An Open Platform for Mixed-Criticality Real-time Ethernet

Gonzalo Carvajal
Department of Electrical Engineering
Universidad de Concepcion
Concepcion, Chile
Email: gcarvaja@udec.cl

Sebastian Fischmeister
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, ON, Canada
Email: sfischme@uwaterloo.ca

***Abstract*—For more than one decade, researchers have considered Ethernet as a natural replacement to legacy fieldbuses in modern distributed applications. However, Ethernet components require special modifications and hardware support to provide strict timing guarantees. In general, the high-cost of deploying hardware components limits the experimental validation of proposed solutions in real-world applications. Despite the vast literature, only a few solutions report real implementations, and they are all closed to the research community, hindering further development for constantly evolving applications.**

This paper introduces *Atacama*, an on-going effort on deploying the first hardware-accelerated and open-source framework for mixed-criticality communication on multi-hop networks. Specialized modules exploit the principles of traditional fieldbus systems to coordinate communication tasks on real-time stations, and can be easily integrated to and coexist with Commercial Off The Shelf (COTS) devices operating with best-effort traffic. Experimental characterization of implemented prototypes report minimal jitter on 1Gbps links, and show that real-time guarantees are resilient to injected best-effort traffic. The framework is available as an open-source project, enabling researchers to verify the results, explore, test, and deploy new networking solutions for modern distributed systems in real-world scenarios.

I. Introduction

Modern distributed real-time applications are constantly growing in the number and complexity of interconnected elements, and then the networking technologies become a key component in systems such as aircrafts, automobiles, and medical devices, among others. Apart from offering high bandwidth, modern networks must also support mixed-criticality communication, providing hard latency guarantees for time-sensitive applications, while still communicating background traffic from best-effort applications [1]–[4].

Low-cost, high-speed, and ubiquitous components make switched Ethernet an attractive networking technology for modern real-time systems. However, standard Ethernet defines a competitive approach to media access aiming to maximize the average throughput, and is unable to provide the strict latency guarantees required in hard real-time applications [5]. Therefore, industry and academia started to explore mechanisms to provide latency guarantees for real-time applications on top of Ethernet [6], [7], which are commonly referred as Real-Time Ethernet (RTE). In general, RTE solutions propose integrating coordination mechanisms in COTS devices to prevent competition between stations.

978-3-9815370-0-0/DATE13/©2013 EDAA

In practice, tight coordination at high-speed is only possible through hardware implementations. The high cost associated to implement custom devices hinders the experimental validation of the proposed mechanisms in real-world applications. Nowadays, despite the vast amount of related literature, only a few solutions report experimental validation, and all of them are closed and then unavailable for researchers.

This work introduces *Atacama*, the first open-source hardware-based RTE framework with support for multi-hop networks. *Atacama* introduces two components to enhance COTS devices and provide tight latency bounds between real-time stations connected through multiple switches: (1) a programmable Application-Specific Instruction-set Processor (ASIP) executing Time-Division Multiple Access (TDMA) schedules to coordinate communication tasks between real-time stations, and (2) a dedicated path in the switches for low latency and low jitter forwarding of real-time frames. Both modules seamlessly integrate to COTS architectures, enabling the coexistence of both real-time and best-effort traffic using traditional Ethernet infrastructure. Our evaluation of functional prototypes shows that the system provides guaranteed tight bounds for real-time frames, even in the presence of best-effort traffic. To the best of our knowledge, this is the first RTE solution that allows researchers to verify the results and build upon, serving as a base for exploring, developing, and testing new networking solutions for real-time applications [8].

The rest of this paper is organized as follows: Section II introduces the concepts and functional principles of the ASIP tailored for real-time communication. Section III describes the custom path for real-time frames inside the switches. Section IV summarizes the experimental evaluation of the prototypes. Finally, Section V concludes the paper.

II. Traffic Coordination in Real-time Stations

Atacama uses the Network Code framework [9] to provide coordinated communication between distributed stations. The framework has three main elements:

- A domain-specific instruction set to describe dynamic TDMA schedules.

