PULSED DATA STREAMS

Hermann Kopetz
Institut für Technische Informatik, Technische Universität Wien, A 1040 Wien, Treitlstrasse 3

Abstract: This paper proposes a new communication primitive for distributed embedded control systems: *the pulsed data stream*. A pulsed data stream is a time-triggered cyclic unidirectional data stream that is transmitted for a short duration at a defined phase of every cycle of a periodic control system. Since the duration of a cycle and the phase and duration of the transmission within each cycle are known a priori, the transmission of pulsed data-streams can be scheduled by the network ahead of their activation in order to minimize the end-to-end latency that is provided to an application. This paper introduces the concept of a pulsed data stream, presents a number of practical examples that demonstrate the utility of pulsed data streams in distributed monitoring and control applications and hints at implementation issues of pulsed data streams in local and wide-area networks.

Keywords: embedded system, real-time, distributed system, distributed control.

1. INTRODUCTION

The widespread availability of powerful, low-cost and dependable silicon devices is dramatically impacting the field of distributed control in embedded systems. The integration of MEMS sensing and actuating elements with a micro-controller on a single silicon die makes it possible to build an intelligent sensor/actuator node that performs front-end signal conditioning and calibration directly at the sensor and present the sensor data in a uniform digital format via a standardized communication protocol to a remote processing component that executes the control algorithm. Real-time communication issues have thus entered distributed control applications at the lowest control-loop level. The temporal performance of the communication protocols within a control loop has thus a direct influence on the quality of control.

Large control systems are organized in a hierarchical manner where control loops are closed at multiple levels. Today it is common practice that different communication protocols are used at the different levels[1]. For example, while at the lowest level field bus protocols, such as CAN[2], are extensively deployed, standard Ethernet is widely used at the higher levels within a given location. In geographically distributed control systems, such as *in power grid control*, standard Internet protocols, such as TCP [3], are deployed. At any level the temporal characteristic of the communication protocol, such as latency and jitter, are determining parameters for the quality of the overall control system.

In this paper a single new communication primitive, the *pulsed data stream*, for the exchange of real-time data in distributed real-time control systems is proposed. *Pulsed data streams* can be used at all levels of the control hierarchy, from low-level local sensor communication up to wide-area distributed control systems, such as the control of the power-grid[4]. *Pulsed data streams* form a uniform mechanism for the exchange of real-time data among the components of a distributed control system.

This paper is organized as follows. In Section two we analyze the communication requirements of distributed control system. Section three introduces the concept of a *pulsed data stream* as a new communication primitive for distributed control systems at all levels of the control hierarchy. Section four presents a number of application scenarios from the field of distributed control. Section five looks at implementation issues and discusses three different scenarios: networks-on-chip, local-area networks and wide-area networks.

2. TIMING AND COMMUNICATION REQUIREMENTS OF CONTROL SYSTEMS.

In this Section we analyze the timing and communication requirements of local and wide area digital control systems. We are looking at three different tasks of a control system: (i) process monitoring, (ii) continuous control, (iii) discrete control.

2.1 PROCESS MONITORING

A *controlled object*, e.g., a car or an industrial plant process, changes its state as a function of time. If we freeze the time, we can describe the current state of the controlled object by recording the values of its state variables at that moment. Possible state variables of a controlled object "car" are the position of the car, the speed of the car, the position of switches on the dash board, and the position of a piston in a cylinder. We are normally not interested in *all* state variables, but only in the *subset* of state variables that is *significant* for our purpose [5]. In a distributed control system, e.g., a nation-wide power grid, care must be taken that the controlled object is observed by all sensors at the same instant of time. A necessary prerequisite for monitoring a large controlled object is thus the availability of a global time of sufficient accuracy at every sensor node. Other than that, monitoring does not require real-time communication protocols that exhibit controlled delay and jitter.

