TOWARDS A REAL-TIME COMMUNICATION NETWORK FOR AUTONOMOUS RAIL VEHICLES

André Luiz de Freitas Francisco, Bernd Schulz, Christian Henke

University of Paderborn – RtM
Pohlweg 98, D-33098 Paderborn, Germany
Tel:. +49 5251 605576, Fax: + 49 5251 605579,
Email: Andre.Francisco@RtM.upb.de

University of Paderborn – LEA
Warburger Str. 100, D-33098 Paderborn, Germany
Tel:. +49 5251 605486, Fax: + 49 5251 605483,
Email: schulz@lea.upb.de, henke@lea.upb.de

Abstract: This paper presents a communication network architecture aimed at supporting the operation of autonomous rail vehicles. The implicit dynamism of this application imposes the design of special mechanisms to guarantee real-time and fault-tolerant data communication, even when logical networks are dynamically created, destroyed or reconfigured. To cope with this requirement, a new protocol is used to build up dynamic communication channels, which are mapped in a hierarchical physical network. The propulsion of the vehicles is based on linear motors placed along the rail track and, for this reason, the cars must not only communicate with each other, but also with the linear motors they are passing by.

Keywords: Real-time Communication, Mechatronic Systems, Hardware Design

1. INTRODUCTION

Mechatronic systems are gradually becoming larger and more complex. Moreover, especially for applications in the automotive, transportation or aerospace domains, there is increasingly the necessity to interconnect a large amount of components in order to realize the required functions in a distributed manner. For this reason, data communication systems play an

important role in the design of new mechatronic systems and must be carefully designed to guarantee real-time and possibly fault-tolerant behavior.

In this paper we deal with the data communication requirements for the RailCab project [7], which is a train system that consists of autonomous vehicles, also called shuttles. Linear motors placed along the rail track and installed on the vehicles are used for propulsion and braking. This technology also allows shuttles to build up or leave vehicle-convoys while moving. To implement such features, a real-time and fault-tolerant communication system to support the management of dynamic logical networks is required. However, the requirements of data communication systems for conventional trains are different [4], because they are mainly meant for traffic control and not necessarily for establishing links between several trains. In addition, the need for the vehicles to control linear motors sections on the track, to decide with other shuttles which convoy strategies to take and to control velocity and distance with respect to other vehicles are all factors that lead to a higher bandwidth utilization.

This paper is organized as follows. In section 2 we briefly introduce a communication protocol to support the functions previously described. Section 3 presents the data network infrastructure concept for the RailCab track and in section 4 we show how dynamic networks can be implemented using the new protocol. Finally we present the hardware used to test the new components and some concluding remarks.

2. THE TRAILCABLE PROTOCOL

The communication system described in this paper can be implemented using the TrailCable communication protocol, which is described in more details in [3]. A network based on this protocol is built up with point-to-point, full-duplex communication links made up of electrical cables or fiber optics. Every node can have one or more ports, and for each port a unique link to a neighbor can be established. Once such a physical infrastructure is available, it is possible to create logical channels as indicated by the example in figure 1.

Each node dynamically assigns the available data links to currently active real-time communication tasks by means of a scheduling algorithm such as the Earliest Deadline First (EDF) [1]. To guarantee real-time behavior, before a new communication task can be admitted into the network an acceptance test must be executed based on the characteristics of the real-time channels, such as arrival rate, length and deadline of the data packets. Thus,

the logical communication channels are limited by factors such as link bandwidth, maximum transmission latency or the number of available message identifiers. The scheduling operation and the routing of frames are controlled by data structures stored locally in each node, the so called *scheduling tables.*

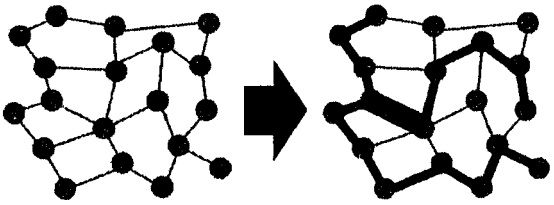

Fig. 1 – A set of communication channels

The TrailCable protocol also implements special mechanisms to cope with fault-tolerance of the real-time data channels, including redundancy of the data links and bandwidth guardians to prevent that traffic overload (that can only be caused by faults) interfere with the correct scheduling of the remaining tasks.

