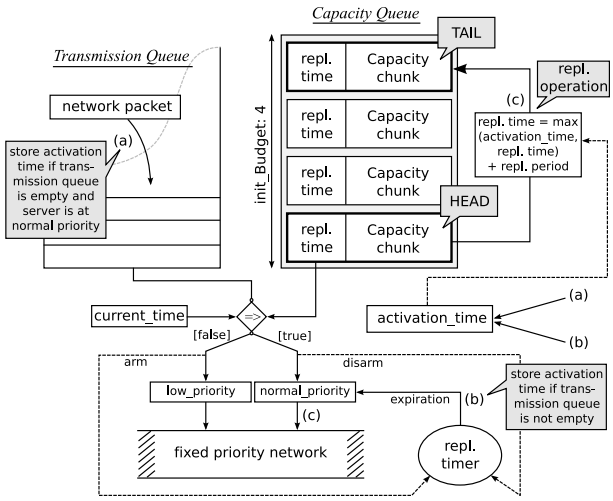

Figure 3: Network Sporadic Server

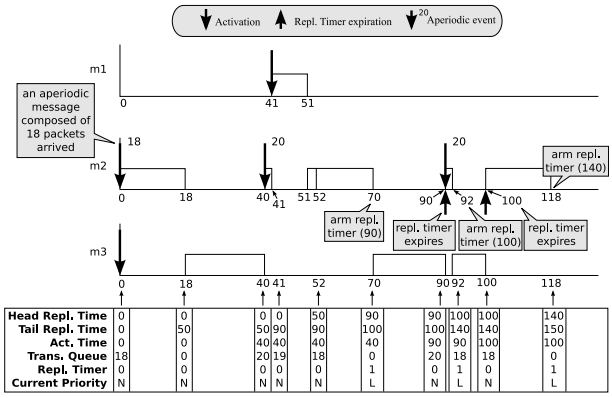

Figure 4: Scheduling sequence using the network sporadic server

packet, and contains a replenishment time, which is an absolute time after which the capacity may be consumed. Initially, all the chunks in the queue have a replenishment time equal to the time at which the queue is initialized. In addition, the system keeps one value associated with each sporadic server: an absolute time called the *activation_time*. Finally, the system has a conceptual replenishment timer associated with each sporadic server.

The priority assigned to messages sent though a sporadic server is determined in the following manner: if the replenishment time of the head of the capacity queue is equal to or earlier than the current time, the server is considered to have execution capacity available, so it is assigned the priority specified by *normal_priority*, and its replenishment timer is disarmed; otherwise, the assigned priority shall be *low_priority*, and the replenishment timer is armed to expire at the replenishment time of the head of the capacity queue. The modification of the capacity queue and, consequently of the assigned priority, is done as follows:

1. Each time the server is made ready at the *normal_priority* level, either because a new message arrived at the transmission queue while it was empty (path (a) in Fig. 3) or because the replenishment timer expired and the transmission queue is not empty (path (b)), the time at which this operation is done is stored in the *activation_time*
2. When a message is sent at the *normal_priority* level a replenishment operation is performed (path (c)), as described in 3. Then, if the replenishment time of the new head of the capacity queue is larger than the current time, the server is assigned the *low_priority* and the replenishment timer is armed to expire at the replenishment time of the head of the capacity queue.
3. Each time a replenishment operation is performed the head of the capacity queue is removed from the queue and reinserted at the tail with a replenishment time equal to the maximum of the *activation_time* and its current replenishment time, plus *repl_period* (see path (c) in Fig. 3).
4. When the replenishment timer expires the server is assigned the *normal_priority* level.

3.1. Example

The following example illustrates the presented network sporadic server algorithm. Consider a system with three periodic message streams with parameters shown in Table 1, with deadline-monotonic priority ordering. Suppose that m_2 is a network sporadic server to transmit aperiodic messages. The sporadic server is given a initial transmission capacity of $C_2 = 20$ packets, and a replenishment period $T_2 = 50$ time units. For simplicity, the transmission time of one packet is supposed to take one time unit.

Table 1: Periodic message streams

Message	C_i	T_i	D_i
m_1	10	200	20
m_2	20	50	50
m_3	50	200	100

Fig. 4 shows a transmission sequence scheduled under the network sporadic server policy defined in this paper. It also shows the evolution of the replenished time value in the head and tail of the capacity queue, the *activation_time* variable associated to the server, the number of packets in the transmission queue, the current priority of the server, normal (N) or low (L) and the status of the replenishment timer which can be armed (1) or disarmed (0). The example is similar to the one used in [25] to illustrate the correction of the premature replenishments defect in the original POSIX sporadic server.

4. Implementation

The network sporadic server policy defined in Sec. 3 has been implemented on real hardware. As a relevant example of fixed priority networks, the Controller Area Network (CAN) [2] was chosen. CAN has been used extensively in the automotive industry to connect Electronic Control Units (ECUs) using a shared bus. CAN features non-preemptive frame transmission and priority-based arbitration through

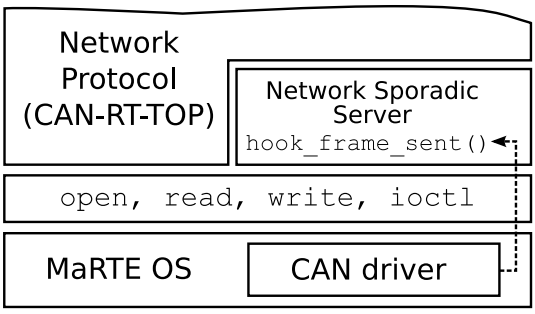

Figure 5: Implementation layers

a bit dominance protocol which enables bounded latencies that can be analyzed through real-time schedulability theory. Popularity of CAN has reached other sectors such as industrial control applications or medical equipment.

Fig. 5 shows the main elements of the implementation. The architecture was implemented on MaRTE OS [11], a hard real-time operating system that follows the Minimal Real-Time POSIX.13 subset and provides an easy-to-use and controlled environment to develope multi-thread real-time applications. The core of MaRTE OS is written in Ada language, and it supports mixed-language applications in Ada, C and C++. In this study, MaRTE OS was extended with a driver that supports the NXP SJA1000 chipset [8], a stand-alone controller for CAN commonly used within automotive and general industrial environments. The driver provides a POSIX character interface (i.e., `open`, `read`, `write`, etc.). In addition, several hooks can be installed inside the driver through the `ioctl` system call.

The implementation of the network sporadic server executes on top of the MaRTE OS interface. The main sources of overhead introduced by the network sporadic servers, compared to using the native CAN protocol, are the following:

- The replenishment operations (see path (c) in Fig. 3) which are executed every time a CAN frame is sent. A hook is installed in the CAN bus driver, through the `ioctl` system call, in order to be notified about the transmission of a CAN frame.
- A replenishment thread, which waits for expirations of the replenishment timers, modifies the priority of the server and updates the activation time.

In addition, as shown in Fig. 5, a previously presented high-level protocol for CAN (CAN-RT-TOP [19]) was adapted to send messages through the developed network sporadic server. Details about the adaptation of the protocol and its source code, distributed under the GNU/GPL v2 license, can be obtained at [4].

5. Evaluation

This section presents evaluation results of the network sporadic server presented in this paper. The evaluation environment consisted of nodes equipped with AMD Duron 800 Mhz processors, 256 MB RAM memory and Adlink PCI-7841 CAN bus cards [1], which are based on the

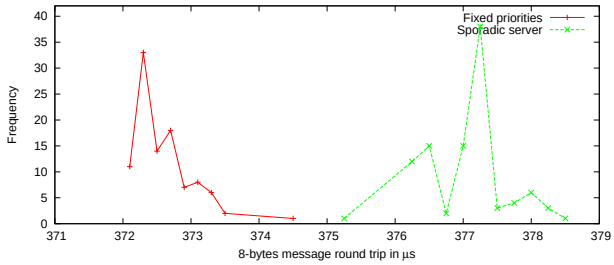

Figure 6: 8-bytes message round-trip measures

NXP SJA1000 controller. The CAN bus was configured to 1Mbps and CAN 2.0B mode. MaRTE OS version 1.8 was built, using the AdaCore GNAT/GPL 2007 (gcc 4.1.3) compiler, with default options which disable assertions, inlines code and perform some optimizations for targets with Local APIC.

5.1. Message round-trip latency measures

In order to measure the influence of the network sporadic server on the end-to-end latency of the message streams, two nodes were connected through the CAN bus and programmed to send query-reply messages continuously under two scenarios: using fixed priorities and using the proposed network sporadic server. Each measure, defined as a round-trip measure, was taken from the instant when the message was sent and the moment when the reply was received. Measures were repeated for 100 times for different message sizes.

Table 2: Maximum round-trip measured values

Bytes	Fixed Priorities	Sporadic Server
8	0.375 ms	0.379 ms
32	1.355 ms	1.372 ms
64	2.643 ms	2.673 ms
512	20.78 ms	21 ms
1488	60.37 ms	61.03 ms

Fig. 6 shows the comparison of the measured values when using 8-bytes messages, which fit in the maximum size of a CAN frame and therefore do not require fragmentation. Table 2 shows the comparison of the maximum measured values for several message sizes. The overhead introduced by the network sporadic server is rather small compared to the transmission times.

5.2. Overhead in the CPU

Table 3 shows execution-time measures of the main sources of processor overhead caused by the sporadic server policy. The first row represents the overhead associated to a replenishment operation. The second row represents the execution time of the body of the replenishment thread which is executed on every replenishment timer expiration.

