Session 4 – Quality of Service

Chair: Eduardo Tovar

Rapporteur: Anis Koubâa

Session Summary: Quality of Service

Rapporteur: Anis Koubâa
IPP-HURRAY Research Group, Polytechnic Institute of Porto
Porto, Portugal
akoubaa@dei.isep.ipp.pt

1. Introduction

Quality of Service (QoS) is one of the major issues in the dimensioning of industrial networks and embedded systems, particularly when timing constraints need to be met. Roughly, the provision of QoS in Real-Time Networks (RTNs) means the specification and the set-up of the set of mechanisms necessary to meet the QoS constraints. The main QoS constraint that must be satisfied in RTNs is the message delay (or response time).

In that direction, The QoS session of the Real-Time Networks Workshop has focused on proposing new mechanisms for supporting and analyzing real-time QoS in industrial and home networks.

2. Talks

Motivated by the inefficiency of COTS Ethernet switches to guarantee real-time communication, the first paper of this session dealt with the proposal of a synchronized approach for ensuring deterministic real-time guarantees in industrial switched Ethernet based on Flexible Time Triggered (FTT) paradigm. The architecture of the FTT-SE (FTT- Switched Ethernet) is based on the master-slave model according to which the master polls its slaves periodically and the communication occurs during a time unit referred to as Elementary Cycle. It has been shown that the FTT-SE approach enables a noticeable improvement of timing constraints in switched Ethernet, but at the cost of a higher implementation complexity.

The second paper of this session was focused on the ability of Universal Plug and Play (UPnP) protocols to provide real-time guarantees in IP-based home networks. This paper proposed an abstract model that enables an efficient QoS management without having to know all underlying details in lower layers.

The last paper of this session proposed a priority based approach that facilitates the integration of several CAN-based subsystems. The basic idea of the paper was to decouple CAN identifiers from their priorities to avoid any kind of conflicts when interconnecting several subsystems together. A comparative time-predictability performance analysis of different decoupling protocols (FTT-CAN, TT-CAN and Server-CAN) has also been discussed.

Enhanced Ethernet Switching for Flexible Hard Real-Time Communication

Ricardo Marau, Paulo Pedreiras, Luís Almeida
DET/IEETA, Universidade de Aveiro, Portugal
{marau,pedreiras,lda}@det.ua.pt

Abstract

Switched Ethernet arose in the last decade as a means to increase global throughput with parallel switching paths, segment the network and create isolated collision domains, thus reducing the non-determinism of the original shared Ethernet. However the services provided by COTS Ethernet switches are not enough to guarantee real-time communication, which lead to the development of several switch Ethernet-based protocols, among which the recently proposed FTT-SE. This paper proposes moving the FTT traffic management into the Ethernet switch and discusses how this architectural change enhances the performance of the transmission control and service differentiation mechanisms as well as how error confinement mechanisms can be efficiently deployed. Preliminary experimental results from a prototype implementation validate the services provided by the enhanced Ethernet switch framework.

1 Introduction

Distributed Embedded Systems (DES) integrating intelligent cooperative nodes are found in a wide range of applications, from automotive to aerospace, passing through the lower layers of both process control and manufacturing industries [1]. In these environments, applications range from embedded command and control systems to image processing, monitoring, human-machine interfacing, etc.

Since its creation, Ethernet has been considered has a potential solution for use in DES due to its large bandwidth, cheap silicon, high availability, easy integration with Internet and clear path for future expandability [2]. Furthermore, using Ethernet also at the lower control level facilitates the vertical integration and may bring along several advantages in maintenance effort.

Ethernet, however, is a general purpose data network and was not originally designed to satisfy the requirements of DES. For this reason several modifications have been proposed, including restrictions to the traffic pattern generated by each node, modifications to the arbitration mechanism and addition of transmission control layers [9].

Since the early 90's the interest in switched Ethernet has been growing steadily, having practically replaced shared-Ethernet (single segment, hub-based). Despite avoiding message collisions and having built-in traffic scheduling capabilities (Figure 1), thus improving the predictability of the network with regard to shared Ethernet, in general switched Ethernet networks are not capable of delivering the real-time communication services needed by DES. Therefore, as for shared Ethernet, several techniques were proposed to overcome its limitations, from shaping the traffic submitted to the switch to limiting that traffic by application design, providing more efficient scheduling policies and admission control or adding transmission control features [9].

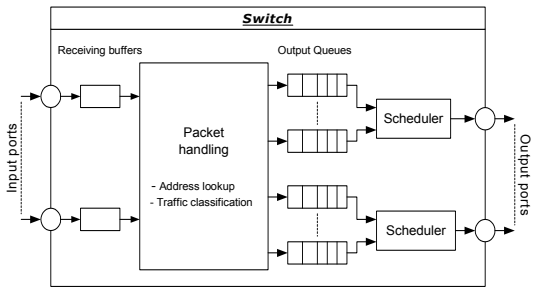

Figure 1: Internal switch architecture

The recently proposed FTT-SE [3] belongs to this latter category of protocols, and exhibits, as main features, global traffic coordination in a common timeline, the possibility for fast and atomic on-line updates to the set of streams, the possibility to support wide ranges of streams periods and the possibility to enforce any traffic scheduling policy.

While using non-standard hardware conflicts with some of the key arguments supporting the use of Ethernet in real-time applications (e.g. cost, availability, compatibility with general purpose LANs), custom switch implementations with enhanced traffic control and scheduling capabilities allows important performance and service breakthroughs, and so a number of approaches of this class have also been proposed in the recent years (e.g. [4], [6] [8]).

This paper proposes integrating the traffic management and transmission control mechanisms of the FTT-SE in an Ethernet switch. The resulting framework allows obtaining important performance gains in the following key aspects:

- A noticeable reduction in the switching latency jitter found in common Ethernet switches;
- An important performance boost of the asynchronous traffic, which in this case is autonomously triggered by the nodes instead of being pooled by the master node;
- An increase in the system integrity since unauthorized transmissions can be readily blocked at the switch input ports, thus not interfering with the rest of the system;
- Seamless integration of standard non FTT compliant nodes without jeopardizing the real-time services.

In the next section the FTT-SE protocol is briefly reviewed. Section 3 presents the architecture of the enhanced

The material in this paper is the subject of a current patent filing.

Ethernet switch. Section 4 describes a prototype implementation and presents some preliminary experimental results. Section 5 concludes the paper.

2 FTT-SE brief overview

FTT-SE is a recently proposed COTS-based real-time protocol [3] for micro-segmented switched Ethernet networks. The FTT-SE protocol is based on the Flexible Time-Triggered (FTT) paradigm and supports arbitrary traffic scheduling policies, periodic and sporadic traffic with temporal isolation, priority levels beyond the eight levels specified in IEEE 802.3D, on-line admission control and bandwidth management and, finally, completely avoids memory overflows inside the switch due to the global traffic scheduling mechanism.

2.1 FTT-SE Medium Access Control layer

The FTT-SE employs a technique called master/multi-slave, according to which the master addresses several slaves with a single poll, considerably alleviating the protocol overhead with regard to the conventional master-slave techniques. The communication occurs in fixed duration slots called Elementary Cycles (ECs), with one master message per cycle called Trigger Message (TM), which contains the periodic schedule for that EC. The periodic messages are referred to as synchronous since their transmission is synchronized with the periodic traffic scheduler. The protocol also supports aperiodic traffic, called asynchronous, which is managed in the background, in the time left within the EC, after the periodic traffic (Figure 2).

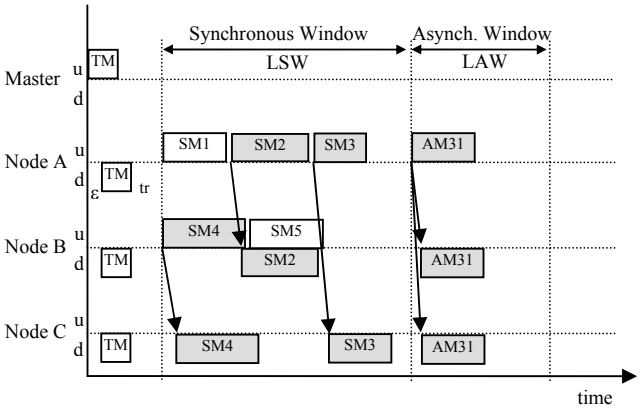

Figure 2: Traffic scheduling in FTT-SE

2.2 Synchronous traffic scheduling

The synchronous traffic scheduling activity is carried out on-line and centrally in the master and the periodic traffic schedules are disseminated by means of the TM (Figure 2). Since the traffic scheduling is local to one node, it is easy to enforce any kind of scheduling policy, as well as perform atomic changes in the communication requirements. This last feature allows for on-line stream admission and removal under guaranteed timeliness as well as on-line bandwidth management. Nodes decode the TM and transmit immediately the scheduled messages with the switch taking care of the serialization. All messages scheduled to one EC fit in that EC and so message queues have a limited and pre-known size and cannot build up from EC to EC.