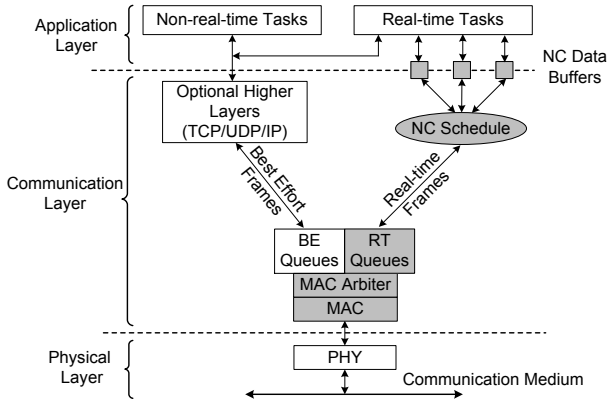

Fig. 1. Layered Model for the Network Code Framework

- A compiler with a verification engine that translates the programs into verified executable schedules.
- The entity that executes the schedules at run time.

This section summarizes the framework as outlined in [9], and describes the architecture of the execution entity.

A. Overview of the Network Code Framework

Fig. 1 shows the model defined for a real-time station. Real-time tasks running in the upper application layer use the services from the programmable communication layer to communicate data within strict timing constraints. Both layers run independently, and they only interact by exchanging data through shared buffers. Real-time tasks produce and consume real-time and best-effort data. Real-time data flows through a custom path (gray blocks in Fig. 1), and their delivery is guaranteed to happen within bounded time. Best-effort data use traditional higher layers and have no delivery guarantee.

Tasks access real-time data by reading/writing predefined buffers at specific times. Predefined schedules implement anisochronous communication rounds, which coordinate the exchange of real-time data using a Reference-Broadcast Synchronization (RBS) scheme [10]. Transmitters receive exclusive access to the medium during specific time slots. On the other end, interested receivers perform the matching operations to extract the application data from incoming frames, and store it into a local data buffer. To ensure conflict-free operation, the schedules must consider a worst-case scenario for the propagation and processing of real-time frames.

Similar to legacy fieldbus systems, Network Code follows a broadcast communication scheme. It also assumes that communication tasks follow well-defined temporal patterns, and all requirements and data structures are known in advance. Under these conditions, the user can apply static verification and analysis [9] to detect potential conflicts before they occur at run time. This is an important property for safety-critical systems requiring evidence-based certification.

B. Network Code ASIP

Fig. 2 shows the general architecture of the Network Code ASIP (NC-ASIP). The design is based on early concepts presented in [11], and have three sections: the memory space, the Ethernet core, and the Network Code Core (NCC).

The memory space and Ethernet core provide the interfaces to the application and the physical layer, respectively. The former contains the data buffers, and the PROG block that holds the programmed schedule. The latter contains the Medium Access Controller (MAC) logic and arbitrates the access to the ports between best-effort and real-time frames.

The NCC implements the main functionality. It features a superscalar architecture, implementing each instruction as an independent finite state machine. The controller fetches and parses the instructions from the PROG block, checking dependencies between consecutive instructions, and triggering concurrent execution whenever it is possible [11]. Each instruction performs a fixed number of steps, making the execution time of a program predictable at design time.

The following paragraphs present a functional description of the instruction blocks in the NCC, which provide control of data flow, timing, and execution flow.

1) Data Flow Control: Transmission of real-time data is driven by create-send sequences. The create instruction moves data from a data buffer to the send buffer. The send instruction encapsulates the data into an Ethernet frame using a special identifier (NC-DATA Type) in the EthType field. For reception, incoming real-time frames automatically trigger the autoreceiver block, which extracts the application data from NC-DATA frames, and stores it in the receive buffer. The data stays in the receive buffer until either a receive instruction moves it to a local application buffer, or a new arriving NC-DATA frame replaces it.

2) Timing Control: The instruction future(L, dl) starts a countdown timer with initial value dl. The halt() instruction stalls the program execution waiting for the expiration of this timer, and then resumes execution of the program at label L.

The sync(mode, dl) block operates in two modes. In master mode, it emits a synchronization frame tagged as NC-SYNC Type. In slave mode, the block stalls the execution of the next instructions until either the arrival of a synchronization frame, or the expiration of a countdown timer with initial value dl.