In order to better evaluate the influence that the measured values, shown in Table 3, represent on the total CPU overhead, simulations of the network sporadic server exe-

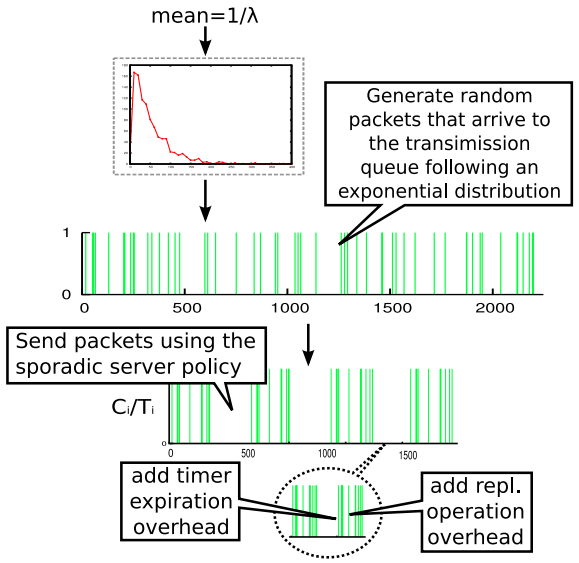

Figure 7: Simulation procedure for the estimation of the CPU overhead under different configurations

Table 3: Sporadic servers measured CPU execution time (in μs)

Measure	Min	Avg	Max
Repl. Program	0.78	0.81	2.34
Repl. Thread	2.69	2.88	3.52

cution have been performed under different configurations. Fig. 7 depicts the procedure followed during the simulations. First, 1000 random aperiodic events (packets arriving to the transmission queue) are generated according to an exponential distribution. Then, packets are sent using the presented sporadic server policy. Each time a replenishment operation or a timer expiration occurs, the corresponding CPU overhead is accounted (maximum measured overhead values, 2.34μs and 3.52μs, were used). When all packets are sent, the total overhead time is divided by the total time to get the overhead as a percentage. The sporadic server was configured with an utilization equal to the mean of the packet inter-arrival instants. It has the highest priority in the network (to evaluate its performance in isolation) and it never transmits at low_priority (i.e., because there are always lower priority messages being transmitted). For simplicity, each packet is supposed to occupy the bus for a constant time of 1 ms. Simulations were repeated for different inter-arrival rates and different server budget/period configurations.

Table 4 contains the overhead results for several inter-arrival rates (defined by $1/\lambda$) of aperiodic events. The overhead is rather small and can be decreased even more by configuring the sporadic server appropriately. Fig. 8 separates the overhead caused by replenishment operations from the timer expirations. Replenishment operations cause a constant overhead since they appear each time a packet is sent. On the other hand, timer expirations overhead can be reduced considerably by increasing the capacity of the server.

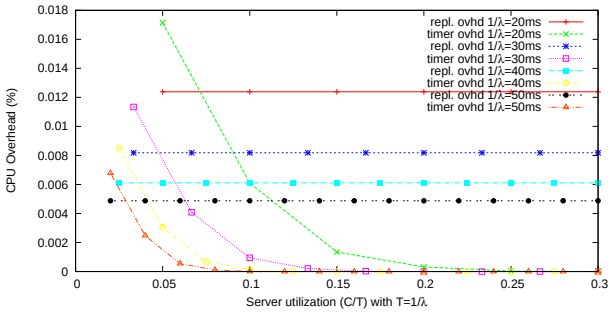

Figure 8: replenishments overhead vs. timer overhead

Table 4: Simulated sporadic server CPU overhead (in %) with parameters $Budget = n$ packets and $Period = n \cdot \frac{1}{\lambda}$

$1/\lambda$	$n=2$	$n=4$	$n=6$	$n=8$	$n=10$
20ms	0.0286	0.0284	0.0275	0.0269	0.0266
50ms	0.0114	0.0112	0.0111	0.0110	0.0109
250ms	0.0023	0.0023	0.0022	0.0022	0.0022
1000ms	0.0005	0.0005	0.0005	0.0005	0.0005

6. Related work

The leaky bucket concept used in network traffic shaping [17, 16] is similar to the concept of server-based scheduling. The leaky bucket algorithm is useful to control that the traffic is sent to the network at a constant rate. However, it does not handle efficiently the available bandwidth since the leak rate is a fixed parameter and, there may be instances when the network is unused while there are packets pending to be sent. The benefits of the network sporadic server when compared to the leaky bucket are a higher capacity, a shorter response time and minimal interference on lower priority tasks, because the available execution capacity is usable without delay at the specified priority level, and because the effects on lower priority tasks are no worse than those of an equivalent periodic task with an execution time equal to the execution capacity, and period equal to the replenishment period.

In [21], server-based mechanisms based on dynamic priorities (EDF) were proposed for scheduling the CAN bus [2]. The algorithms proposed in that work are based on a master-slave architecture where nodes are synchronized to a trigger message sent periodically by the master. Although the use of dynamic priorities may allow optimal resource utilization, the overhead generated by the necessary synchronization messages and the scheduling algorithm must be taken into account. The benefits of the network sporadic server when compared to that work, are the ability to provide faster response times while minimizing the overhead and the fact that it does not require a complicated implementation.

In addition, the network sporadic server can be integrated with previously presented protocols that organize the bus time as a sequence of time- and event-triggered windows. Fig. 9 depicts an FTT-CAN [15] cycle, divided into synchronous and asynchronous windows. The use of net-

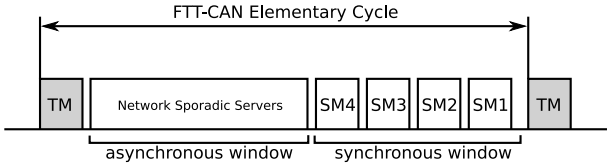

Figure 9: Network sporadic server integrated with FTT-CAN

work sporadic servers in the asynchronous window makes it possible to provide bandwidth isolation between aperiodic message streams. For instance, if a software babbling idiot failure [13] occurs, the messages transmitted by the misbehaving task would be shaped by the server and would not affect the deadlines of other message streams with lower priority.

7. Conclusions

The sporadic server is a very interesting scheduling policy for handling resource reservations, which are key to smoothing the integration stage of software components in complex distributed hard real-time systems. This paper described how to adapt a recently corrected version of the POSIX sporadic server, originally intended for scheduling tasks, to the case of fixed-priority networks. The algorithm was optimized to take into account the discrete nature of the network packets. An implementation on the CAN bus was also described together with its evaluation. The measured performance shows that bandwidth isolation can be achieved at rather low overhead both on the processor and the network resources. The algorithm can be applied to other networks where message streams compete for the media access through fixed priorities. For example, a porting exists to provide reservations on the RTEP protocol [20]. The work presented in this paper has been used in [22, 5] to implement contract-based network bandwidth reservations in the context of a flexible scheduling framework.

References

[1] ADLINK Website. `http://www.adlinktech.com/`.

[2] CAN Specification Version 2.0. 1991, Robert Bosch GmbH, Postfatch 30 02 40, D-70442 Stuttgart.

[3] FlexRay Consortium. `http://www.flexray.com/`.

[4] FRESCOR Fieldbus Systems (D-ND1). `http://www.frescor.org/index.php?page=publications`.

[5] FRESCOR Website. `http://www.frescor.org`.

[6] IEEE Std. 1003.d-1999. Information Technology -Portable Operating System Interface (POSIX)- Part 1: System Application Program Interface (API) Amendment: Additional Realtime Extensions [C Language]. The Institute of Electrical and Electronics Engineers.

[7] ISO/IEC 9945-1:2003. Standard for Information Technology -Portable Operating System Interface (POSIX).

[8] NXP Website. `http://www.nxp.com`.

[9] L. Abeni and G. Buttazzo. Integrating multimedia applications in hard real-time systems. In *RTSS '98: Proceedings of the IEEE Real-Time Systems Symposium*, pages 4–13, Washington DC, USA, 1998. IEEE Computer Society.

[10] A. Albert. Comparison of Event-Triggered and Time-Triggered Concepts with Regard to Distributed Control Systems. In *Proceedings of Robert Bosch GmbH Embedded World*, pages 235–252, Nuremberg, February 2004. `http://www.semiconductors.bosch.de/pdf/embedded_world_04_albert.pdf`.

[11] M. Aldea and M. González Harbour. MaRTE OS: An Ada Kernel for Real-Time Embedded Applications. In *Proceedings of the International Conference on Reliable Software Technologies*, Ada-Europe-2001, Leuven, Belgium, May 2001. Lecture Notes in Computer Science. `http://marte.unican.es`.

[12] R. J. Bril, J. J. Lukkien, and W. F. J. Verhaegh. Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited. In *ECRTS '07: Proceedings of the 19th Euromicro Conference on Real-Time Systems*, pages 269–279, Washington, DC, USA, 2007. IEEE Computer Society.

[13] I. Broster and A. Burns. The Babbling Idiot in Event-triggered Real-time Systems. In *Proceedings of the Work-In-Progress Session, 22nd IEEE Real-Time Systems Symposium*, pages 25–28, 2001.

[14] G. C. Buttazzo. *Hard Real-Time Computing Systems*. Kluwer Academic Publishers, 2002.

[15] J. Ferreira, P. Pedreiras, L. Almeida, and J. A. Fonseca. The FTT-CAN Protocol for Flexibility in Safety-Critical Systems. *IEEE Micro*, 22:46–55, 2002.

[16] E. Hernández and J. Vila. A new approach to optimize bandwidth reservation for real-time video transmission with deterministic guarantees. *Real-Time Imaging*, 9(1):11–26, 2003.

[17] C. F. John Evans. Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice. Morgan Kaufmann Publishers, 2007.

[18] J. Liu. Real-Time Systems. Prentice Hall, 2000.

[19] J. López Campos, J. J. Gutiérrez, and M. González Harbour. CAN-RT-TOP: Real-Time Task-Oriented Protocol over CAN for Analyzable Distributed Applications. In *Proceedings of the 3rd International Workshop on Real-Time Networks (formerly RTLIA)*, Catania, Sicily (Italy), 2004.

[20] J. M. Martínez and M. González Harbour. RT-EP: A Fixed-Priority Real Time Communication Protocol over Standard Ethernet. In *10th International Conference on Reliable Software Technologies, Ada-Europe*, pages 180–195. Springer, June 2005.

[21] T. Nolte, M. Nolin, and H. Hansson. Real-Time Server-Based Communication for CAN. *IEEE Transactions on Industrial Informatics*, 1(3):192–201, August 2005.