The FTT master holds information about the nature of the data exchanges regarding the type of addressing (unicast, multicast and broadcast) and which end nodes are involved. With this information the master computes which messages follow disjoint paths (i.e., non overlapping source and destination nodes) and thus build schedules that exploit this parallelism, increasing the aggregated throughput. For non-multicast switches only unicast and broadcast streams can be considered. For true multicast switches the standard Internet Group Multicast Protocol (IGMP, RFC 2236) is used to setup up multicast groups.

2.3 Asynchronous traffic handling

Unconstrained aperiodic communication may generate bursts that fill in output queues, leading to long priority inversions in typical FIFO queues and possibly to queue overflow and consequent packet losses. Using switches with two (or more) priority levels and assigning to the asynchronous traffic a lower priority level than to the synchronous one does alleviate the problem. Nevertheless, due to the non-preemptive nature of packet transmission asynchronous messages can still block the synchronous messages or the TM. The blocking effect is, however, bounded to one packet. Adequate mechanisms are still required to constrain the asynchronous load and burstiness to prevent buffer overflows and consequent interference with the high priority periodic traffic [5][7]. Therefore, the use of traffic shaping or smoothing schemes is required.

Alternatively, polling can be used, being more robust and timely but less efficient. In this case, the transmission instants are adequately planned by the global scheduler but synchronization delays will increase the response times. In this case the asynchronous traffic is treated essentially as the synchronous one, except that slaves may or may not transmit a pooled message, depending on its readiness status.

FTT-SE can use any of the mechanisms above, depending on the requirements of each application. The polling approach is more adequate for situations requiring precise timeliness. When the non-preemption blocking is tolerable, the dual-priority approach seems better suited.

3 FTT enabled Ethernet switch architecture

Figure 3 depicts the FTT enabled Ethernet switch integrating the traffic management services provided by the Master node in FTT-SE systems. The System Requirements Database (SRDB) is the central repository for all the information related to the traffic management, namely the message attributes for both synchronous and asynchronous traffic (e.g. period/minimum inter-arrival time, length, priority, deadline), information about the resources allocated to each traffic class (e.g. phase durations, maximum amount of buffer memory) and global configuration information (e.g. elementary cycle duration, data rate). Change requests to the message set are submitted to an admission control (plus optional QoS manager), ensuring continued real-time traffic timeliness. The SRDB is periodically scanned by a scheduler, which builds a list of synchronous messages (EC-Schedule) that should be produced in the following EC. A dispatcher task periodically sends the EC-schedule to the

switch ports having attached FTT nodes.

For FTT-SE/FTT-Ethernet systems the master role is confined to the functionalities defined above. However, the integration of the switching services with the traffic scheduling permits a tight control of the packet flow and resource utilization inside the switch. At the beginning of each EC the global dispatcher directly accesses the port dispatcher, which sends the trigger message and keeps temporal information about each of the phases within the EC. Each output port has 3 queues, one for each traffic class (synchronous, asynchronous and non real-time messages (NRT)). During the EC the port dispatcher transmits messages submitted to each of these queues, according to the EC phase. This mechanism confines the different traffic classes to the respective phases. If e.g. a malfunction node sends a synchronous message outside of the synchronous phase, the message is discarded and does not interfere with the asynchronous or non real-time phases. On the other hand asynchronous messages (either real-time or non real-time) do not need to be pooled, contrarily to what happened for FTT-SE. The port dispatcher only transmits messages from the asynchronous or NRT queues if the time left within the respective window is enough.

Both FTT and non FTT-compliant nodes can be seamlessly attached to the FTT enabled switch. Thus, on the ingress side the first operation carried out is the packet classification, which consists only in inspecting the Ethernet type field. When the message is identified as an FTT message it is subject to a verification process and, if judged valid, is appended to the synchronous or asynchronous message queues, according to its nature. Conversely, if the message is non-FTT it is simply appended to the NRT queue. The segmentation of the global memory pool, keeping the messages of each class in independent subdivisions allows avoiding memory exhaustion for the real-time messages, a situation that standard switches do not guarantee [5]. The real-time traffic is subject to an explicit registration. During the registration process the producers must state the message properties, in particular the length and periodicity (for periodic messages; minimum inter-arrival time for sporadic ones). With this data it is possible to compute and pre-allocate the amount of memory that each traffic class requires and thus guarantee that the resources are enough for all admitted messages. These elements, however, are not available for the NRT traffic. Thus it is not possible to predict the amount of memory necessary and, consequently, the NRT queue may become full, leading to drops of NRT packets. However, the higher layers protocols (e.g. TCP) are tolerant to occasional message drops, only with a negative impact in the performance. This situation is not critical since this traffic is granted with best effort guarantees, only.

The validation process gathers data both from the EC-schedule and from the RTDB. Regarding synchronous messages, the analysis of the EC-schedule allows detecting failures in the time domain, namely the transmission of unscheduled messages or the late transmission of scheduled messages resulting from mal-function nodes. An equivalent set of tests (e.g. minimum inter-arrival time, burstiness) may also be performed for asynchronous messages with those that fail the validation process being trashed. The policing and enforcement of the traffic attributes in the time domain guarantees the timeliness of the real-time traffic even in the presence of malfunctioning nodes.

Whenever a message is placed in the global memory pool, a packet forwarding process is executed. Control messages, targeted to the master are submitted to the Admission control/QoS manager module and possibly result in changes on the SRDB. Data messages should be forwarded to the target nodes. The forwarding mechanism of FTT messages is based on a producer-consumer model, and does not depend on MAC addresses. Whenever an FTT message arrives the Packet Forwarding module inquires the SRDB to determine the set of ports having consumers attached, and updates the output queue (synchronous or asynchronous, depending on the message nature) of each one of these ports. Non-FTT messages are forwarded according to the normal procedures of standard Ethernet switches, based on the MAC address.

4 Experimental results

A prototype implementation, based on the RT-Linux real-time operating system with the Ethernet layer provided by the LNet network stack, was carried out to validate the extended services provided by the FTT-enabled switch. This prototype switch is based on a Pentium III PC at 550MHz with four 3Com 3C905B PCI network interface cards

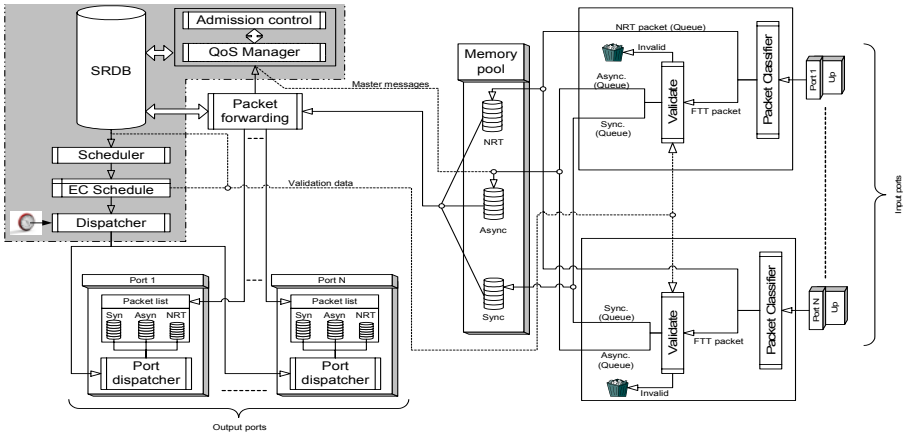

Figure 3: Switch internal architecture

The first experiment consists in the implementation of a policing service for the synchronous traffic. The ID of incoming synchronous messages is matched against the EC-schedule and discarded if a positive match is not found. This way only scheduled messages are disseminated, guaranteeing that the synchronous window is not overrun. To verify the correct behavior of the policing service we configured a setup with 1 synchronous message with period Ti=3ECs while the respective producer slave was tampered to send that message every EC. With the setup we observed that the consumer node only received the scheduled messages, one every 3ECs, and the extra messages were discarded.