3) Execution Flow Control: The branch(guard, L) instruction enables the implementation of dynamic schedules that make on-the-fly decisions based on guards. Guards check for specific conditions based on values of buffers, execution history, flags, etc. If the evaluated guard returns TRUE, the schedule will continue execution at the specified label L; otherwise, the schedule will continue with the next instruction. For example, the system can stop transmitting data when a certain variable lies below a predefined threshold.

III. Dedicated Real-time Path in the Switches

This section introduces the dedicated path enabling low-latency and low-jitter forwarding of real-time frames in multi-hop topologies. The gray modules in Fig. 3 implement a queue-free path for coordinated frames inside the switch. The path resembles a traditional bus topology used in legacy fieldbus systems, which naturally spans across multiple switches.

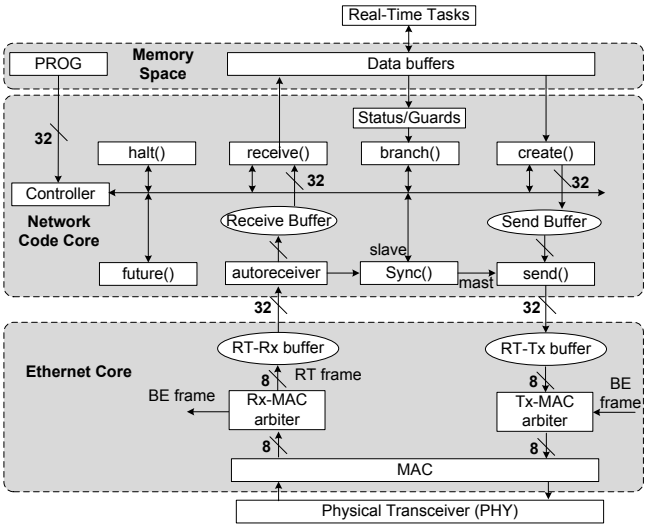

Fig. 2. Architecture of the Network Code ASIP

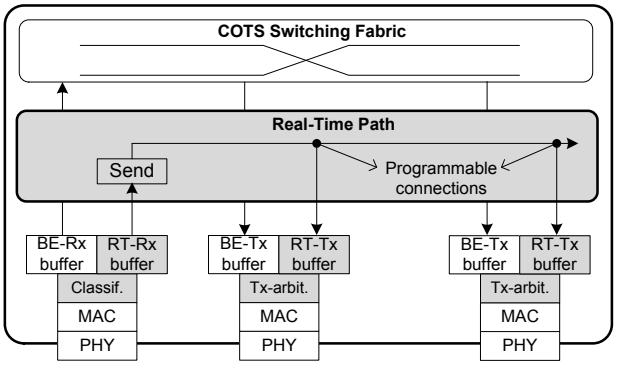

Fig. 3. Enhanced Ethernet Switch

A. Classifying and Forwarding Real-time Frames

Each input port includes a classifier to separate real-time from best-effort frames. The switch parses and stores incoming real-time frames (tagged as NC-DATA or NC-SYNC type) in the corresponding RT-Rx buffer. As soon as the RT-Rx buffer asserts its non-empty flag, the send block starts moving the frame to the dedicated RT-Tx buffer in all the other ports (cut-through forwarding). Considering the broadcast nature and coordination of real-time frames generated from the NC-ASIPs, the dedicated path avoids any address processing and queuing mechanisms to handle contention. This is an essential characteristic to reduce the jitter inside the switch.

Best-effort frames follow the traditional path through the COTS switching fabric. The Tx-arbiter module gives strict priority access to the output ports to real-time frames. Whenever one of the RT-Tx buffers has data to transmit, the arbiter automatically gives it access to the output ports, potentially interrupting ongoing best-effort transmissions. Once emptying the real-time buffer, the arbiter gives access to the output port back to the best-effort buffer. As a result, stations using COTS components communicate transparently, but their effective bandwidth will be reduced according to the amount of real-time traffic flowing through the network. Because real-time transactions are defined in advance, it is possible to estimate the available bandwidth for best-effort traffic beforehand.