The choice to use the TrailCable is justified by the fact that managing dynamic networks under real-time conditions can be supported by this new protocol, as will be shown in the following sections.

3. NETWORK INFRASTRUCTURE

Before looking in details how dynamic networks are built up, it is necessary to present the communication infrastructure to be used. To construct the physical network for the RailCab track we initially assume that three main components must be interconnected: The servos that control the linear motors, the Shuttle/Track Communication Points (STCP) and the section manager. Their functions are the following:

- *Servo* – equipment that controls the linear motors based on the following reference values: frequency f and current I of the desired stator wave, position x and length l of the convoys or a single vehicle.
- *STCP* – because different technologies can be used for transmitting data between the rail track and the moving shuttles, we use the so called STCPs to abstract the components that are responsible for establishing this wireless communication.
- *Section managers* – are computers that collect the information of all shuttles in a certain track section, such as position and velocity. The data

are then used for different functions, ranging from distributing the reference values to the linear motor servos to supporting the convoy formation phase.

A test-track (fig. 2) of about 530m (scale 1:2,5) for the RailCab project was built at the University of Paderborn and is used to evaluate different subsystems. We start by showing how the current network infrastructure looks like and suggest afterwards a new concept that can be scaled up to larger systems.

Fig. 2 – The RailCab test-track

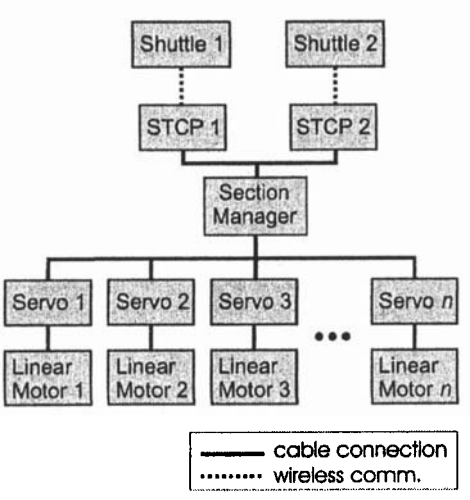

Fig. 3 – Current test-track infrastructure

The current infrastructure of the test-track [6] consists of 83 linear motor sections that are installed along the rail and the respective servos. One section manager controls all servos via a CAN bus interface and is also connected to two STCPs (radio modems), one for each of the two vehicles that can travel simultaneously on the test-track. An architecture representation of this system is depicted in figure 3.

Although the presented architecture meets the current test-track communication requirements, it is clear that such a centralized structure can not be scaled up for larger systems as a lot of problems would arise. In the typical application scenarios, the nodes (servos, STCPs and servo managers) are linearly distributed over great distances on the track and therefore factors like communication latency, cabling effort, required bandwidth, electrical interferences, fault-tolerance, among others, prohibit a centralized approach.