The second experiment consists in the verification of the enforcement of the traffic temporal isolation. The experimental setup is configured with an EC of 40ms, with the last 3ms of the EC dedicated to the NRT traffic. The NRT test load consists in UDP packets carrying 1400 data bytes, periodically sent every 5ms. The load is generated with PackEth (http://packeth.sourceforge.net/) running on a plain Linux distribution (RedHat 9.0). Figure 4 depicts the histogram of the time differences between consecutive NRT messages in the uplink.

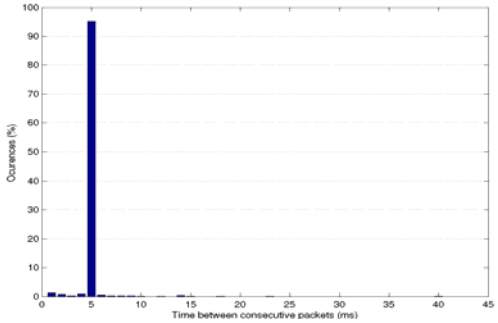

Figure 4: Histogram of the differences between consecutive NRT messages (uplink)

While NRT messages can be submitted at any time instant, the FTT-enabled switch only forwards them to the output port(s) in the NRT window, which in this setup is configured to use the last 3ms of the EC. This confinement mechanism significantly changes the message transmission pattern between the uplink and the downlink. Figure 5 shows the time difference between the beginning of the EC and the reception of the NRT messages being clear the confinement of these to the NRT window (37 to 40ms after the EC start). Therefore the NRT traffic does not interfere with the synchronous or asynchronous real-time traffic, despite being generated at arbitrary time instants by a standard node not implementing the FTT protocol.

5 Conclusions and future work

The advent of switched Ethernet has opened new perspectives for real-time communication over Ethernet. However, a few problems subsist related with queue management policies, queue overflows and limited priority support. While several techniques were proposed to overcome such difficulties, the use of standard Ethernet switches constraints the level of performance that may be achieved. In this paper we proposed an enhanced Ethernet switch, implementing FTT-class services. The resulting architecture inherits the FTT features, namely flexible communication with high level of control to guarantee timeliness, while permits a noticeable reduction in the switching latency jitter found in common Ethernet switches, an important performance increase of the asynchronous traffic, seamless integration of standard Ethernet nodes and a substantial increase in the system integrity as unauthorized transmissions from the nodes can be readily blocked at the switch input ports. On-going work addresses the FPGA implementation of the switch.

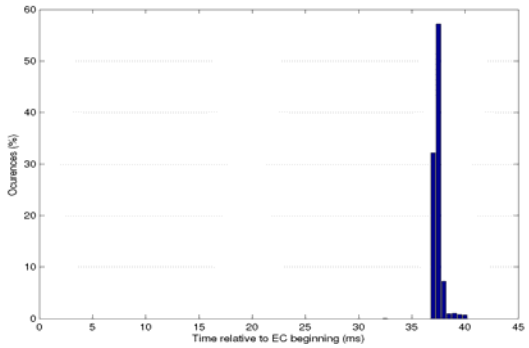

Figure 5: Histogram of the differences between the beginning of the EC and NRT messages (downlink)

6 References

[1] Thomesse, J-P. “Fieldbus and Interoperability”. Control Engeneering Practice, 7(1), pp81-94. 1999.

[2] Decotignie, J-D. A perspective on Ethernet as a Fieldbus. Proc of the FeT’2001 – 4th Int. Conf. on Fieldbus Systems and their Applications, pp138-143. Nancy, France. November 2001.

[3] Almeida, L., Pedreiras, P. "Hard Real-Time Communication over COTS Ethernet Switches". " Work in Progress Session of the 26th IEEE International Real-Time Systems Symposium.

[4] Hoang, H. Jonsson, M., Hagstrom, U., Kallerdahl, A. "Switched Real-Time Ethernet with Earliest Deadline First Scheduling - Protocols and Traffic Handling". Proc of WPDRTS 2002, the 10th Intl. Workshop on Parallel and Distributed Real-Time Systems. Fort Lauderdale, Florida, USA. April 2002.

[5] Pedreiras, P., R. Leite, L. Almeida. Characterizing the Real-Time Behavior of Prioritized Switched-Ethernet. RTLIA’03, 2nd Workshop on Real-Time LANs in the Internet Age, (satellite of ECRTS’03), Porto, Portugal, July 2003.

[6] Varadarajan, S., Chiueh, T. “EtheReal: A Host-Transparent Real-Time Fast Ethernet Switch”. Proc of the 6th Int Conference on Network Protocols, pp. 12-21. Austin, USA. Oct 1998.

[7] Loeser, J., H. Haertig. “Using Switched Ethernet for Hard Real-Time Communication”. Proc Parallel Computing in Electrical Engineering, International Conference on (PARELEC'04), pp. 349-353, September 07 - 10, 2004, Dresden, Germany.

[8] Real-Time PROFINET IRT. http://us.profibus.com/profinet/07

[9] P. Pedreiras, L. Almeida. Approaches to Enforce Real-Time Behavior in Ethernet. in *The Industrial Communication Systems Handbook*, R. Zurawski (ed). CRC Press, ISBN:0-8493-3077-7, 2005.

The network capability model of UPnP™-QoS v3, an interoperable QoS framework for admission control and scheduled access

Michael van Hartskamp[1]

Philips Research, High Tech Campus 34, 5656 AE Eindhoven, The Netherlands

***Abstract* — Home networks are increasing in popularity. For commercial content offerings, a high guaranteed Quality of Service is needed. The heterogeneous nature of the IP-based home network complicates delivering real-time guarantees. In this paper we describe the network capability model of a proposed interoperable middleware for QoS control on the basis of UPnP [1]. The model describes QoS capabilities of underlying Layer 2 QoS technologies. It enables QoS Managers to manage the network and understand the impact of their management actions, without explicit understanding of the details of every technology.**

***Key words* — Home Networks, Interoperability, Quality of Service, UPnP, UPnP-QoS.**

I. INTRODUCTION

Home networks are increasing in popularity. More and more households have (wireless) Ethernet-based home networks connecting their PC(s) or laptop(s) to the Internet. Currently applications such as IPTV and VoIP are entering the home, yet they are often terminated at the door step and do not take the final step through the home network. For most service providers, the lack of real-time or Quality of Service guarantees inhibit mass-market deployment.

While for commercial content offerings, a high guaranteed Quality of Service is needed, the currently deployed technologies, such as the IP-protocol suite, Ethernet and Wi-Fi / Wireless Ethernet, do not go beyond 1) best effort delivery and 2) an attempt at fairness to all network users. In other words they do not provide the required real-time guarantees. Future Layer 2 technologies, which are currently or were recently standardized, such as IEEE 802.11e, WiNET, and HomePlug AV do provide mechanisms for admission control and scheduled delivery of packets, bringing support for real-time. What is still lacking for an actual deployment is a standardized middleware that provides applications with a uniform interface to use those features of the different underlying technologies in heterogeneous networks.

In this paper we contribute a network capability model. The network capability model indicates for heterogeneous networks, the devices of which have certain QoS capabilities. This model is proposed as part of an interoperable middleware for QoS control on the basis of UPnP [1]. UPnP is the de facto standard for discovery and description of home networking devices. The UPnP forum has also defined device control protocols for various devices such as AV media servers and AV media renderers. UPnP is also the backbone for the standards and guidelines of DLNA [2].

Our approach is based on the assumption that admission control is essential to guard a sufficient minimum level of Quality of Service by preventing structural overloading of the network. It is understood, that admission control is a necessary but not sufficient condition. In wireless or power line networks, there are no hard guarantees and hence point-to-point solutions have to be applied to maintain an acceptable quality in the presence of varying resource availability. These techniques are not discussed in this paper.