IV. Experimental Results

Using the NetFPGA platform [12], we implemented an evaluation kit containing four independent instances of the NC-ASIP described in Section II-B, and a prototype for a four-port enhanced switch that integrates the custom path to the COTS reference switch design available for the NetFPGA [12]. Both devices are able to operate at line rate over 1Gbps links.

This section summarizes experimental results for achieved predictability and robustness of timing behavior using the implemented prototypes.

A. End-to-End Latency

We define the end-to-end latency (EEL) as the time span for moving a byte of application data from a buffer in the origin to the receive buffer in the destination. In *Atacama*, the EEL between real-time stations with N_S switches in the path is:

$$\text{EEL} = \text{Tx}_{\text{ASIP}} + \text{Rx}_{\text{ASIP}} + N_S\text{Sw} + (N_S + 1)\text{L} \qquad (1)$$

where

- Tx_{ASIP}: defined for create-send sequences. Time span since triggering of the create block until the first byte of application data leaves the MAC.
- Sw: switch forwarding time since the first byte of data arrives to the MAC in the input port until it leaves the MAC in the output port.
- Rx_{ASIP}: time span since the first byte of data reaches the MAC in the destination until stored in the receive buffer.
- L: effects of PHYs and cables linking two devices. Time span since the first byte of data leaves the MAC in the origin port until it reaches the MAC in the other end.

Using the Xilinx's Chipscope tool, we characterized the worst-case for each term in (1) as the highest observed value in a set of 100 samples for each case (we omit the details due to limited space). We observed that, in each case, the difference between the highest and lowest value in the set was 16 [ns], which we attribute to the drift between the different clock domains in the devices and the sampling clock. The latency on the physical links must be characterized according to the specific configuration. For this experiment, we considered two Broadcom BCM5464SR PHYs linked with 7.62m CAT6 cables. Using the observed values, we model the EEL for a variable of length v_l (in 32-bits words) in *Atacama* as:

$$\text{EEL}(N_S, v_l) = 1.272 + 1.576N_S + 0.032v_l[\mu s] \qquad (2)$$

To verify the model, we measured the latency between two instances in the NC-ASIP platform exchanging real-time frames with three switches in the path. We additionally connected three ports of an IXIA traffic generator injecting broadcast best-effort frames at 800Mbps each. Table I compares the highest observed latency from a set of 100 samples for each

TABLE I
WORST-CASE END-TO-END LATENCY

v_l [words]	**model** [μs]	**observed** [μs]	**error**
20	6.64	6.608	0.48 %
35	7.12	7.088	0.45 %
70	8.24	8.224	0.20 %

case to the worst-case value obtained from (2). As we see, even under the highly saturated scenario for best-effort traffic, the derived model provides a tight and effective upper-bound for the latency of real-time frames.

B. Robustness Against Co-existing Best-effort Traffic

To illustrate the robustness of the real-time guarantees, we used an IXIA traffic generator to collect long-term statistics over controlled streams of Ethernet frames. One port of the generator emits two periodical reference streams of timestamped frames. One stream generates frames of configurable length tagged as NC-DATA, with a gap of 100µs between consecutive frames. The other stream is similar, but the frames are tagged as best-effort type. These streams propagate through three switches to a second port in the generator which collects the instantaneous propagation latency for each frame class. Two additional ports in the generator broadcast best-effort frames of 1,000B at a rate of 470Mbps each. This configuration generates an aggregated bandwidth close to over-utilization, and allows us to stress the effects of jitter in the latency of the different classes.

Fig. 4 shows the observed latency versus the transmission rate on the timestamped streams. Latency measurements are only valid when there are no lost frames between the transmitter and receiver. The markers show the average latency over a sample set of 10^7 frames for each type. The ends of the bars represent the minimum and maximum value on each set. On the one hand, we see that the latency and jitter of the reference best-effort stream increases with the transmission rate. When the timestamped best-effort stream raises over 50Mbps, the total traffic in the network exceeds the capacity of the links, and the reference best-effort stream starts reporting dropped frames. On the other hand, the real-time path provides a bounded latency and no losses for real-time frames, independent of the best-effort traffic flowing through the network. The *cut-through* forwarding implemented in the real-time path also reduces the propagation latency in relation to the traditional *store-and-forward* used in most COTS switches.