[22] D. Sangorrín, M. González Harbour, H. Pérez, and J. Javier Gutiérrez. Managing Transactions in Flexible Distributed Real-Time Systems. In *Proceedings of the 15th International Conference on Reliable Software Technologies*, Ada-Europe 2010, Valencia, Spain, June 2010.

[23] J. Scarlett and R. Brennan. Re-evaluating Event-Triggered and Time-Triggered Systems. In *11th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA '06*, pages 655–661, Sept. 2006.

[24] B. Sprunt, L. Sha, and J. Lehoczky. Aperiodic Task Scheduling for Hard-Real-Time Systems. In *The Journal of Real-Time Systems 1(1)*, pages 27–60. Kluwer Academic Publishers, 1989.

[25] M. Stanovich, T. P. Baker, and M. González Harbour. Defects of the POSIX Sporadic Server and How to Correct Them. In *Proceedings of the 16th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2010*, Stockholm, Sweden, April 2010.

Computing optimal window sizes to enforce dependable communications in time-triggered Controller Area Networks

Michael Short
Electronics & Control Group,
Teesside University,
Middlesbrough, UK.
e-mail: m.short@tees.ac.uk

Imran Sheikh
Embedded Systems Laboratory,
University of Leicester,
Leicester, UK.
e-mail: si52@le.ac.uk

***Abstract*—The enforcement of time-triggered communications over Controller Area Network (CAN) has recently become a topic of great interest, as it is thought to increase the predictability and overall reliability of a network. A potential drawback with existing time-triggered CAN implementations lies in the enforcement of single-shot message transmissions; single bit-errors directly lead to message omissions, and duplicated message instances will often be required for critical message streams. To address this problem, this paper proposes a simple technique to act as an extension to existing time-triggered transmission schemes, in which an effective 'window' is defined for each message. A bounded amount of re-transmission is allowed for each message within its defined window, providing increased reliability in the presence of errors. Message instances will either be delivered on-time or omitted, with the probability of omission lower than some pre-specified failure rate. The paper proposes an efficient algorithm for calculating optimal message window sizes, to ensure that statistical guarantees of timely delivery in the presence of errors are provided. The paper is concluded with a short discussion of implementation considerations, followed a simulation study which provides initial supportive evidence for the proposed technique.**

Keywords-Controller Area Network; Time-triggered; Bounded re-transmission; Fault-tolerance.

I. INTRODUCTION

The Controller Area Network (CAN) protocol was originally intended to allow event-triggered communications between unsynchronized nodes in automotive applications [1][2]. Recently, time-triggered communication over CAN has generated a large amount of interest - this is thought to increase the predictability and overall reliability of the network, along with several other benefits (see, for example, [2][3][4]). A potential drawback with most existing time-triggered CAN implementations lies in the enforcement of single-shot message transmissions. Although this effectively bounds worst-case message transmission times, single bit-errors may directly lead to critical message omissions [3][5]. This can be contrasted with the behavior of a regular CAN network, in which unbounded re-transmission attempts are allowed. This native approach effectively ensures delivery of critical messages, at the expense of unbounded message transmission times and hence predictability [3][5]. This can be considered as two extremes of behavior, neither of which is acceptable for most networked real-time applications. To address this problem, this paper proposes a simple technique to act as an extension to existing time-triggered transmission schemes, which effectively provides a bandwidth-efficient compromise between the two extremes. In the proposed scheme, an effective transmission window is defined for each message, within which a bounded amount of re-transmission is allowed following bit or burst errors. The window sizes (as opposed to the underlying frame sizes) can then be used to create the frame schedule, using any existing frame packing algorithm. The remainder of this paper is structured as follows. Section II reviews previous work in the area. Building from this, Section III describes the proposed windowed transmission technique, and describes an algorithm to efficiently calculate the optimal size of a window for a given message. Section IV briefly describes some implementation issues, whilst Section V describes a case-study with promising initial results. The paper is concluded in Section VI.

II. PREVIOUS WORK

A. Time-Triggered CAN Communications

A number of hardware and software-based protocol extensions and modifications have been proposed to enable time-triggered communications on CAN; comprehensive reviews are provided by Short & Pont [3] and Rodriguez-Navas et al. [6]. The described techniques tend to rely on the use of a global clock which, in turn, supports a Time Division Multiple Access (TDMA) message schedule. Key to achieving clock synchronization is the reliable broadcast of time reference messages from a 'time master' node. These reference messages are then generally used with a hardware- or software-based distributed clock synchronization algorithm.

Several software-only synchronization algorithms have been described. When using such techniques, clock synchronization at an accuracy level of 100 µs is typical; however techniques giving accuracies up to 1 µs are known. An example of the latter category is the family of 'shared-clock' algorithms which – at the expense of a small local CPU overhead - provide time-triggered communications without the need for additional hardware, or complicated software clock synchronization algorithms [3][7][8].

From a hardware perspective, the Time-Triggered CAN (TT-CAN) protocol uses a global clock synchronization method to provide time-triggered operation of CAN at the hardware level [9]. Again, the protocol provides a maximum accuracy of +/- 1 μs, and supports a static TDMA schedule which can provide 'empty' slots that allow normal message arbitration for dynamic messages. A full implementation of TT-CAN normally requires dedicated hardware and, at the present time, such hardware has not been widely adopted.

The general goal of all these protocols – whether hardware or software based - is the creation of a collision free (and hence arbitration free) bus access schedule, such as that depicted in Fig. 1 [3]. Due to the finite clock error ε which always exists between any two clocks in the distributed system, a small inter-slot spacing $P \geq 2\varepsilon$ must be employed. In the general case, designing a message schedule to meet a given set of period requirements is strongly NP-Hard, but in practice many fast algorithms (both optimal and heuristic) are known to generate feasible TDMA schedules (see e.g. [10][11]). It is not uncommon to achieve bus utilizations in excess of 90% using such techniques [3].

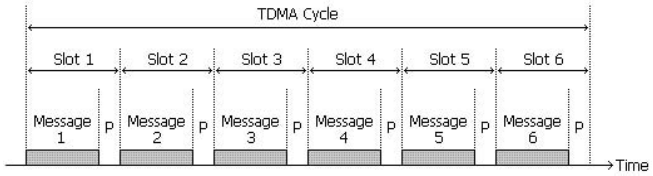

Figure 1. Typical TDMA structure with inter-slot spacing *p*.

B. Fault-Tolerant CAN Communications

When messages are required to be sent over multiple redundant CAN channels to improve reliability, replica determinism and the notion of global time become of great importance [3][6][9]. Replica determinism can be enforced - in part - by the use of single-shot transmissions or upper bounding the latest time that a message may commence transmission. If replica determinism can be enforced, then multiple redundant and fault-tolerant CAN networks may be operated in parallel to increase the reliability of the physical layer [3].

When messages are subject to interference such as electromagnetic disturbances, this tends to manifest itself as random bit errors on the network. In response to any detected errors, under the CAN protocol an error frame is generated - which may have a length of up to 31 bits [12] – followed by a re-transmission attempt. In a real-time system this re-transmission can be very problematic due to deadlines being missed in a 'domino-style' effect; see, for example, [5] for further details. Experimental studies would seem to place the Bit Error Rate (BER) for CAN in the region of 10^{-10} in 'benign' environments, increasing to 10^{-6} in 'aggressive' environments [13]. In some extreme cases, BERs as high as 10^{-3} have been reported for vehicles operating in 'hostile' environments, for example when in close proximity to large electromagnetic radiation sources such as radio transmission stations [14]. In more aggressive environments, it has also been reported that around 90% of these errors occur in short correlated bursts, with durations typically between 5 to 20 bit times [5][13][14]. Although the paper is principally concerned with uncorrelated error arrivals, some initial considerations of these correlated burst errors are also given.

With these points in mind, in a real-time application some form of timeout or upper bound is required to limit the worst-case transmission time of a given frame. In many hard real-time systems, it is arguably better not to receive an instance of a periodic message at all, than to receive the instance late [3][5]. Unfortunately, such timeouts are not provided as a standard CAN feature; at the expense of local CPU overheads, several techniques have been described to enforce this behavior. Previous works such as [3] and [6] have suggested that only single-shot transmissions be attempted in the TDMA round, and it is in fact an enforced requirement in TT-CAN networks [9]. For critical message streams, duplication of message instances is the principal means to achieve the desired reliability. For a message requiring *c* bits to be transmitted in an environment with a BER of β, if *r* message duplicates are sent then the probability of failure reduces with *r* as follows [3]:

$$\lambda = \left(1 - (1-\beta)^c\right)^r \tag{1}$$

However, as will subsequently be shown, in many cases sending full message duplicates provides a bandwidth-inefficient solution to this problem. Since bandwidth is relatively scarce in CAN networks, this can be a major issue. Another interesting scheme – similar to the current work - is the 'Timely CAN' scheme proposed in [5]. The authors propose a reliable technique to upper-bound the latest time that a particular message can be scheduled for re-transmission in native CAN networks under an optimal priority assignment, in order to prevent domino-effect deadline misses. From an estimate of message worst-case response times,[1] message transmission times are subtracted to obtain the required 'timeout' parameters. In the worst-case, single bit errors can still lead to omissions; however if 'slack' is placed in the timeout, then a limited amount of re-transmission can be achieved. The amount of slack to employ can be determined by adding an error model to the response time calculations, where error arrivals are treated as sporadic events, see e.g. [12][15]. In the next Section, an efficient and generic solution applicable to TDMA-operated CAN networks will be proposed.

[1] It should be noted that the original analysis provided in [5] contains an error due to 'push-through' blocking; see [12] for further details.

III. PROPOSED WINDOWED TRANSMISSION SCHEME

The proposed windowed transmission scheme is relatively simple. We make the assumption that a CAN message set is made up of n message streams, each stream f being described by the following four parameters:

$$f_i = (p_i, c_i, d_i, \lambda_i) \tag{2}$$

Where p_i is the message period, c_i is the message (worst-case) transmission time, d_i is the message relative deadline and λ_i is the target failure rate of the message stream, specified as a probability of unsuccessful transmission per unit time. Note that λ_i may be derived from a higher-level safety analysis, e.g. a Safety Integrity Level (SIL).