The remainder of this paper is structured as follows. In Section II we provide a short overview of the home network and in particular UPnP. The following section describes some of the relevant details and workings of IEEE 802.11e in more detail.

Section V presents an overview of our solution. The solution is based on a network capability model. The model is proposed for inclusion in UPnP-QoS version 3. It allows the discovery of layer 2 capabilities of certain devices. This enables the end-to-end QoS setup.

The following section details the behavior of the involved devices by describing the interaction and the relation with Layer 2 setup.

Section VII shortly discusses how the proposed solution enables decomposition of end-to-end requirements such as delay and loss requirements. With our solution it is possible to avoid local optimizations.

Finally we draw our conclusions. The network capability model is a suitable way to describe and manage the QoS capabilities present in current and future home networks. The submission to UPnP-QoS enables applications to manage QoS in heterogeneous networks in a standardized way.

II. HOME NETWORKING

A. Home Networking Architectures

The DLNA guidelines provide a model for logically describing Home Networking AV devices (see [3]). The DLNA devices are assumed to be used for applications such as AV streaming of various quality, media uploads and downloads, etc.

The DLNA model is based on, what are at least traditionally, standards from the IT-world: Ethernet, IP,

[1] Parts of the work reported here were performed within the ongoing European Research Program BETSY- IST-004042.

HTTP; protocols that are not traditionally known for their real-time capabilities. But most of them so well established that their unmodified adoption is essential and their limitations need to be circumvented.

For discovery, description, and control, the DLNA guidelines use the UPnP protocol.

B. UPnP

In UPnP, a Control Point invokes actions on a UPnP service which is running on a UPnP device. The UPnP device advertises one or more services with certain (often standardized) actions. The Control Point, whose behavior is not standardized, determines capabilities of a device (service) and then decides what the device (service) will do. The UPnP forum has already standardized various services and devices for applications ranging from home automation, via gateways to AV-applications.

Since 2005, services enabling priority-based QoS are defined. Through UPnP an application interfaces with Layer 2 (or 3) technologies for QoS, by-passing the IP-layer [4].

Currently in UPnP, the QoS working committee is extending these services to support admission control and scheduled access, i.e. parameterized QoS. In this paper we describe our network capability model which is proposed as a part of these extensions.

III. IEEE 802.11E

For the heterogeneous home network, it is crucial to work on the basis of different existing layer 2 technologies. In this section we provide background on a popular Layer 2 technology: Wi-Fi WMM and the underlying IEEE 802.11e [5] to motivate our network capability model.

WMM is a certification program of the Wi-Fi alliance on the basis of IEEE 802.11e, prioritized QoS. It also offers a simple admission control functionality. The complete IEEE 802.11e also specifies scheduled access and it is expected that this will also become part of a Wi-Fi certification program. In the remainder we consider the scheduled access function "HCCA" of IEEE 802.11e.

In IEEE 802.11e, a wireless QoS enabled station (QSTA) connects to the QoS-enabled Access Point (QAP). Two stations connected to the same AP communicate via the AP. With HCCA the QAP polls a QSTA after which the QSTA may transmit a packet.

Requests for QoS are always initiated from a QSTA, whether the station will be sending or receiving. The requests present a traffic specification consisting of among many others mean data rate, peak data rate, delay bound, and minimum PHY rate to the QAP which subsequently decides on the viability. Typically the QAP has an overview of all admitted streams, but individual QSTA(s) do not.

To save bandwidth, a direct link protocol enables direct transmission between two QSTA, by-passing the QAP. In this case, the sending station is responsible for the QoS request.

IV. MAJOR DESIGN ALTERNATIVES

In this paper, we follow a centralized approach for QoS management. The basic idea behind the centralized approach is that a QoS request is forwarded to a central entity. This entity decomposes end-to-end requirements and subsequently appropriately instructs the individual devices. Another example of a centralized approach is in [8] and it shows how real-time requirements can be met in a heterogeneous environment.

We believe this centralized solution is possible given the small size of a typical home network. When meeting end-to-end requirements we do not have to suffer from local optimizations. But to enable the central controller to make such "wise" decisions, some information has to flow from the individual devices to the controller in order to support its decisions.

This is different from an RSVP-style approach (see [6]) where the receiver sends a QoS request to the sender. On its way to the sender it passes devices. Every device determines whether it can support the request. If not the request is denied and returned to the receiver. When the request reaches the source and is accepted, a positive acknowledgment is returned. Some provisions have been foreseen for shared media [7] but these are not really used. End-to-end requirements are decomposed at every intermediate device by tightening the requirements for the others upstream.

There are two prime reasons for us to choose a centralized approach. First, the central approach is in line with the UPnP device architecture where a Control Point instructs a service on a device to do something. Secondly, a centralized approach, at least theoretically, allows better decompositions of end-to-end (real-time) requirements which a per-hop approach does not bring.

V. THE NETWORK CAPABILITY MODEL

In this section we describe our network capability model. Our approach is based on UPnP and we follow the design principles of UPnP by employing discovery and description in this case to discovery and description of QoS capabilities. In this paper we describe the network capability model which we have proposed to version 3 of the QosDevice service. For space reasons we can only describe one mapping of the model to a layer 2 technology, which is on IEEE 802.11e.

The goal of the model is enabling end-to-end requirements decomposition into, commonly called, "per-hop requirements" and appropriately configuring the network such that those "per-hop requirements" can be met. For this the network capability model needs to define those "hops" and indicate which devices have the capabilities to manage "which hop".

In our approach the "per-hop" concept is formalized by the concept of QoS segment to accommodate Layer 2 technologies such as AV Bridges [9] that come with their own QoS management spanning multiple "hops".

The basic QoS capabilities are admission of a stream and the release of resources. Another capability is to list the admitted streams to get an impression of the occupied resources. Since the model was derived with various actual implementations and technologies in mind, the capabilities to admit and release are often capabilities to perform Layer 2 signaling that leads to admission or release.

One of the crucial steps to take is to identify which device has the QoS capability to admit a certain stream in a certain QoS segment. The QoS capabilities are often topologically qualified, i.e., certain technologies or implementations offer in a certain device only admission for a specific link or even just for the outgoing direction, etc.

In the next section we go into the details of the model, provide and provide an example.

A. Segmentation

First consider the network as a graph consisting of vertices and edges. Since network connections are not necessarily bidirectional, edges are directed. The example of the wireless network where traffic can flow through the AP as well as through a Direct Link indicates that even with the Layer 2 spanning tree protocol the graph is not necessarily loop-free.

The goal of the segmentation is to capture the extent to which middleware actions induce Layer 2 signaling. Those middleware actions are the UPnP-actions invoked on a UPnP device by a UPnP Control Point called QoS Management Entity. For the middleware management to work some independence between the actions is needed. I.e., the middleware actions must not have side effects and preferably not unnecessary restrict the order of invocation.

We now define a QoS segment in an abstract way as a unit of independent management.

The collection of all QoS segments is closed under finite union and finite intersection. It is a cover of the graph. It is easy to see that there is a subcover such that every edge is contained in exactly one element of this subcover. The minimal QoS segments containing at least one edge typically coincide with the Layer 2 domains. In every actual deployment, such a subcover is determined through technology specific rules followed by the individual devices.

As a general principle, the smaller the QoS segments in the home network the more management steps are performed by a QoS Management Entity.

B. QoS capabilities

The three primary QoS capabilities are: Admit, Release, and List. When a device has the capability to admit it is capable to perform an admission control function as well as to reserve the resources (on networks governed by scheduled access).

There are different ways in which a device can support the capability to admit. A straight forward method is for a device such as a wireless QAP or another device with the Layer 2 scheduler. Such a device could (theoretically) easily perform algorithms for admission control and calculate appropriate schedulers. However, in many actual implementations this capability cannot be used directly. A method which is supported with (nearly) every technology is to rely on Layer 2 signaling. A device implements the capability to admit whenever it has the ability to use Layer 2 signaling to perform admission control. In this case the Layer 2 signaling protocol is limiting. The request may support only a limited set of parameters (e.g. only peak and not average bandwidth), return a limited answer (yes, no, but not “no, but you can admit x bits per second”) and most frequently only for a limited number of streams, e.g., only for streams that flow through the device. In some Layer 2 technologies the latter limitation is not there and similarly general results as with admission at the scheduler can be obtained.