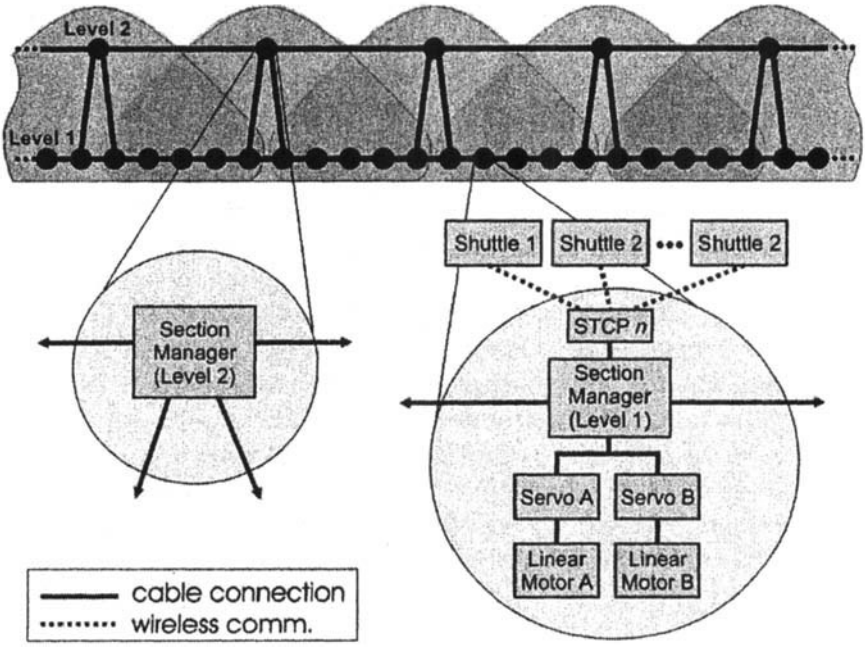

Fig. 4 – The new network concept

To build up a scalable network, the topology shown in figure 4 can be employed. Each node at the lowest level consists of a section manager, a STCP and two linear motors with their respective servos. Via the STCP, the shuttles currently traveling over the correspondent track section can communicate with the fixed network infrastructure. The real-time data communication channels passing through different section managers can be

mapped in the hierarchical network using the same procedure that is used in figure 1.

For the sake of clarity, the connections in the figure 4 are represented as simple point-to-point links, but the communication topology can be constructed to tolerate link errors, as shown if figure 5. In this scenario, even if a single link fails, the system is able to continue working properly as the data interfaces are replicated. The topology in figure 5 is therefore robust enough to support a limited number of not consecutive node failures, because at least one alternative route would still be available.

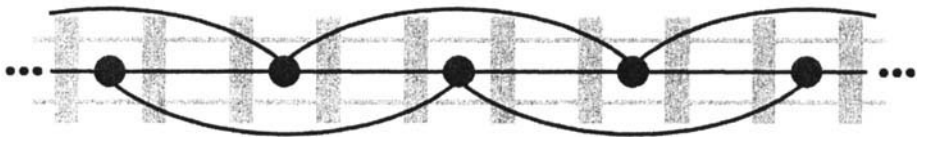

Fig. 5 – A redundant topology

It can also be seen in figure 4 that a higher communication level made up of interconnected section managers (level 2) exists. The objective of this second level is not to overload the primary communication path by transmitting packets node by node over great distances. Should this necessity exist, the data packets are transmitted from a section manager of level 1 to the correspondent section managers of level 2, which then forward the message via the upper "shortcut" links to the destination region. Once the message is received by the respective section managers of level 2, it is directed to the final node (section manager of level 1).

The reason why the section managers build up overlapping regions (fig. 4) is to cope with fault-tolerance. Should one of the section managers of level 2 fail, a second one is still available to assume the functionalities of the faulty neighbor. Another advantage of the presented architecture is that it can be extended by means of additional hierarchy levels if bandwidth or other communication requirements should increase.

4. DYNAMIC NETWORKS

Now that the basic network topology was presented, it is possible to show how the communication channels between different shuttles are maintained when the vehicles are moving. During normal operation, communication channels can be reconfigured, added, and removed dynamically. A possible scenario, which consists of three shuttles, is depicted in figure 6a. The nodes A to F build up the "level 1" segment identified in the figure 4. For the sake of simplicity in fig. 6a, we will consider only the data flow from left to the right, but the concept also applies

for the inverted direction. It can be seen in figure 6a that every participating node has the same *scheduling table,* where the *ID* 1 indicates that the information received by a node was generated at its direct neighbor, the *ID* 2 at a neighbor one hop away and so on. This is possible due to the fact that when a data packet is forwarded by a node, the *ID* of the message is replaced by a new one.