A. Basic Concept

The proposed technique intends to provide the following multi-criteria real-time/reliability guarantee: if a message is delivered, it is delivered on time; if a message is *not* delivered (omitted), *the failure rate for omission is* $\leq \lambda_i$. The scheme is best illustrated by way of example. Suppose we have a critical message 'X' that requires two duplicate copies to be sent every TDMA cycle. Fig. 2 (top) shows such a situation, where for clarity the duplicated copy is assigned a slot immediately following the original. Suppose that a bit-error occurs in both of these slots, as shown in Fig. 2 (middle) – this will lead to both messages effectively being dropped, and can potentially result in wasted bandwidth as only single-shot transmission is allowed. However, now consider the situation depicted in Fig 2 (bottom). Both slots are merged in a single window of length m, where m is specified in network bit times; (re)transmission of message X is only allowed from the start of the slot, up to the point labeled 'Upper Bound X', i.e. m-c bit times after transmission has first been requested.

As can be seen in the Figure, this has a positive effect on the reliability of the message delivery; even when numerous bit-errors occur, the probability of successful message delivery can be maximized. As the slots have effectively been merged, only a single inter-slot spacing is required. In fact, in most cases the required window size can be reduced by a significant amount over sending duplicated single-shot copies; much better use can be made of the available bandwidth. As the window size is fixed, all the desired properties of a time-triggered communication system are retained. Given a set of slot sizes m_i for each of the messages, the TDMA schedule may be created using appropriate techniques and packing algorithms. However, this raises an important question; namely, for a given message length and bit error probability, how large does the transmission window m have to be to ensure the message will be delivered with the required probability? This question will be addressed in the following two Sections, beginning with computation of optimal window sizes.

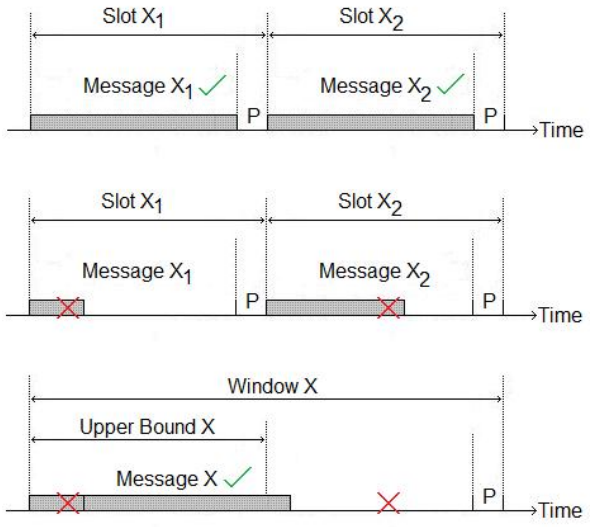

Figure 2. Basic concept of a windowed transmission.

B. Computing The Optimal Window Sizes

This Section will first consider how the probability of a successful transmission can be computed for increasing values of m. We consider the transmission of successive bits on a CAN network in a noisy environment to effectively be a Bernoulli process, with the probability of success given by $(1-\beta)$, and the probability of failure given by β, where β is the BER and $d = 1$. The following Theorem establishes that the probability of successfully transmitting a message in a window of size j can be formulated as a one-dimensional Bernoulli sequence problem.

Theorem 1: If the probability of a bit error is β, then the probability p_j that at least b bits have been consecutively transmitted without error in a sequence of j bits (with $j > b$) can be calculated recursively as follows:

$$p_j = p_{j-1} + (1 - p_{j-b-1}).\beta.(1-\beta)^b \tag{3}$$

Proof: by induction on j.

Base case(s): for $j \leq b$. When the number of bits transmitted is less than b, achieving a consecutive stream of b bits (with or without error) is impossible, thus $p_0 = p_1 = p_2 \ldots = p_{j-1} = 0$. For the case $j = b$, it is a well known statistical property that $p_b = (1-\beta)^b$.

Inductive step: for all $j>b$. We have that p_j must be equal to the summed probabilities of a successful outcome in any one of the previous j-1 trials ($\sum$ term), plus the additional probability that a successful sequence has just been received with the j^{th} bit completing the run of b successes (Δ term). Given the recursive solution, taking the summation of p_{j-1} at

every step gives the required $\sum$ term, since at step j $p_j = (\sum_j + \Delta_j) = (\sum_{j-1} + \Delta_{j-1} + \Delta_j) = (\sum_{j-2} + \Delta_{j-2} + \Delta_{j-1} + \Delta_j) \ldots = (\sum_{i<j} p_i + \Delta_j)$. Considering the Δ term, as shown in Fig. 3 we have for this additional term that the following series of events must occur:

(1) At step j, the last b bits have been received without error; i.e. bits j-b+1 though j inclusively have been successes, with combined probability $(1-\beta)^b$.

(2) The bit received at step j-b must have been received in error to start the sequence (else we have had b+1 or more consecutive successes), with probability β.

(3) Prior to this - at step j-b-1 - no successful run of b consecutive bits had been received, which by definition of p_j is given by the probability $(1-p_{j-b-1})$.

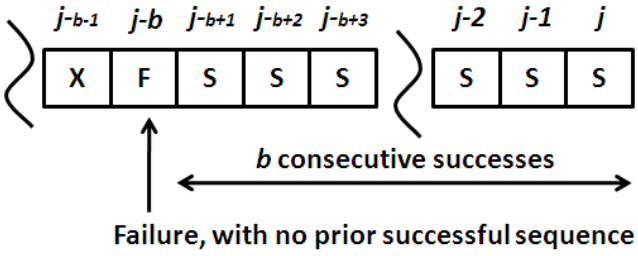

Figure 3. Conditions required for a successful outcome at the j^{th} step.

Thus the Δ term consists of the combined probability of these events occurring, i.e: $\Delta_j = (1-p_{j-b-1}).\beta.(1-\beta)^b$. Thus we have that $p_j = (\sum_j + \Delta_j) = p_{j-1} + (1-p_{j-b-1}).\beta.(1-\beta)^b$, and the Theorem is proved. □

From this result, a simple algorithm may be derived to determine the optimal window size for a particular message. This is the subject of the next Section.

C. Optimal Window Sizing Algorithm

A relatively simple algorithm to determine the smallest value of m (window size) such that a message requiring b bits can be transmitted with a failure probability $\leq \lambda$, in an environment having a BER of β. Such an algorithm is given in Fig. 4. According to the result of Theorem 1, it is clear that the loop between lines 08 and 11 will not terminate until the first value of m such that $(1.0 - p_m) \leq \lambda$ has been found, or m exceeds the message relative deadline d which is supplied as an input parameter and expressed in network bits.

D. Algorithm Analysis

Given that p_j is a monotone increasing function of j (due to the $\sum$ term combined with the Δ term, which is always > 0 for input values $\lambda \in (0, 1)$ and $\beta \in (0, 1)$), the algorithm is guaranteed to converge. However, for small λ and large β, convergence may take some time; given the real-valued representation of these parameters, it is very difficult to provide a (worst-case) run-time bound on the time complexity of the algorithm, when the input is expressed in bits. In a real-time application, however, it can be observed that the message can never meet its deadline (with probability less than λ) if its window size exceeds the message deadline. As such, the additional termination condition applied at line 08 ($m < d$) bounds m such that the algorithm terminates when $m = d$, where d is the relative deadline of the message, expressed in bit times. Note that an appropriate error message can be generated at this point. As with response time analysis, this effectively bounds the time complexity to be pseudo-polynomial in the task parameters. For n message streams, optimal window sizes may therefore be computed with complexity $O(nd_{max})$.

```
01 Proc Optimal_Window(b, λ, β, d)
02 {
03 FOR i:= 0 TO b-1 DO:
04     p_i = 0;
05 END FOR
06 p_b := (1-β)^b;
07 m := b;
08 WHILE ((1.0 - p_m) > λ) AND (m < d) DO:
09     m := m+1;
10     p_m := p_m-1 + ((1-p_m-b-1).β.(1-β)^b);
11 END WHILE
12 RETURN(m);
13 }
```

Figure 4. Algorithm 1: computing the optimal window size m.

In terms of the required memory storage space, it can be noted that only the last (b+1) values are required to be stored for the recursion. Therefore an array of size b+1 can be used to limit the required memory storage of the algorithm; this array may be indexed m *mod*(b+1) to effectively overwrite old calculations. Since b is upper-bounded by the very nature of the CAN protocol – a message can only carry up to 8 bytes - the memory storage requirements have complexity $O(1)$.

E. Illustrative Examples

Example 1: Suppose we have a message with $b = 5$ bits required to be transmitted, with a failure probability $\lambda = 0.1$, in an environment having a BER of $\beta = 0.2$. Running the algorithm described above on this example yields the values as tabulated for p_m in Table I, terminating when the target failure rate is achieved with $m = 19$ and the probability of success $p_{19} = 0.91$. By contrast, if we must allocate only duplicate copies of the 5-bit message, then according to equation (1), 6 copies (= 30 bits) would be required; the windowed approach leads to a saving of 36.7% in terms of the number of bits requiring transmission.

Example 2: Suppose we have a critical CAN message using 11-bit identifiers, requiring 8 bytes of data. Thus b = 166 bits are required to be transmitted (135 bits in the main message plus 31 bits for a worst-case error frame [12]), with a period p = 100 ms. The safety integrity level of the system

TABLE I. COMPUTING THE OPTIMAL WINDOW SIZE FOR EXAMPLE 1

m	p_m	M	p_m	m	p_m	m	p_m
0	0.00	5	0.33	10	0.66	15	0.83
1	0.00	6	0.39	11	0.70	16	0.86
2	0.00	7	0.46	12	0.74	17	0.88
3	0.00	8	0.52	13	0.78	18	0.89
4	0.00	9	0.59	14	0.81	19	0.91

specifies a target failure rate of $\lambda = 1.0 \times 10^{-9}$ failures/hour, in an environment with a BER of $\beta = 2.6 \times 10^{-7}$. Since 36000 messages are to be sent every hour, the probability that a single instance of this critical message is not delivered should be $\leq 2.78 \times 10^{-14}$. Running the algorithm described above on this example reveals that the target failure rate will be achieved with $m = 489$, and the probability of failure with this size window is $p_{489} = 1.34 \times 10^{-14}$. By contrast, if we must allocate only duplicate copies of the 166-bit message, then according to equation (1), 4 copies (= 664 bits) would require transmission. In this case the windowed approach leads to a saving of 26.4% in terms of the number of bits requiring transmission.