The capability to Release is similar and frees the assigned network resources. The capability to List enables the listing of streams that were assigned resources. This capability makes most sense when available at the device with the scheduler. It allows a QoS Management Entity to determine the viability of an admission before actually making it through UPnP-QoS and induced Layer 2 signaling.

C. Topological Qualifiers

As indicated by the examples most QoS capabilities are not general but qualified. A device can expose a limited capability, e.g., due to layer 2 signaling limitations. The following topological qualifiers are identified in the model.

QoS segment for an Interface: this device offers the QoS capability for every stream on the entire QoS segment for the given interface. A possible device to expose this capability is on the QAP for the QoS segment of the Access point and its associated stations. But see the practical limitations above.

Link: this device offers the QoS capability for the specified link. An example can again be found in IEEE 802.11e. A station (QSTA) identifies a link to the AP and the station has the capability to set up QoS for that link through the Layer 2 signaling. The Layer 2 signaling does not enable a station to set up QoS both for the link to the QAP and onwards to a next station. As an example, a QoS Management Entity will read the model and derive that this station can set up QoS from the station to the QAP but that it needs another action to set up QoS from the QAP to the next station. The fact that both links share the same underlying network medium is (weakly) expressed through the fact that both links are in the same QoS segment.

Direction: this device offers the QoS capability only for the specified direction of the identified link. Consider the Direct Link example. Here the device exposes the capability to admit only for the outgoing direction of the direct link.

D. Example of reported QoS capabilities

In this section we provide a simple example. Consider Figure 1 below representing a home with 6 wireless stations in 2 wireless networks connected through a wired backbone. There are three "minimal" QoS segments S, T, and U.

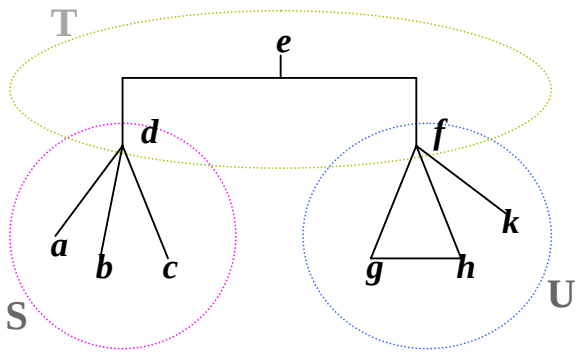

Figure 1 Example home network with QOS capabilities

We will first describe the reported QoS capabilities and then in the following section how to make use of these capabilities.

In segment S, device *d* offers the QoS capabilities to admit, release and list for the entire segment. Device *a* offers the admit and release capabilities for the link $a \leftrightarrow d$, *b* for $b \leftrightarrow d$, and *c* for $c \leftrightarrow d$, where $x \leftrightarrow y$ denotes the link (in both directions) between *x* and *y*.

In segment T, no QoS capabilities are available, e.g. with plain Ethernet. There can be no QoS management.

In segment U, device *f* offers the QoS capability to list. *g* offers the admit and release capabilities for links $g \leftrightarrow f$ and $g \rightarrow h$ (i.e. only in the direction from *g* to *h*). Similarly device *h* offers it for the links $h \leftrightarrow f$ and $h \rightarrow g$. But *k* only offers these capabilities for $k \leftrightarrow f$.

VI. Use of the Model

A device interested in participating in QoS setup hosts the UPnP QosDevice service. For every interface, the device determines the QoS segment in which it participates. The QoS segment is identified through an agreed Layer 2 specific algorithm: typically the identity of an elected or pre-determined leader is used, e.g., in wireless the MAC address of the AP or in IEEE 1394 or AV bridges the "root".

Next this QosDevice service reports its QoS capabilities. The topological qualifications are expressed via a hierarchically ordered structure (QoS segment, link, direction(s)) to avoid unnecessary repetitions.

Let us now follow the steps of performing an end-to-end admission for a stream flowing from *a* to *h* in the figure.

A QoS Management Entity is a UPnP control point for the (standardized) QosDevice service. The UPnP functionality is used to identify the available services. Through actions on the QosDevice service and comparison of the QoS segment IDs the QoS Management Entity determines the QoS segmentation of the network. This is also illustrated in the example above where the QoS Management Entity has now identified segments S, T, and U and may safely conclude that setup actions in segment S will not impact segment T or U (and similarly for segment T and U)

Also the path of the stream for which QoS is requested is determined. The QosDevice (since version 1) provides information to assist in the path determination.

Now the QoS Management Entity proceeds to setup QoS for every segment. For a given segment, the following steps are performed.

First, identify whether a QosDevice in the segment advertises the QoS capability to admit for the entire segment. If so (for example at device *d* in segment S), it invokes the admit action on this QosDevice and the process is completed. If not (for example in segment U), for every link in the QoS segment through which the stream passes, it identifies whether there is a QosDevice service which advertises the QoS capability to admit for that link (and in the desired direction) and admit at this device. In our example the link $f \leftrightarrow h$ is a relevant link and *h* offers the capability to admit.

After actually performing the admission request on the individual devices, device *d* probably performs only some internal calculations towards a new schedule. Device *h* on the other hand needs to rely on layer 2 signaling to *f* to effectuate its capability to admit. Device *f* will execute such signaling and then report the result to the QoS Management Entity.

Observe that if the stream passes through multiple links in the QoS segment, every link has to be individually set up through an admission action (cf. the example of IEEE 802.11e).

Finally, a QosDevice that receives the admit action registers the request and either performs the admission itself and updates its scheduler's polling tables, or performs a Layer 2 specific request which ensures schedules are setup and/or admission is evaluated.

VII. Some Remarks on End-2-End Requirements

The network capability model as described here indicates *whether* devices have the QoS capability to admit. This model does not describe how the admission request has to be formulated if such a QoS capability exists. For the latter a standard traffic specification needs to be used which lists several parameters which are commonly used in current Layer 2 technologies: such as mean bandwidth and peak bandwidth, but also other parameters relating to loss and delay requirements.

We believe the approach we have taken, i.e. enabling a QoS Management Entity to manage the end-to-end QoS setup allows avoiding local optimizations. It is expected that through additional interaction with the QosDevice service, local preferences on the decomposition are brought to the attention of the QoS Management Entity.

VIII. Conclusions

This paper describes a network capability model for discovery and description of Quality of Service capabilities of devices. The model is rich enough to capture common state-of-the art link-layer QoS technologies such as IEEE 802.11e (others such as HomePlug AV, MoCA and AVB were verified outside this paper), yet sufficiently abstract to enable efficient QoS management without awareness of all details of every underlying Layer 2 QoS technology. With knowledge of the model a QoS management entity can perform end-to-end admission control on the home network where needed by relying on layer 2 signaling and understanding interdependencies between setup of parts of the network.

The model also facilitates the decomposition of end-to-end requirements in a centralized manner, avoiding sub-optimal decisions as are possible in e.g. RSVP.

The model is part of UPnP-QoS. In this way applications are offered an interoperable mechanism for their QoS management.

Acknowledgement

The author wishes to acknowledge the members of the UPnP-QoS working committee and in particular Daryl Hlasny (Sharp), Ally Yu-kyoung Song (LGE), and Puneet Sharma (HP) for many interesting and fruitful discussions that lead to the described model.

The author also wishes to thank Gerhard Fohler for his stimulating knowledge on real-time and IP.

Finally the author wishes to thank Peter van der Stok for a long-standing cooperation on QoS in general and his remarks and help to improve this paper in particular.