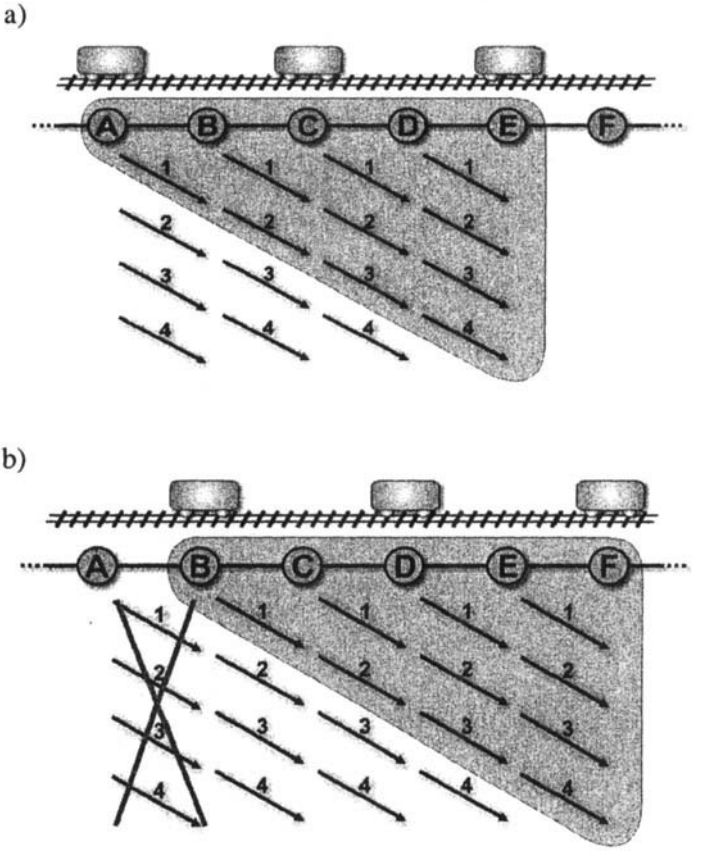

Fig. 6 - Maintenance of dynamic channels

When shuttles keep moving along the track and change their position (figure 6b), it is not necessary to alter all *scheduling tables* at all nodes. Instead, the reconfiguration process takes place only at the channel ends. The link from node A to node B is not needed anymore as soon as the left-most shuttle has reached the position B. Therefore, node A becomes idle and its scheduling and routing operations can be partially disabled. From this moment, node B turns into the new origin. At the channel end, the communication channel must be expanded by activating node F with the same *scheduling table* used by the other participants.

The employment of reusable IDs instead of exclusive addresses represents an elegant manner to cope with the described "moving" logical networks, because the management of the communication links can be done by means of similar data structures in the different nodes.

In the given example, a logical network consists of 5 nodes, but the concept can be also extended for larger clusters, by including additional entries in the *scheduling tables*.

5. INITIAL IMPLEMENTATION

To test the concepts described throughout this paper on the test track, three electronic boards have been developed. The first one is adapted from [8] and consists of a PowerPC MPC555 microcontroller, an FPGA, an Ethernet interface, and four LVDS ports. The second one is a bridge from the TrailCable protocol to the interface used by the motor servos [2]. The third board couples the electrical LVDS interface to fiber optics (fig. 7).