F. Bursty Links

The analysis given in the Sections above considers only the case of uncorrelated error behaviors. In order to begin to consider the effects of burst errors, a basic approach would be to consider β to be the burst error rate, with each burst having duration of exactly u bits. In this case, bursts can be considered by adapting the delta term as follows: $\Delta_j = (1-p_{j-b-u}).\beta.(1-\beta)^b$, with the error in j-b^{th} bit is considered to be the *last* in a sequence of u consecutive errors (taking the convention that $p_j = 0$ for $j \leq 0$). With $D = 1$, the bit and burst error analysis become identical. Employment of more detailed error models (e.g. based on Markov chains [16]) which are known to accurately model burst behaviors, are considered an area of future work. The next Section describes some practical issues related to the implementation of the proposed transmission scheme.

IV. IMPLEMENTATION ISSUES

A. Software Based Solutions

Although some modern CAN controllers now have direct hardware support for single-shot message transmissions, this will not directly extend to the current scheme. In order to implement message timeouts, one possible solution would be as follows. As has been argued in [3], in time-triggered systems (under fault-free conditions), when a message is scheduled for transmission by the host CPU the bus should *already* be in the idle state. To enforce a timeout on message i, the host can simply set a (high-priority) timer interrupt to occur m_i time units into the future, just prior to setting the transmit request (TXRQ) flag in the CAN controller. When this interrupt subsequently occurs, the host immediately resets the TXRQ flag in the controller. The main advantage of this solution is that it is relatively simple. However it has several drawbacks; any jitter and latency affecting the servicing of the ISR – coupled with clock drift between the host timer and the CAN bus oscillator - may skew the actual (real) time the transmission is cancelled. It also requires the use of a timer with at least as good a precision as a CAN network bit time, may also place a relatively large load on the CPU if there are numerous messages to transmit.

B. Hardware Based Solutions

Modifications to the CAN protocol at the silicon level have traditionally been impractical. However the advent of programmable logic devices such as FPGA's and the maturation of hardware description languages such as VHDL have changed this situation somewhat. In particular, the authors of the current paper have previously developed a fully CAN-conformant soft-core protocol controller [17]. The advantage of such an 'open' hardware solution is that various extensions to (or modifications of) the CAN protocol can be implemented and investigated with relative ease (e.g. [18]). In the context of the current work, we have proposed and implemented the following small but powerful modification to CAN.

In addition to allowing each CAN object in the controller to be operated in 'standard' or 'single-shot' mode, a third mode of operation – 'window' mode - was introduced. In this mode, a 32-bit hardware counter C (which, when active, is incremented by 1 with every bit time on the bus) and two 32-bit match registers (CLB and CUB) were added to each CAN object. In addition, the host CPU sees an 'effective' TXRQ bit, but this is in fact a dummy, used only for interface purposes; a 'hidden' register TXRQ# is employed to control the real transmission logic. When a message transmission is initiated by the host (setting TXRQ), the counter C is reset to 0; setting of the TXRQ# bit of the CAN object is delayed until C = CLB. Also, when C = CUB, both the TXRQ and TXRQ# bits are automatically re-set in hardware. Since C is incremented at the same rate as the bit-time, this provides for a programmable 'allowed transmission window' for a given CAN object which does not require the need for further CPU intervention. Additionally, the impact of jitter and latency on the host CPU is minimized, as the timer is referenced to the local oscillator. The only potential drawback of this solution is that it requires a very small increase in hardware complexity. However, since this modification not only allows the effective implementation of the proposed windowed transmission, but would also allow the implementation of alternate (similar) protocols such as the Timely CAN [5] and shared-clock [8] protocols, this very simple hardware change could easily be incorporated into future generations of CAN controllers.

V. SIMULATION STUDY

In order to begin to investigate the proposed technique, a small simulation study was carried out. This study was carried out to assess the potential bandwidth savings that may be achieved by employing the windowed technique as the message parameters are varied. Three experiments were carried out. In each experiment, 100,000 message streams were generated with parameters randomly selected (using a uniform distribution) from the following intervals: $p_i \in [1, 1000]$ ms, $DLC_i \in [0, 8]$ bytes, $\lambda_i \in [10^{-9}, 10^{-5}]$ failures/hour. Standard and extended identifiers were randomly employed; a 31-bit worst-case error frame was also added to the length of each message. The generated target failure rates cover all four SILs as specified in IEC 61508, and individual message failure rates were derived from the target λ's based on their periods. Three different classes of BER were employed; Benign/Normal with $\beta \in [10^{-9}, 10^{-7}]$, Normal/Aggressive with $\beta \in [10^{-7}, 10^{-5}]$ and Aggressive/Hostile with $\beta \in [10^{-5}, 10^{-3}]$. In each case the percentage reduction in the number of bits requiring transmission was calculated when employing windowed transmission and message duplicate transmission. The average and maximum recorded values for each environment are as shown in Table II. Also shown is the average and maximum recorded window sizes m, which also gives an indication of the quick convergence of the slot sizing algorithm, even for low failure rates and high BER's.

TABLE II. REDUCTIONS IN BITS REQUIRED FOR TRANSMISSION

	Percent Reduction		Window Size	
Environment	**Average**	**Maximum**	**Average**	**Maximum**
Benign / Normal	2.7	33.3	280	571
Normal / Aggressive	12.0	33.2	410	797
Aggressive / Hostile	29.1	39.1	810	1883

From this Table, it can be seen that the average effectiveness of the proposed technique depends upon the target environment; the worse the expected BER, the higher the average reduction in required bandwidth. For hostile environments, an average 29% reduction can be achieved; given the scarcity of available bandwidth with CAN, this is a potentially large saving. In normal and benign environments, the average reductions drop to 12% and 2.7% respectively; however, the best case reductions remain at around 33%. In each case, the worst-case reductions were 0%, i.e. the technique performed no worse than sending full duplicates.

VI. CONCLUSIONS AND FURTHER WORK

This paper has considered a simple windowed transmission technique for use with CAN. An algorithm to calculate the smallest slot sizes such that messages can be sent with a pre-specified success probability has been presented. In comparison to sending full message duplicates, simulation studies indicate that the technique can – in some cases - significantly reduce the required bandwidth, whilst maintaining reliability and timeliness. Future work will extend the proposed methodology into a practical network realization; some promising initial results from such a practical implementation are described in [19].

REFERENCES

[1] R. Bosch, "CAN Specification 2.0", Postfach, Stuttgart, Germany: Robert Bosch GmbH, 1991.

[2] H. Kopetz, "A Comparison of CAN and TTP", *Annual Reviews in Control*, Vol. 24, pp. 177–188, 2000.

[3] M. Short and M.J. Pont, "Fault-Tolerant Time-Triggered Communication Using CAN", *IEEE Transactions on Industrial Informatics*, Vol. 3, No. 2, 2007.

[4] M. Short, M.J. Pont and J. Fang, "Assessment of performance and dependability in embedded control systems: methodology and case study", *Control Engineering Practice*, Vol. 16, pp. 1293–1307, 2008.

[5] I. Broster and A. Burns, "Timely use of the CAN protocol in critical hard real-time systems with faults", In Proceedings of the *13th Euromicro Conference on Real-time Systems (ECRTS)*, Delft, The Netherlands, June 2001.

[6] G. Rodriguez-Navas, S. Roca, and J. Proenza, "Orthogonal, fault-tolerant and high-precision clock synchronization for the Controller Area Network", *IEEE Transactions on Industrial Informatics*, Vol. 4, No. 2, pp. 92–101, May 2008.

[7] M.J. Pont. *Patterns for time-triggered embedded systems*. Addison-Wesley, 2001.

[8] D. Ayavoo, M.J. Pont, M. Short, and S. Parker, "Two novel shared-clock scheduling algorithms for use with CAN-based distributed systems", *Microprocessors and Microsystems*, Vol. 31, No. 5, pp. 326-334, 2007.

[9] G. Leen and D. Heffernan, "TTCAN: a new time-triggered controller area network", *Microprocessors and Microsystems*, Vol. 26, No. 2, pp. 77-94, 2002.

[10] R. Saket and N. Navet, "Frame Packing Algorithms for Automotive Applications", *Journal of Embedded Computing*, Vol. 2, No. 1, pp 93-102, 2006.

[11] Z. Hanzálek, P. Burget and P. Šůcha, "Profinet IO IRT Message Scheduling", In: *Proceedings of the 21st Euromicro Conference on Real-Time Systems*, Dublin. July, 2009, pp. 57-65.

[12] I. Davies, A. Burns, R. Brill, and J. Lukkien, "Controller Area Network (CAN) schedulability analysis: refuted, revisited and revised", *Real-Time Systems*, Vol. 35, No. 3, pp. 239–272, 2007.

[13] J. Ferreira, A. Oliveira, P. Fonseca and J.A. Fonseca, "An Experiment to Assess Bit Error Rate in CAN", In *Proceedings of the 3rd International Workshop on Real-Time Networks*, June 2004.

[14] B. Gaujal and N. Navet, "Fault Confinement Mechanisms on CAN: Analysis and Improvements", *IEEE Transactions on Vehicular Technology*, Vol. 54, No. 3, pp. 1103-1113, 2005.

[15] S. Punnekkat, H. Hansson and C. Norstrom, "Response time analysis under errors for CAN", In: *Proceedings of the 6th Real-Time Technology and Applications Symposium*, IEEE Computer Society Press, pp 258–265, 2000.