References

[1] UPnP, UPnP Device Architecture 1.0, available on www.upnp.org
[2] Digital Living Network Alliance, www.dlna.org
[3] DLNA Press presentation march 2006, www.dlna.org
[4] UPnP-QoS QosDevice service definition, 2005, available on www.upnp.org
[5] IEEE, "IEEE 802.11e," *IEEE Amendment*, 2005.
[6] R. Braden, ed., RFC 2205, *Resource ReSerVation Protocol (RSVP)*, 1997
[7] R. Yavatkar, et al., RFC 2814, *SBM (Subnet Bandwidth Manager): A Protocol for RSVP-based Admission Control over IEEE 802-style networks*, 2000
[8] L. Rizvanovic and G. Fohler, *The MATRIX: A QoS Framework for Streaming in Heterogeneous Systems*, International Workshop on Real-Time for Multimedia (RTMM), 2004
[9] M. Johas Teener, AV Bridge summary, http://www.ieee802.org /1/files/public/docs2006/resb-mjt-bridge-summary-060111.pdf

This page left intentionally blank

Facilitating subsystem integration by decoupling priority and identifier in CAN messages

Thomas Nolte†‡, Lucia Lo Bello‡
‡RETISNET Lab, Dept. of Computer Engineering and Telecommunications
University of Catania
Catania, ITALY

Hans A. Hansson †
†Mälardalen Real-Time Research Centre
Dept. of Computer Science and Electronics
Mälardalen University
Västerås, SWEDEN

Abstract

When integrating subsystems on a common shared communication infrastructure these subsystems are likely to suffer from, and introduce interference among, each other. For CAN-based systems, the CAN message identifier is especially important, as it not only does identify the message, but it also determines the message's priority. Hence, special care needs to be taken when assigning identifiers to messages. This paper outlines how CAN-based systems are engineered today, and indicates the potential and benefits of decoupling the message priority from the message identifier. Three solutions to this are existing today: TT-CAN, FTT-CAN and Server-CAN. In this paper their strengths and weaknesses in an integration context are discussed. Also, the flexibility offered by the solutions is compared.

1 Introduction

The Controller Area Network (CAN) [8] is one of the major network technologies used in many application domains requiring embedded communications. It is particularly important in the automotive domain. A typical CAN application is any type of embedded system with real-time requirements and cycle times of $5 - 50ms$. However, CAN is used for many non real-time applications as well.

Traditionally in many application domains, subsystems have been developed incrementally as new functionalities have been added to the system. Looking at the automotive domain, value- and safety-adding functions, such as ABS and VDC, have been introduced over the years. Initially, they could be integrated as (mostly) independent subsystems having their own dedicated hardware in terms of Electronic Control Units (ECUs) and communications network. However, as the number of subsystems increases, there are strong trends towards integration of the subsystems on a common distributed architecture, rather than using a separate architecture for each subsystem. Hence, a crucial issue to solve is the migration from federated systems to integrated systems [10].

Looking at CAN-based embedded systems, it is a natural consequence that subsystems affect each others temporal performance once they are integrated on the same CAN network. This is due to the characteristics of the CAN message identifier, which defines both message priority (CAN PRIO) and message identity (CAN ID). Hence, decoupling the CAN ID from the CAN PRIO has the potential to simplify the integration process of CAN-based systems, allowing for flexible usage of CAN IDs. In this paper, three techniques that decouple the CAN ID from the CAN PRIO are presented, and their strengths and weaknesses are discussed from an integration point of view. Also, their provided offline (e.g., at design time) and online (e.g., allowing for dynamic addition and removal of subsystems) flexibility is compared.

Note that, in a CAN-based system, IDs are required to be unique for two reasons (1) due to the CAN arbitration mechanism (i.e., message collision resolution) and (2) to allow for message filtering (i.e., message identification). This implies that not any two subsystems are allowed to send a message with the same CAN ID at the same time. If this would happen as a consequence when integrating subsystems, one of them must change its conflicting IDs.

However, by the usage of recent schedulers, such as TT-CAN [9], FTT-CAN [1] and Server-CAN [15, 16], that decouple CAN ID and CAN PRIO, this change of IDs would not affect the temporal performance of the subsystems, thus allowing for an easier integration.

The outline of this paper is as follows: In Section 2 the technical properties of CAN message frame transmission are explained, showing the role of the CAN ID. In Section 3 CAN ID assignment in practice is shown, followed by Section 4 presenting three CAN schedulers that decouple the

CAN ID from the CAN PRIO. Section 5 discusses the three schedulers in the context of integration. Finally, the paper is concluded in Section 6.

2 CAN technical properties

CAN is a broadcast bus, which uses deterministic collision resolution to control access to the bus (so-called Carrier Sense Multiple Access / Collision Resolution, CSMA/CR). CAN transmits messages in an event-triggered fashion using frames containing 0 to 8 bytes of payload data. These frames can be transmitted at speeds of 10 Kbps up to 1 Mbps.

2.1 Frame arbitration

The CAN ID is required to be unique, in the sense that two simultaneously active frames originating from different sources must have distinct CAN IDs. Depending on the CAN standard used, the CAN ID can be either 11 bit (standard format) or 29 bit (extended format). Besides identifying the frame, the CAN ID serves two purposes: (1) assigning a priority to the frame, and (2) enabling receivers to filter frames (identifying the contents of the frame).

The basis for the access mechanism is the electrical characteristics of a CAN bus. During arbitration, competing communication adapters simultaneously put their CAN IDs, one bit at the time, on the bus. Bit value “0” is the dominant value. Hence, if two or more communication adapters are transmitting bits at the same time, and if at least one communications adapter transmits a “0”, then the value of the bus will be “0”. By monitoring the resulting bus value, a communications adapter detects if there is a competing higher priority frame (i.e., a frame with a numerically lower CAN ID), and in such a case it stops transmission. Because CAN IDs are unique within the system, a communications adapter transmitting the last bit of the CAN ID without detecting a higher priority frame must be transmitting the highest priority active frame, and can start transmitting the body of the frame, i.e., following the CSMA/CR rule. CAN therefore behaves as a global priority-based queue [1], i.e., a fixed priority non pre-emptive system, since at all communication adapters (nodes) the message chosen during arbitration is always the active message with the highest priority. Globally, the message with the highest priority will always be selected for message transmission.

3 Assigning CAN IDs in practice

The CAN ID is often used for scheduling purposes to fulfil temporal requirements. Here, the CAN ID can be set manually (static) or according to some online algorithm (dynamic). Also, the CAN ID can be assigned to message frames by the usage of tools, allowing for a design of the CAN-based system on a higher level than on per-message basis. Finally, as specified by several standards, the CAN ID can be solely used for message identification purposes.

[1] This is true if “FullCAN” communication controllers are used, e.g., Intel 82527 and Microchip MCP2510.

3.1 Static CAN ID used for scheduling

From a scheduling point of view, the most natural CAN ID assignment method is found in the context of priority-driven scheduling, where priorities are assigned to messages according to the Rate Monotonic policy [11]. This since Fixed Priority Scheduling (FPS) is the scheduling policy implemented by the CAN arbitration mechanism. Real-time analysis techniques have been presented to determine the schedulability of CAN message frames [21], and revisited in [2].

3.2 Dynamic CAN ID used for scheduling

Several methods for Dynamic Priority Scheduling (DPS) have been proposed for CAN. By manipulating the CAN ID online, and therefore dynamically changing the message priority, several approaches to mimic Earliest Deadline First (EDF) type of scheduling have been presented [6, 12, 22].

For example, the usage of a Mixed Traffic Scheduler (MTS) [22] attempts to achieve a high utilisation (like EDF). Using the MTS, the CAN IDs are manipulated online in order to reflect the current deadline of each message. Hence, since the CAN ID is dynamically changing, only a part of it can be used for actual identification of the CAN frame.

Looking at non real-time messages, a common way to send them on a CAN network is to allocate them message identifiers with lower priority than all real-time messages. In this way blocking of a real-time message by non real-time messages can be restricted to at most the duration of the transmission of one message. However, unwise message identifier assignment to non real-time messages could cause some of them to suffer from starvation. To provide Quality of Service (QoS) for non real-time messages several approaches have been presented [4, 13]. These approaches dynamically change message identifiers in a way preventing systematic penalisation of some specific messages.

In general, by dynamically manipulating the IDs of the CAN frames, these solutions all reduce the number of possible CAN IDs to be used by the system designers. This could be problematic, since it interferes with other design activities, and is even sometimes in conflict with adopted standards and recommendations [5, 19].

3.3 Static CAN ID assigned by tools

The growing complexity of automotive networked systems has resulted in tools to assist the system designer in the CAN ID allocation process. To understand this complexity, consider [15] where the network architectures for three cars are given. Here, the Volvo XC90 is said to contain around 40 ECUs, the BMW 7 series has around 65, and the VW Passat has around 45. Other car models are known to have up to 70 ECUs. These ECUs are part of the automotive subsystems to be integrated on a shared automotive architecture. The complexity of such an integration process is apparent, as the CAN ID represents the message priority.