V. CONCLUSIONS

This work introduced *Atacama*, the first open-source hardware-accelerated RTE framework with support for mixed-criticality traffic in multi-hop topologies. The framework features an ASIP that coordinates communication tasks in real-time stations using a TDMA approach, and dual-pathways in switches to provide hard latency guarantees for real-time frames even with coexisting best-effort traffic. Using implemented prototypes, we derived a model that provides a tight upper-bound for the communication latency between distributed real-time stations, only limited by physical characteristics such as the drift between different clock domains and uncertainty in the physical links. The experiments showed that latency guarantees are robust to the injection of best-effort traffic. *Atacama* is the first experimental framework openly available, allowing researchers to validate the results, build upon, and evaluate their solutions in real-world applications.

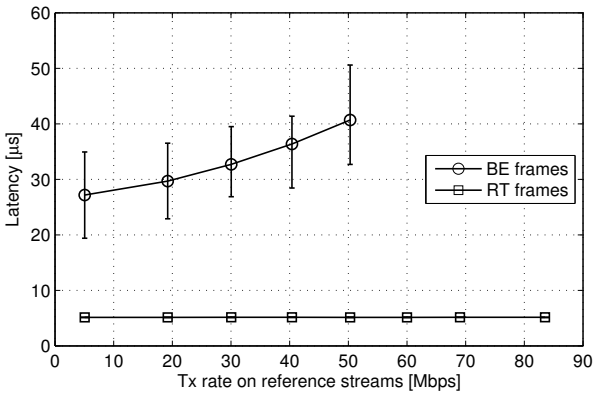

Fig. 4. Robustness of Real-time Latency

ACKNOWLEDGMENT

This was partially supported was supported in part by programs MECESUP and CONICYT from the Chilean government, and projects NSERC DG 357121-2008, ORF-RE03-045, ORF-RE04-036, ORF-RE04-039, APCPJ 386797-09, CFI 20314 and CMC, ISOP IS09-06-037, and the corresponding industrial partners associated.

REFERENCES

[1] R. Maier, "Event-Triggered Communication on Top of Time-Triggered Architecture," in *Proc. of the 21st Digital Avionics Systems Conference*, vol. 2, 2002, pp. 13C5–1 – 13C5–9.
[2] H. Kopetz, "Event-Triggered versus Time-Triggered Real-Time Systems," in *Operating Systems of the 90s and Beyond*, 1991, pp. 87–101.
[3] R. Zurawski, *The Industrial Information Technology Handbook*. CRC Press, 2005.
[4] M. Lukasiewycz, S. Chakraborty, and P. Milbredt, "FlexRay Switch Scheduling: A Networking Concept for Electric Vehicles," in *Design, Automation Test in Europe (DATE)*, 2011, pp. 1 –6.
[5] R. Seifert and J. Edwards, *The All-New Switch Book: The Complete Guide to LAN Switching Technology*, 2nd ed. Wiley Publishing, 2008.
[6] J.-D. Decotignie, "The Many Faces of Industrial Ethernet," *IEEE Industrial Electronics Magazine*, vol. 3, no. 1, pp. 8 –19, march 2009.
[7] M. Felser, "Real-Time Ethernet - Industry Prospective," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1118–1129, Jun. 2005.
[8] "Embedded Software Group, University of Waterloo," http://esg.uwaterloo.ca.
[9] S. Fischmeister, O. Sokolsky, and I. Lee, "A Verifiable Language for Programming Communication Schedules," *IEEE Trans. Comput.*, vol. 56, no. 11, pp. 1505–1519, nov 2007.
[10] J. Elson, L. Girod, and D. Estrin, "Fine-grained Network Time Synchronization Using Reference Broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, Dec. 2002.
[11] S. Fischmeister, R. Trausmuth, and I. Lee, "Hardware Acceleration for Conditional State-Based Communication Scheduling on Real-Time Ethernet," *IEEE Trans. Ind. Informat.*, vol. 5, no. 3, pp. 325–337, aug 2009.
[12] "Official NetFPGA project webpage," http://www.netfpga.org.