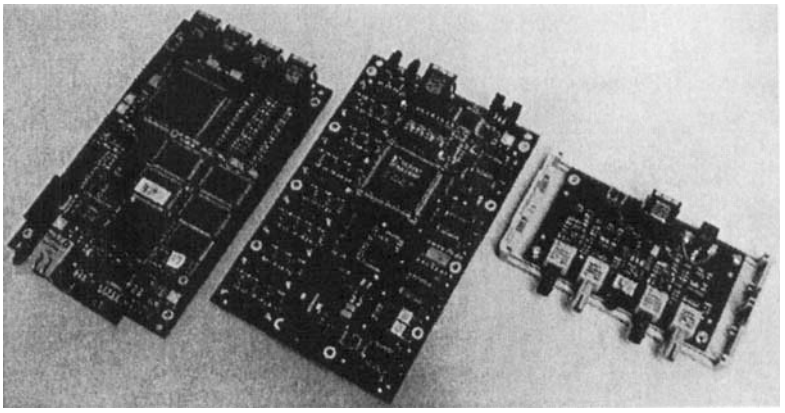

Fig. 7 – Evaluation Hardware

The TrailCable protocol is being entirely implemented in FPGAs, using the VHDL programming language and the Xilinx ISE 7.1 development suite. Data rates up to 32Mbps for links of about 100 meters (using the fiber optics adapter) can be reached using the current hardware. The size of the data packets is initially restricted to 128 bytes and the packet forwarding operation in a node takes less than 1μs. In order to construct the communication protocol in a resource efficient manner, special components of the FPGA architectures, such as distributed and block memories are used to reduce the area needed by the protocol functions in the programmable

logic devices. This is done because resource efficiency is also a factor that should be taken into account when building embedded systems.

6. CONCLUSION

A concept to build up the data communication network of a new train system was presented in this paper. It is based on the TrailCable protocol, which allows real-time data channels to be dynamically reconfigured, besides providing fault-tolerance capabilities.

Using the approach presented, it is possible to create overlapping logical networks that are used for two purposes: to create a hierarchical topology for organizing the data flow and to establish "moving" logical subnets. In the latter case, the shuttles within a certain area are able to communicate with each other, abstracting the fact that they are moving.

Due to the physical behavior of the system, shuttles do not need to communicate with all others over a long track, but mainly with their neighbors. For this reason, the proposed architecture still allows the scalability of the system, what can be done, for example, by adding extra hierarchy levels in the network physical structure.

Finally, to implement and validate the concepts described in this paper, a new hardware platform was specially constructed and an optimized architecture for the TrailCable protocol is being developed to allow a resource efficient implementation for the network to be tested at the RailCab test-track.

REFERENCES

[1] Buttazzo, G. "*Hard Real-Time Computing Systems – Predictable Scheduling Algorithms and Applications*" Kluwer Academic Publishers, Boston/Dordrecht/ London, 1997.

[2] de Freitas Francisco, A. L.; Rettberg, A.; Hennig, A.: *Hardware Design and Protocol Specification for the Control and Communication within a Mechatronic System.* In: Proceedings of IFIP Working Conference on Distributed and Parallel Embedded Systems (DIPES'04). Toulouse, France. Kluwer Academic Publishers, 23-26 August 2004

[3] de Freitas Francisco, A. L.; Rammig, F. J.: *Fault-Tolerant Hard-Real-Time Communication of Dynamically Reconfigurable, Distributed Embedded Systems.* In: 8th IEEE International Symposium on Object-oriented Real-time distributed Computing. Seattle, Washington, USA. May 18-20, 2005.

[4] European Train Control System (ETCS) Project. Available online at: http://etcs.uic.asso.fr/.

[5] Kandlur, D., Shin, K. G. & Ferrari, D. *"Real-Time communication in point-to-point networks"*. IEEE Trans. on Parallel and Distributed Systems, Oct. 1994, pp. 1044-1056.

[6] Pottharst, A.; Schulz, B.; Böcker, J.: *"Kommunikationssysteme einer Anlage mit doppelt gespeistem Linearmotor"*. In: Fachmesse und Kongress (SPS/IPC/DRIVES). Nürnberg, Germany, 2004

[7] University of Paderborn. *The RailCab Project.* Available online at: http://www.railcab.de. 2004.

[8] Zanella, M.; Robrecht M.; Lehmann, T.; Gielow, R.; de Freitas Francisco, A. L.; Horst, A.: *"RABBIT: A Modular Rapid-Prototyping Platform for Distributed Mechatronic Systems"*. In: Proc. of the 14th Symposium on Integrated Circuits and Systems Design; Brasília, Brazil, 2001.