As a step to overcome this, one example is found in the case of the XC90, where Volvo is using the Volcano concept [3, 18]. The Volcano concept provides tools for packaging data (signals) into network frames, both for CAN and other networks. The result is that several signals are allocated into message frames with an appropriate static CAN ID to fulfil these signals' temporal requirements. This simplifies the design, development and maintenance of an automotive system. Using the Volcano tools it is also possible to perform a timing analysis of the system, needed at the design stage to schedule the transmissions of real-time variables in such a way that their timing constraints are met.

3.4 Static CAN ID specified by standards

Several standards specify the usage of CAN IDs for various application domains. For example, the CANopen protocol [5] was released in 1995, designed for motion-oriented machine control networks, and can be found in various application domains today, e.g., medical equipment, off-road vehicles, maritime electronics, public transportation, building automation etc. The CANopen standards cover application layer and communication profile, a framework for programmable devices, and recommendations for cables and connectors. The application layer and the CAN-based profiles are implemented in software.

Another example is SAE J1939 [20], published by SAE in 1998, which specifies how messages are defined for engine, transmission and brake systems in truck and trailer applications. Nowadays, SAE J1939 is widely used in these applications, and standardised as ISO 11992.

Looking at other application domains, for tractors and machinery for agriculture and forestry, a SAE J1939-based ISO standard is used: ISO 11783 [7], and NMEA 2000® [14] defines a SAE J1939/ISO 11783 based protocol for marine usage.

These standards all restrict the use of message identifiers by prescribing which identifiers to use for messages carrying specific data.

4 Decoupling CAN ID from CAN PRIO

In this section we describe three CAN schedulers that decouple the message priority from the message identifier: TT-CAN, FTT-CAN and Server-CAN.

4.1 TT-CAN

Time-triggered communication on CAN is specified as TT-CAN, the ISO 11898-4 standard, an extension to original CAN. In TT-CAN, the exchange of messages is controlled by the progression of time, and all nodes are following a pre-defined static schedule. One node, the master node, is periodically (or on the occurrence of a specific event) transmitting a specific message, the Reference Message (RM), which acts as a reference in time. All nodes in the system are synchronising with this message, which gives a reference point in the temporal domain for the static schedule of the message transactions. This schedule is based on a time division scheme, where message exchanges may only occur during specific time slots or in time windows (so called Time Division Multiple Access, TDMA). Hence, the master's view of time is referred to as the network's global time.

4.2 FTT-CAN

Flexible Time-Triggered CAN (FTT-CAN) supports priority-driven scheduling in combination with time-driven scheduling. In FTT-CAN, time is partitioned into fixed size Elementary Cycles (ECs) that are initiated by a special message, the Trigger Message (TM). This message contains the schedule for the synchronous traffic (time-triggered traffic) that shall be sent within this EC. The schedule is calculated and sent by a specific node called the master node. FTT-CAN supports both periodic and aperiodic traffic by dividing the EC in two parts. In the first part, the asynchronous window, a (possibly empty) set of aperiodic messages are sent using CAN's native arbitration mechanism. In the second part, the synchronous window, traffic is sent according to the schedule delivered in the TM. The synchronous window can be scheduled according to an arbitrary scheduling policy. Experimental results have been shown for EDF in [17].

4.3 Server-CAN

Using Server-CAN, as with FTT-CAN, the network is scheduled by a specialised master node (called M-Server), partitioning time into ECs. Also, these ECs are initiated by a specific message, the TM, which is constructed and sent by the master node. By having a centralised scheduler, various (also EDF-based) share-driven scheduling policies can

	Static CAN ID used for scheduling	Dynamic CAN ID used for scheduling	Static CAN ID assigned by tools	Static CAN ID specified by standards
Objective	To provide predictable real-time message transmissions on the CSMA/CR MAC implemented by CAN.	Same as "Static CAN ID used for scheduling" although providing a higher schedulability bound or increased fairness of non real-time messages.	To achieve a system optimised with respect to, e.g., schedulability and/or message response-times.	To allow for interoperability between subsystems, usually application domain specific.
Usage	Academically scrutinised and nowadays often found in embedded systems.	Academic.	A common application is found in the automotive domain, where cars are characterised by high manufacturing volumes and stringent product cost pressures.	Examples are trucks, trailers, tractors, heavy vehicles etc. that usually are more "open" for configurability than cars, which in turn require the usage of standards.
Overhead on nodes	None, as no processing is required.	Some, as the CAN IDs are manipulated during runtime.	Small, due to encoding and decoding of signals into message frames.	None, as no processing is required.
Overhead on network	None, as no overhead is introduced in the messages and no protocol specific messages are sent.	None, as no overhead is introduced in the messages and no protocol specific messages are sent.	None, as no overhead is introduced in the messages and no protocol specific messages are sent.	None, as no overhead is introduced in the messages and no protocol specific messages are sent.
Offline flexibility	Yes, as the system designer has full control over the CAN IDs (although it might be complex without assisting tools).	Less compared with "Static CAN ID used for scheduling", due to a lower number of available CAN IDs.	Yes, more or less depending on the tool used.	None, as the standards have to be followed.
Online flexibility	None, as nothing is done online.	None, although the online manipulation of CAN IDs has the potential to provide a dynamic behaviour.	None, as nothing is done online.	None, as nothing is done online.
Facilitating subsystem integration	No, as the objective here is predictability on a message basis rather than simplifying subsystem integration.	No, and even less compared with "Static CAN ID used for scheduling", due to the lower number of CAN IDs available to use.	Yes, provided that all system details are given to the tool so that a proper optimisation can be made.	Yes, given that all subsystems comply with the standard.

Table 1. Comparative assessments of CAN IDs in practice.

be implemented. The difference between Server-CAN and FTT-CAN is that the latter has fixed size ECs whereas when using Server-CAN the length of an EC is upper-bounded, but can be shortened depending on the actual messages sent. In this way, efficient usage of network slack (e.g., when messages are not sent, or shorter than initially intended) is provided. Also, Server-CAN does not have windows inside an EC. Instead, all traffic is scheduled using network access servers denoted N-Servers.

5 Discussion on subsystem integration

Looking at how CAN IDs are assigned in practice, Table 1 outlines the differences among the four approaches presented in Section 3. Properties evaluated are overhead, flexibility and support of subsystem integration.

Static and dynamic CAN IDs used for scheduling purposes provide valuable scheduling performance but no inherent mechanisms supporting subsystem integration. Using static CAN IDs there is a strong dependency between the subsystem and its corresponding set of CAN IDs. Once several subsystems are integrated, there is a risk of temporal conflicts caused by the originally assigned priorities of the messages belonging to the different subsystems. Using dynamic CAN IDs the situation can even be worse, in the sense that the number of available CAN IDs is lower as (commonly) part of the CAN ID is used to dynamically adjust the message's priority.

The usage of a tool to optimise CAN ID assignments is good from an integration point of view, however, the runtime flexibility is limited as nothing extra (compared with

	Decoupling using TT-CAN	Decoupling using FTT-CAN	Decoupling using Server-CAN
Objective	To provide a time-triggered session layer to standard CAN.	To provide flexibility to CAN, allowing for a mix of statically and dynamically scheduled messages. Time is partitioned into fixed size ECs that in turn are partitioned into time- and event-triggered windows .	To provide a uniform way of handling message streams on CAN. Time is partitioned into dynamic size ECs, where time- and event-triggered traffic are jointly scheduled using server-based techniques.
Usage	Although a SAE standard, not widely used. Found in, e.g., some automotive concept applications.	Academic and educational use (robotics).	Academic only (so far).
Overhead on nodes	Some, due to the TT-CAN session layer (although hardware implementations exist). Higher overhead on the master node compared with the other nodes.	Yes, due to the online scheduling and TM encoding performed by the master node. Also, some overhead on every node due to the decoding of the TM.	Yes, due to the online scheduling and TM encoding performed by the M-Server, and on every node due to the TM decoding at the N-Servers.
Overhead on network	Yes, one protocol message: The cyclic transmission of RM. Also, bandwidth is lost due to the enforcement of TDMA time slots.	Yes, one protocol message: The cyclic transmission of TM. Also here bandwidth is lost due to the fixed size ECs. Moreover, the length of the EC affects the periodicity of the EC and in turn the overhead caused by FTT-CAN on the network.	Yes, two protocol messages: The cyclic transmission of TM and STOP. Also here, as with FTT-CAN, the length of the EC affects the overhead.
Offline flexibility	High, as message transmissions can be optimised into time slots.	High. Some messages can be considered static, for which pre run-time guarantees can be given.	High, as all message streams can be encapsulated into N-Servers providing bandwidth isolation.
Online flexibility	No.	Yes, FTT-CAN provides admission control for dynamic adding and removing of message transmissions in the static window.	Yes, Server-CAN provides full control over message transmissions using an admission control for dynamic adding, changing and removing of N-Servers. Also, the online scheduling allows for implementation of advanced bandwidth sharing algorithms for an adaptive behaviour.
Facilitating subsystem integration	Yes, as TDMA provides temporal partitioning, hence separating subsystems in time avoiding interference.	Yes, although the static window of FTT-CAN offers a lower temporal resolution compared with TT-CAN, due to the scheduling of messages in ECs.	Yes, due to the same reasons as for FTT-CAN.