[16] E.N. Gilbert, "Capacity of a Burst-Noise Channel", *Bell Systems Technical Journal*, Vol. 39, pp. 1253–1265, 1960.

[17] I. Sheikh, and M. Short, "A low-cost and flexible approach to CAN conformance testing". In: *Proceedings of the 6th International Conference on Informatics in Control, Robotics and Automation (ICINCO 2009)*, Milan, Italy, pp. 97-104, July 2009.

[18] I. Sheikh, M. Short and M. Hanif, "Improving Information Throughput and Transmission Predictability in Controller Area Networks", To appear in: *Proceedings of the IEEE International Symposium on Industrial Electronics*, Bari, Italy, July 2010.

[19] I. Sheikh, M. Short and A. Imran, "Fault Injection Analysis of a Windowed Transmission Scheme for Controller Area Networks", Paper Submitted to the 49th Allerton Conference, May 2010.

Schedulability Analysis for Multi-level Hierarchical Server Composition in Ethernet Switches

Rui Santos, Paulo Pedreiras,
IEETA / University of Aveiro
Aveiro, Portugal
{rsantos,pbrp}@ua.pt

Moris Behnam, Thomas Nolte
MRTC / Mälardalen University
Västerås, Sweden
{moris.behnam,thomas.nolte}@mdh.se

Luis Almeida
University of Porto
Porto, Portugal
lda@fe.up.pt

Abstract

The FTT-enabled (Flexible Time-Triggered) Ethernet Switch provides flow-based dynamic scheduling that allows to handle bursty traffic in a bandwidth efficient way. For that, this switch uses adaptive resource-reservation, associating servers to flows or groups of flows. This way, flows have a guaranteed, but bounded, access to the communication resources. These servers can take up a compositional multi-level hierarchy and they can be adapted on-line to make a better use of the available bandwidth. To assure a continued real-time behavior, the FTT-enabled Ethernet Switch integrates an admission control mechanism, which screens all adaptation and/or reconfiguration requests. Whenever such requests may compromise the flow timeliness or exceed the available memory, they are rejected. This paper focuses on the flow timeliness verification, only, providing a response-time based schedulability analysis that permits assessing the schedulability of a hierarchical composition of servers and flows. [1]

1 Introduction

Many current embedded applications are complex entities structured in components and usually assuming a hierarchical composition. However, these applications can suffer limitations in accessing resources due to lack of frameworks that provide adequate resource access control in such complex systems. For example, in the network domain, integrating different applications with different communication requirements under real-time constraints can generate problems of resource allocation (bandwidth) and temporal isolation between streams or across applications. One recent paradigm that favors the development of frameworks to support hierarchical structures is component-based design in which applications are built by composing diverse components developed separately. The benefits range from reduction of design complexity to more efficient resource sharing, satisfying individual service requirements of each component and enforcing mutual temporal isolation.

Architectures based on servers that act as containers in the temporal domain have been recognized as an effective means to enable such kind of resource sharing [1] and they can be the basis for resource partioning and virtualization, supporting the separation between the applications and the hardware platform on which they will execute. Following this trend, the FTT-enabled (Flexible Time-Triggered) Ethernet Switch [2] provides a framework to carry out hierarchical composition of servers that divides the network resource in an efficient way and allows an easy and natural mapping of the applications onto the network. Moreover, the use of servers for flow management allows handling heterogeneous kinds of traffic with arbitrary arrival patterns and with temporal isolation.

This paper presents an extension of previous work in this framework, particularly that reported in [3], which defined an adaptation and reconfiguration protocol that allowed adding, removing and modifying servers and the associated asynchronous flows. An on-line admission control using a light utilization-based schedulability analysis enforced continued timeliness even during changes. The traffic scheduling followed the *blocking-free non-preemptive model*, which applies when the traffic is scheduled in cycles, within partitions, and before the start of the respective partition (or window), adhering strictly to the partition duration. This implies one aspect, there is an extra delay since an arriving packet might have to wait up to one cycle to be considered by the scheduler for possible transmission. An alternative that improves the latency of the switch is to enable the scheduling during the partition, executing it whenever a packet arrives. In this case, the previous analysis does not

[1]This work was partially supported by the iLAND project, call 2008-1 of the EU ARTEMIS JU Programme, by the European Community through the ICT NoE 214373 ArtistDesign and by the Portuguese Government through the FCT project HaRTES - PTDC/EEA-ACR/73307/2006 and Ph.D. grant - SFRH/BD/32814/2006.

hold and the blocking caused by the transmission of non-preemptive packets of lower priority servers must be taken into account. Therefore, this paper presents a schedulability analysis based on response time for a multi-level hierarchical server composition that handles the asynchronous flows within the FTT framework and considers the blocking referred above.

The remainder of the paper is organized as follows. Section 2 presents an overview of schedulability analysis for hierarchical scheduling frameworks. Section 3 introduces the basics of the FTT-enabled Ethernet Switch and describes its integration with the server-based traffic scheduling. Section 4 presents the schedulability analysis and an algorithm for determining the response time. Finally, Section 5 concludes the paper and addresses future work.

2 Related work

The use of servers in networking is common, being the *leaky bucket* the most well-known. The leaky-bucket is, in fact, part of a general servers category called *traffic shapers* [4], which have the purpose of limiting the amount of traffic that a node can submit to the network within a given time window, bounding the node burstiness. These servers use techniques similar to those used by CPU servers, based on capacity that is eventually replenished.

Particularly regarding Real-Time Ethernet (RTE) protocols, some very limited forms of server-based traffic handling can also be found. Some protocols enforce periodic communication cycles with reserved windows for different traffic classes (e.g. PROFINET-IRT [5] and Ethernet Powerlink [6]). This is a trivial composition of several PS that results in an inefficient use of the network bandwidth. Other protocols, such as [4], implement traffic shapers in the end nodes that behave similarly to a DS. However, due to infrastructural limitations, none of these protocols supports arbitrary server policies nor their hierarchical composition and dynamic adaptation or creation/removal.

Another related area, despite typically considering preemptive task scheduling, is that of general hierarchical scheduling frameworks (HFS). Deng and Liu [7] began proposing two levels HFS for open systems, where subsystems may be developed and validated independently. Kuo and Li [8] introduce for such two levels a schedulability analysis based on Fixed Priority Scheduling (FPS) with a global scheduler. Shin and Lee [9] present a generic scheduling interface model in order to construct hierarchical scheduling frameworks. Almeida and Pedreiras [10] present a response time analysis for the periodic server model and address the problem of designing a server to fulfill the application constraints. Arvind *et al.* [11] generalize the periodic resource model for compositional analysis of hierarchical scheduling frameworks.

Finally, another related area is that of synchronization protocols in HFS. For example, SIRAP [12] addresses CPU resource sharing among several subsystems that execute within servers and it proposes inserting idle-time (iit) whenever the remaining capacity is not enough to execute an access to a shared resource. However, only two level HFS are addressed, while in this paper we seek explicitly the support to multi-level HFS.

3 FTT-enabled Ethernet Switch

The FTT-enabled Ethernet Switch was created in the scope of FTT paradigm [13]. The FTT paradigm is a master multi-slave communication protocol, where a Master node coordinates the transmissions of other nodes (Slaves) by means of the periodic transmission of a Trigger Message (TM) that contains the schedule for a fixed-duration time slot, designated Elementary Cycle (EC) (Figure 1). The communication is organized in an infinite succession of such Elementary Cycles (ECs).

The FTT framework defines three traffic classes: 1) periodic real-time messages triggered by the master (referred to as *synchronous* since their transmission is synchronized with the master traffic scheduler); 2) aperiodic or sporadic real-time traffic, autonomously triggered by the nodes; and 3) non real-time traffic (classes 2 and 3 are referred to as *asynchronous*). The EC is organized in two windows, synchronous and asynchronous, which convey the corresponding traffic classes.

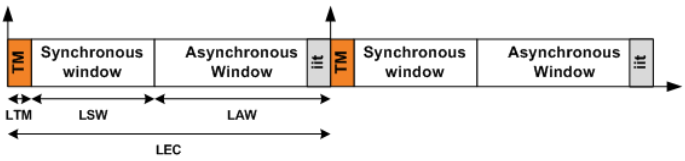

Figure 1: Elementary Cycles

At the begining of each EC the switch broadcasts the TM to all slave nodes, identifying which messages should be transmitted. Synchronous messages are always polled by the TM. In the particular case of the FTT-Enabled Ethernet Switch, no polling for the asynchronous traffic is necessary since the switch is aware of the EC structure (Figure 2) and has a complete control of the message forwarding procedure. Therefore, the asynchronous messages may be sent by the respective sources at arbitrary instants since the switch is able to queue them in dedicated memory pools and transmit them to the respective destinations only during the asynchronous windows. Appropriate scheduling mechanisms, e.g. servers, may be used to schedule the queued asynchronous messages.

The autonomous confinement of messages by the FTT-enabled Ethernet Switch is one of the distinctive features

of this protocol with respect to its predecessors, namely the FTT-SE protocol. This latter protocol relies on Commercial Of-The-Shelf (COTS) Ethernet switches and thus all nodes have to comply with the protocol, i.e., have to integrate a specific device driver, to ensure that the message transmissions occur only at adequate time instants. This is an important limitation since legacy nodes cannot be part of the network. Additionally, the management of the asynchronous traffic is less efficient since this type of traffic has also to be scheduled by the master node and an explicit signaling mechanism by nodes informing the master about the ready asynchronous traffic is required. Finally, the tight control of the message forwarding combined with the awareness of the message requirements also allows the switch to detect failures in the time domain, such as nodes that transmit asynchronous messages at higher rates than the ones declares or that send synchronous messages that where not scheduled by the Master node, preventing its transmission.

Summing up, the FTT-enabled Ethernet Switch provides the following features:

1. online admission control, dynamic QoS management and arbitrary traffic scheduling policies;
2. high system integrity with unauthorized real-time messages being eliminated at the switch input ports;
3. asynchronous traffic autonomously triggered by the nodes, with arbitrary arrival patterns;
4. high configurability: fully synchronous mode, adjustable mixed synchronous/asynchronous mode and fully asynchronous mode;
5. a standard node can take advantage of the real-time services simply negotiating with the switch the creation of a server, i.e., a virtual channel (the negotiation can even be done by a third party node);
6. a standard node can readily transmit non-real-time traffic using a background server thus not interfering with the real-time traffic.