Table 2. Comparative assessments of approaches decoupling the CAN ID from the CAN PRIO.

native CAN) is done with regards to the transmission of messages online. Also, the result of the optimisation can be degraded if some system does not allow for re-assignment of identifiers (due to, e.g., the usage of legacy and/or proprietary systems).

When conforming to one of the standards presented in Section 3.4 the situation is better from an integration point of view. Of course, this requires that all subsystems follow the same standard, and it might not always be possible to enforce such a requirement.

Instead, we argue for decoupling the message identifier from the message priority, since it removes a great obstacle in the context of subsystem integration: A change in an identifier to make it system-unique would not affect the scheduling. Such decoupling is partially or totally achieved using TT-CAN, FTT-CAN or Server-CAN. The differences among these techniques are outlined in Table 2.

TT-CAN is a pure TDMA approach, allowing for subsystem integration, although limiting the online flexibility of the system. FTT-CAN performs the scheduling online. However, the decoupling of message identifier and message priority is only valid for the synchronous window of

an EC. It is possible to configure FTT-CAN such to only have a synchronous window, although there is always some bandwidth potentially lost due to the fixed size ECs and the lack of a bandwidth reclamation mechanism. The usage of Server-CAN, on the other hand, completely decouples the message identifier from the message priority, by scheduling all messages according to EDF using server-based scheduling techniques. Also, bandwidth is efficiently reclaimed thanks to the dynamic length ECs. Hence, greatest flexibility is provided using Server-CAN. However, all three approaches come at a cost in terms of protocol overhead: The cost of using Server-CAN is evaluated in [15]. Also, note that TT-CAN and the synchronous usage of FTT-CAN are optimised for periodic traffic whereas Server-CAN allows for both periodic and aperiodic traffic. Here, the two former provides less jitter for periodic traffic compared with Server-CAN, but do not have the same support for aperiodics.

6 Summary

This paper argues for the importance of decoupling CAN IDs from CAN PRIOs in the context of subsystem integration. Three CAN schedulers that allow for this feature are presented, where we particularly argue for Server-CAN as a strong candidate due to its combination of flexibility and full decoupling of CAN ID from CAN PRIO.

Acknowledgements

This work was supported by the ARTIST2 Network of Excellence on Embedded Systems, IST-004527, under the Adaptive Real-Time Cluster, and by the Swedish Foundation for Strategic Research (SSF) via the research programme PROGRESS.

References

[1] L. Almeida, P. Pedreiras, and J. A. Fonseca. The FTT-CAN protocol: Why and how. *IEEE Transaction on Industrial Electronics*, 49(6):1189–1201, December 2002.

[2] R. J. Bril, J. J. Lukkien, R. I. Davis, and A. Burns. Message response time analysis for ideal Controller Area Network (CAN) refuted. In J.-D. Decotignie, editor, *Proceedings of the 5^{th} International Workshop on Real-Time Networks (RTN'06) in conjunction with the 18^{th} Euromicro International Conference on Real-Time Systems (ECRTS'06)*, Dresden, Germany, July 2006.

[3] L. Casparsson, A. Rajnák, K. W. Tindell, and P. Malmberg. Volcano - a revolution in on-board communication. *Volvo Technology Report 98-12-10*, 1998.

[4] G. Cena and A. Valenzano. An improved CAN fieldbus for industrial applications. *IEEE Transaction on Industrial Electronics*, 44(4):553–564, August 1997.

[5] CiA. CANopen communication profile for industrial systems, based on CAL. CiA Draft Standard 301, rev 3.0, October, 1996. http://www.canopen.org.

[6] M. Di Natale. Scheduling the CAN bus with earliest deadline techniques. In *Proceedings of the 21^{st} IEEE International Real-Time Systems Symposium (RTSS'00)*, pages 259–268, Orlando, FL, USA, November 2000. IEEE Computer Society.

[7] ISO 11783-1. Tractors and machinery for agriculture and forestry – serial control and communications data network – part 1: General standard for mobile data communication. *International Standards Organisation (ISO)*, ISO Standard-11783-1, 2005.

[8] ISO 11898. Road vehicles – interchange of digital information – Controller Area Network (CAN) for high-speed communication. *International Standards Organisation (ISO)*, ISO Standard-11898, November 1993.

[9] ISO 11898-4. Road vehicles – Controller Area Network (CAN) – part 4: Time-triggered communication. *International Standards Organisation (ISO)*, ISO Standard-11898-4, December 2000.

[10] H. Kopetz, R. Obermaisser, P. Peti, and N. Suri. From a federated to an integrated architecture for dependable embedded real-time systems. Technical Report 22, Technische Universität at Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2004.

[11] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):40–61, January 1973.

[12] M. A. Livani and J. Kaiser. EDF consensus on CAN bus access for dynamic real-time applications. In J. Rolim, editor, *Proceedings of the IPPS/SPDP Workshops*, volume Lecture Notes in Computer Science (LNCS-1388), pages 1088–1097, Orlando, Florida, USA, March 1998. Springer-Verlag.

[13] L. Lo Bello and O. Mirabella. Randomization-based approaches for dynamic priority scheduling of aperiodic messages on a CAN network. In *LCTES '00: Proceedings of the ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Embedded Systems*, pages 1–18, London, UK, 2001. Springer-Verlag.

[14] NMEA 2000® Standard. The National Marine Electronics Association. http://www.nmea.org.

[15] T. Nolte. *Share-Driven Scheduling of Embedded Networks*. PhD thesis, Department of Computer and Science and Electronics, Mälardalen University, Sweden, May 2006.

[16] 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.

[17] P. Pedreiras and L. Almeida. A practical approach to EDF scheduling on Controller Area Network. In *Proceedings of the IEEE/IEE Real-Time Embedded Systems Workshop (RTES'01) at the 22^{nd} IEEE Real-Time Systems Symposium (RTSS'01)*, London, England, December 2001. Technical report, Department of computer science, University of York, England.

[18] A. Rajnák. Volcano: Enabling correctness by design. In R. Zurawski, editor, *The Industrial Communication Technology Handbook*, pages 32–1 to 32–18. CRC Press, Taylor & Francis Group, 2005.

[19] SAE J1938 Standard. Design/Process Checklist for Vehicle Electronic Systems. *SAE Standards*, May 1998, SAE.

[20] SAE J1939 Standard - The Society of Automotive Engineers (SAE) Truck and Bus Control and Communications Subcommittee. SAE J1939 Standards Collection, 2004, SAE.

[21] K. W. Tindell, H. Hansson, and A. J. Wellings. Analysing real-time communications: Controller Area Network (CAN). In *Proceedings of 15^{th} IEEE International Real-Time Systems Symposium (RTSS'94)*, pages 259–263, San Juan, Puerto Rico, December 1994. IEEE Computer Society.

[22] K. M. Zuberi and K. G. Shin. Non-preemptive scheduling of messages on Controller Area Network for real-time control applications. In *Proceedings of the 1^{st} IEEE Real-Time Technology and Applications Symposium (RTAS'95)*, pages 240–249, Chicago, IL, USA, May 1995. IEEE Computer Society.