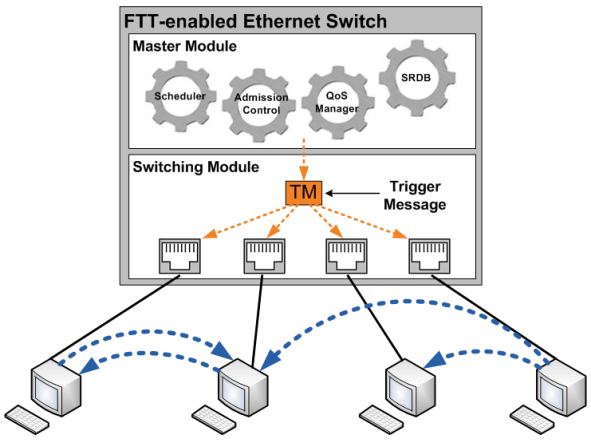

Figure 2: FTT-enabled Ethernet Switch.

3.1 FTT EC structure as composition of servers

As mentioned above, in the FTT-enabled Ethernet Switch the traffic is divided in synchronous and asynchronous classes, associated with disjoint windows that fill in the usable part of the EC. These windows appear once in each EC (Figure 3) and have a bounded size (LSW and LAW, respectively). Note that LSW correspond to an upper bound (the synchronous traffic may use up to LSW in each EC) while LAW refers to a lower bound (asynchronous traffic can use at least LAW in each EC), since the asynchronous window reclaims the bandwidth not used by the synchronous one. Using a server terminology, the synchronous window is associated with a server characterized by a period $T_{SW} = T_{EC}$ and a (maximum) capacity $C_{SW} = LSW$, while the asynchronous window is associated with a server with a (minimum guaranteed) capacity of $C_{AW} = LAW$ and a period $T_{AW} = T_{EC}$. Note that LEC, LSW and LAW are FTT configuration parameters that can be tuned to suit the global application needs.

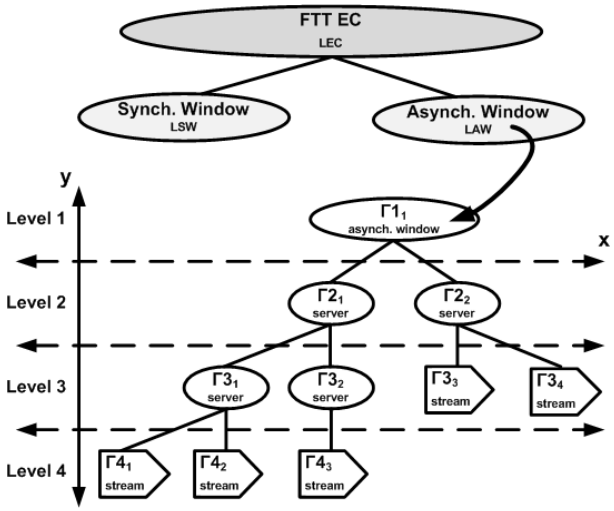

Figure 3: Server Hierarchy.

3.2 Hierarchical Server Composition in the scope of the FTT-enabled Ethernet Switch

In recent work [14] [13], it was proposed to integrate hierarchical server composition on the FTT framework to manage the asynchronous traffic. As illustrated in Figure 3, asynchronous message streams (or streams, for short) are handled by servers. On its hand servers may also depend on other servers. Each server should have enough resources to handle its childs, should they be streams or other servers. Servers and streams are abstracted by components. At each level a component Γ_{y_x} is identified by both index y and x. The index y identifies the level in the hierarchy and the index x identifies the component inside that level. This way,

$y = 1, .., NL$ and $x = 1, .., NC_y$, where NL is the maximum number of levels in the hierarchy and NC_y is the maximum number of components in the level y.

The underlying FTT framework puts some important constraints on the server operation that affect the system schedulability, namely: 1) Ethernet does not permit packet preemption thus preemption is not allowed. Consequently, packets of high priority components may be blocked by ongoing transmissions of lower priority ones. 2) Overruns are not allowed by design, since the capacity is strictly enforced (the switch does not initiate a message transmission that does not fit in the remaining capacity). The combination of 1) and 2) results in a potential appearance of idle time at the end of each server instance, whenever the remaining capacity is not enough to transmit the following queued packet. Despite negative from the schedulability point of view, this *modus operandi* enforces a strict temporal isolation between all components all the way through the top of the hierarchy. Thus, ECs are completely regular and the TM does not suffer any interference from packets managed by server components inside the asynchronous window.

4 Schedulability analysis

Assuring a continued real-time behavior requires the execution of an admission control procedure every time the message set is changed. In the basis of this admission control procedure there is a schedulabilty test. In [3] it is presented an on-line admission control using a light utilization-based schedulability analysis. However, that analysis is based on the *blocking-free non-preemptive model*, which requires that the traffic has to be scheduled in cyclic fashion, within partitions, and before the start of the respective partition (or window). This paper removes such dependency, supporting an unrestricted activation model, i.e. streams enter the scheduling process immediately after being received by the switch. Although more complex, this operation mode reduces the stream forwarding latency, which is an important merit factor in many application scenarios.

4.1 Traffic and resource model

The asynchronous streams are at the end of the hierarchy (Figure 3) and they are characterized in (1) through the sporadic real-time model, where C_{y_x} is the maximum transmission time and $Tmit_{y_x}$ represents the respective minimum interarrival time. $Mmax_{y_x}$ and $Mmin_{y_x}$ is the transmission time of the largest and smallest packet, respectively, transmitted by this stream. P_{y_x} identifies the parent server, i.e, the server to which the stream is connected to and RT_{y_x} its computed response time.

$$AS_{y_x} = (C_{y_x}, Tmit_{y_x}, Mmax_{y_x}, Mmin_{y_x}, P_{y_x}, RT_{y_x}) \quad (1)$$

On the other hand, inside the asynchronous window the servers (components) assume a hierarchical composition with multi-level, forming several branches. The individual server Srv_{y_x} (2) is characterized by its capacity C_{y_x}, its replenishment period T_{y_x}, and a few figures extracted from the set of children components, either servers or streams, namely the maximum and minimum packet transmission times $Mmax_{y_x}$ and $Mmin_{y_x}$ respectively. Moreover, the Srv_{y_x} is characterized by a parent server P_{y_x} and its computed response time RT_{y_x}. Note that despite the similarity between the characterization of servers and streams, there is a fundamental difference since the stream implies actual transmission time that uses the capacity of the respective server.

$$Srv_{y_x} = (C_{y_x}, T_{y_x}, Mmax_{y_x}, Mmin_{y_x}, P_{y_x}, RT_{y_x}) \quad (2)$$

4.2 Schedulability algorithm

As referred before, the servers capacities are strictly enforced and overruns, e.g., caused by a non-preemptive packet transmission that extends beyond the exhaustion of the respective server capacity, are not allowed. This is avoided by inserting idle-time (iit), called *self-blocking* in the scope of SIRAP [12], whenever the remaining server capacity is not enough for the transmission of a full packet. This way, the remaining capacity is wasted and the pending transmission is delayed for successive server instances when enough capacity is available (Figure 4). Therefore, the maximum inserted idle-time (iit) that a server component Γ_{y_x} can suffer is equal to the maximum packet transmission time managed by this server ($Mmax_{y_x}$). On the other hand, $Mmax_{y_x}$ also allows knowing which is the maximum blocking caused by the respective component Γ_{y_x} to the higher priority components in the same branch and in the same level. Moreover, the $Mmin_{y_x}$ is used to know the maximum memory required in each branch, but this subject is out of the scope in this paper. This way, before performing any change to the message set it is necessary to assess its impact in the parameters $Mmax$ and $Mmin$ along the hierarchy. Therefore, the schedulability algorithm presented in this section is executed in two phases.

4.2.1 Schedulability algorithm - first phase

The first phase of the algorithm simulates the requested change in the hierarchy and by going from the bottom to the top aims to find the $Mmax$ and $Mmin$ packet transmission time in each branch. This means, for instance, at the end of this phase, the component that manages the asynchronous window Γ_{1_1} will have the maximum ($Mmax$) and

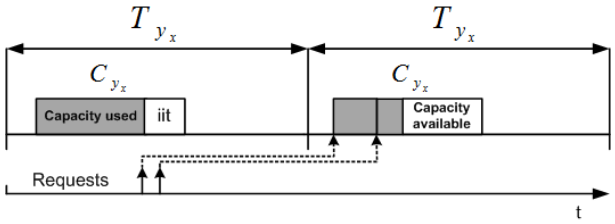

Figure 4: Inserting idle-time (iit) to enforce servers capacities

the minimum ($Mmin$) packet transmission time among all the asynchronous streams transmitted in that window.

Example. As an example, consider the components shown in Figure 3. Assume that the streams have the following $Mmax$ and $Mmin$, respectively: $\Gamma_{3_3}(120000, 8000)$, $\Gamma_{3_4}(121000, 8000)$, $\Gamma_{4_1}(117000, 8000)$, $\Gamma_{4_2}(118000, 8000)$, $\Gamma_{4_3}(119000, 8000)$. Given such scenario, after the first phase of the schedulability algorithm, the servers (components) in the hierarchy inherit the maximum $Mmax$ and the minimum $Mmin$ of their children thus resulting in $\Gamma_{3_1}(118000, 8000)$, $\Gamma_{3_2}(119000, 8000)$, $\Gamma_{2_1}(119000, 8000)$, $\Gamma_{2_2}(121000, 8000)$, and finally $\Gamma_{1_1}(121000, 8000)$.

4.2.2 Schedulability algorithm - second phase

The second phase consists in verifying, from the top to the bottom of the hierarchy, the schedulability of each component and consequently the schedulability of the whole system. For this purpose, a local schedulability analysis under FPS, presented in [9], is used:

$$\forall \Gamma_{y_x} \exists t : 0 < t \leq T_{y_x}, \mathbf{rbf}_{y_x}(t) \leq \mathbf{sbf}_{P_{y_x}}(t), \quad y = 2..NL \text{ and } x = 1..NC_y, \tag{3}$$

where, $\mathbf{rbf}_{y_x}(t)$ denotes the request bound function of the component Γ_{y_x} that, for each instant t, quantifies the maximum load submitted to the parent component P_{y_x} by the component itself together with interference of its high priority components and also together with blocking of low priority components. On the other hand, $\mathbf{sbf}_{P_{y_x}}(t)$ is the supply bound function associated to the parent component of Γ_{y_x} that computes the minimum bandwidth supply provided to its children, in each instant t. Consequently, the worst case response time of a component Γ_{y_x} is given by the first intersection between the $\mathbf{rbf}_{y_x}$ and $\mathbf{sbf}_{y_x}$, and it is described as follows:

$$RT_{y_x} = \text{ shortest } t^* : \mathbf{rbf}_{y_x}(t^*) = \mathbf{sbf}_{P_{y_x}}(t^*) \tag{4}$$

After a suitable schedulability analysis, the requested change is introduced in the hierarchical structure and the simulated configuration performed in the first phase takes effect. On the other hand, if the schedulability analysis fails the old configuration remains unchanged.

Supply bound function. In order to define the $\mathbf{sbf}_{y_x}(t)$, we use the *Explicit Deadline Periodic* (EDP) resource model [11] that generalizes the periodic resource model for compositional analysis of hierarchical scheduling frameworks. An EDP resource model is given by $\Omega = (\Pi, \Theta, \Delta)$, where Θ is the units of the resource within Δ time units (deadline) and with period Π of repetition. This way, mapping to our framework, a component is defined as $\Gamma_{y_x} = (\Pi_{y_x}, \Theta_{y_x}, \Delta_{y_x}) = (T_{y_x}, C_{y_x}, RT_{P_{y_x}})$, where the $RT_{P_{y_x}}$ is the response time of the parent component. However, for the first component in the hierarchy (Γ_{1_1}), the asynchronous window, we consider the Δ equal to the capacity of the window (C_{1_1}), resulting that $\Gamma_{1_1} = (\Pi_{1_1}, \Theta_{1_1}, \Delta_{1_1}) = (T_{1_1}, C_{1_1}, C_{1_1})$. Therefore, according to the EDP model, and assuming the same notation, the $\text{sbf}_{y_x}(t)$ is defined as follows:

$$sbf_{\Gamma_{y_x}}(t) = \begin{cases} b\Theta y_x + \max\{0, t - a - b\Pi y_x\}, \\ \qquad t \geq \Delta_{y_x} - \Theta_{y_x} \\ 0, \quad otherwise \end{cases} \tag{5}$$

where $a = (\Pi_{y_x} + \Delta_{y_x} - 2\Theta_{y_x})$ and $b = \lfloor (t - (\Delta_{y_x} - \Theta_{y_x}))/\Pi_{y_x} \rfloor$. Moreover, $\Pi_{y_x} = T_{y_x}$, $\Theta_{y_x} = C_{y_x}$, $\Delta_{y_x} = C_{y_x}$ when $y = 1$ or $\Delta_{y_x} = RT_{P_{y_x}}$ when $y > 1$.

Request bound function. The $\mathbf{rbf}_{y_x}(t)$ of a component Γ_{y_x} is given by the following equation, similarly to the analysis of SIRAP [12]:

$$\mathbf{rbf}_{y_x}(t) = C_{y_x} + IS_{P_{y_x}}(t) + IH_{y_x}(t) + IL_{y_x}, \tag{6}$$

$$IS_{P_{y_x}}(t) = \left\lceil \frac{t + Mmax_{P_{y_x}}}{T_{P_{y_x}}} \right\rceil \times Mmax_{P_{y_x}}, \tag{7}$$

$$IH_{y_x}(t) = \sum_{\Gamma_{y_j} \in hp(y_x)} \left\lceil \frac{t}{T_{y_j}} \right\rceil \times C_{y_j} \tag{8}$$

$$IL_{y_x} = \max_{\Gamma_{y_j} \in lp(y_x)} Mmax_{y_j}, \tag{9}$$

where $IS_{P_{y_x}}(t)$ is the maximum inserted idle-time (self-blocking) from the parent component and it is modeled by a virtual component of high priority with a period $T_{P_{y_x}}$, a capacity equal to $Xmax_{P_{y_x}}$ and a phase equal to $Xmax_{P_{y_x}}$. $IH_{y_x}(t)$ is the interference from the components with higher priority in the same level and in the same branch, IL_{y_x} is the blocking from components with lower priority.

Despite the similarities with the analysis in [12], this new approach introduces the impact of our multi-level hierarchi-

cal framework in which the impact of the inserted idle-time in the child components is accounted for in $IS(t)$.

4.3 Computing the response time

The response time of each component Γ_{y_x} (with $y > 1$) can be obtained solving iteratively equation (4), and making use of the inverse of the supply bound function as follows:

$$RT_{y_x} = \text{ earliest } t^* : t^* = \mathbf{sbf}^{inv}_{P_{y_x}}(\mathbf{rbf}_{y_x}(t^*)) \qquad (10)$$

A simpler but less tight upper bound for the response time of each component can be obtained considering a linear lower bound to the supply bound function, also proposed in [9]. This linear lower bound is depicted in Figure 5 and results in $\mathbf{sbf\_lb}_{y_x} = (t-(\Pi_{y_x}-\Delta_{y_x}-2\Theta_{y_x}))\alpha$. Therefore the response time upper bound ($RT\_up_{y_x}$) can be obtained replacing $\mathbf{sbf}_{y_x}$ in (10) by $\mathbf{sbf\_lb}_{y_x}$.

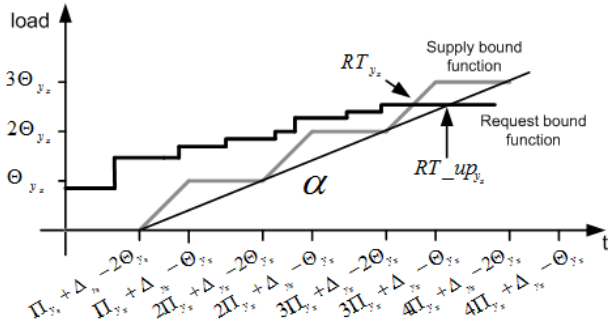

Figure 5: Response time

5 Conclusions

Component-based design is a powerful design paradigm to address the growing complexity of embedded applications. Moreover, server-based scheduling is an effective means to deploy component-based applications, particularly when organized in a hierarchical framework. In this paper, we addressed the case of multi-level hierarchical server-based scheduling within Ethernet switches using a specific switch, namely the FTT-enabled Ethernet Switch. We have developed a schedulability analysis that allows verifying whether a given composition of servers is schedulable. This analysis applies to cases in which servers might experience blocking caused by on-going packet transmissions associated to lower priority servers, and it complements the work in [3] that applies when such blocking is eliminated using the *blocking-free non-preemptive model*. This latter approach, however, presents a longer switching latency, which might not be desirable. A full comparisons and analysis of these two approaches will be carried out in future work.

References

[1] I. Shin and I. Lee, "Compositional real-time scheduling framework with periodic model," *ACM Trans. Embed. Comput. Syst.*, 2008.

[2] R. Santos, R. Marau, A. Oliveira, P. Pedreiras, and L. Almeida, "Designing a Costumized Ethernet Switch for Safe Hard Real-Time Communication," in *Proceedings of the 7th IEEE Workshop on Factory Communication Systems*, May 2008.

[3] R. Santos, A. Vieira, R. Marau, P. Pedreiras, A. Oliveira, L. Almeida, and T. Nolte, "Improving the efficiency of Ethernet switches for real-time communication," in *Proceedings of the 1st International Workshop on Adaptive Resource Management*, April 2010.

[4] J. Loeser and H. Haertig, "Low-Latency Hard Real-Time Communication over Switched Ethernet," in *Proc. of the 16th IEEE Euromicro Conf. on Real-Time Systems*, 2004.

[5] PROFInet, "Real-Time PROFInet IRT," http://www.profibus.com/pn.

[6] "Ethernet Powerlink - online information," http://www.ethernet-powerlink.org/.

[7] L. Deng and J. Liu, "Scheduling real-time applications in an open enviroment," in *Proceedings of the 18th IEEE Inter. Real-Time Systems Symposium*, December 1997.

[8] T. Kuo and C. Li, "A fixed-priority-driven open enviroment for real-time applications," in *Proc. of the 20th IEEE International Real-Time Systems Symposium*, December 1999.

[9] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *Proceedings of the 24th IEEE International Real-Time Systems Symposium*, December 2003.

[10] L. Almeida and P. Pedreiras, "Scheduling within temporal partitions: response-time analyisis and server design," in *Proceedings of the 4th ACM International Conference on Embedded Software*, September 2004.

[11] A. Easwaran, M. Anand, and I. Lee, "Compositional Analysis Framework using EDP Resource Models," in *Proceedings of the 28th IEEE International Real-Time Systems Symposium*, December 2007.

[12] M. Behnam, I. Shin, T. Nolte, and M. Nolin, "SIRAP: A synchronization Protocol for Hierarchical Resource Sharing in Real-Time Open Systems," in *Proceedings of the 7th ACM International Conference on Embedded Software*, October 2007.

[13] R. Santos, R. Marau, A. Vieira, P. Pedreiras, A. Oliveira, and L. Almeida, "A Synthesizable Ethernet Switch with Enhanced Real-Time Features," in *Proceedings of the 35th IEEE Industrial Electronics Society*, November 2009.

[14] R. Santos, A. Vieira, R. Marau, P. Pedreiras, A. Oliveira, L. Almeida, and T. Nolte, "Implementing Server-based Communications within Ethernet Switches," in *Proceedings of 2nd International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems*, December 2009.