In the Final Report of the US-Canada Task force on the NE American power outage in 2003, the following remarks are contained about synchronization and timing of the collected data [6] p. 162: *A valuable lesson from the August 14 blackout is the importance of having time-synchronized system data recorders. The Task Force's investigators labored over thousands of data items to determine the sequence of events, much like putting together small pieces of a very large puzzle. That process would have been significantly faster and easier if there had been wider use of synchronized data recording devices. . . . Today at a relatively modest cost, all digital fault recorders, digital event recorder and power system disturbance recorders can and should be time-stamped at the point of observation using a Global Positioning System (GPS) synchronizing signal.*

2.2 CONTINUOUS CONTROL

In *continuous control* a continuous process is sampled periodically by a digital system—giving rise to a *hybrid system*. From the viewpoint of the distributed digital control system, the progress of time is partitioned into an (infinite) set of consecutive cycles, where within every cycle the same sequence of periodic activities is performed: waiting for the beginning of the next cycle, sample the inputs at the (often physically dispersed) sensors, send the data to a component that executes the control algorithm, and finally distribute the data to the actuators that act on the physical parameters of the process. The instants, when these activities are started, should be highly predictable [1], p. 19-4: *Periodicity is not mandatory, but often assumed as it leads to simpler algorithms and more stable and secure systems. Most of the algorithms developed with this assumption are very sensitive to period duration variations, jitter in the starting instant. This is especially the case of motor controllers in precision machines. Simultaneous sampling of inputs is also an important stability factor.*

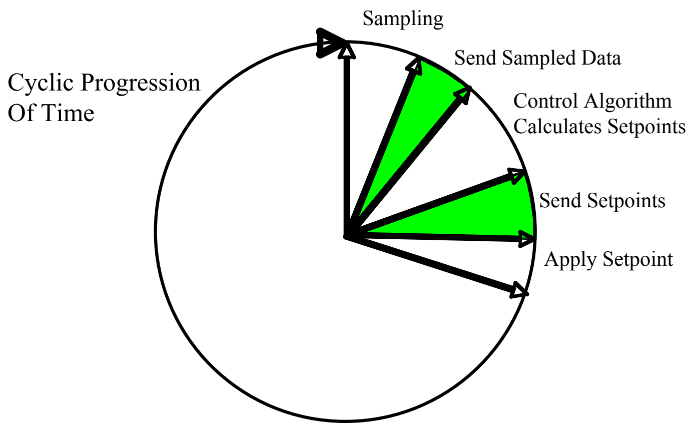
Sampling
Send Sampled Data
Cyclic Progression Of Time
Control Algorithm Calculates Setpoints
Send Setpoints
Apply Setpoint

Fig. 1: Cyclic Model of Time and phases of a control system.

Fig. 1 suggest a *cyclic*, not a *linear* model time. Cyclic systems are very common in control systems and in biological systems [7]. The progress in cyclic systems is inherently *time-triggered*. The progression of time (and activities) within each cycle can be depicted by the concept of *phase*, ranging from 0 degree to 360 degree for the full cycle. The trigger for an activity is thus the instant when the time progresses to a defined phase start point. Fig. 1 depicts five phase start points: the start of sampling, start sending the sampled data, start the execution of the control algorithm, start sending the set-points to the actuators and finally apply the set-points to the controlled object. The interval between the *start of sampling* and the *application of the set-points* to the controlled object should be minimized in order to reduce the dead-time of the control loop and thus increase the quality of control. Within every cycle of activities there are two communication phases: the *sending of input data* to the control algorithm and the *sending of output data* to the actuators as depicted by the shaded areas in Fig. 1. Reducing the duration of these communication phases reduces the dead-time in the control loop and thus increases the achievable quality of control. Since many of the widely used communication protocols, such as CAN, Ethernet and TCP do not minimize delay and jitter, the quality of control in these systems is unnecessarily degraded.

2.3 DISCRETE CONTROL

Discrete control systems are basically event-driven system. A significant state change causes control actions by the computer system. In many of the event-driven systems, the significant state changes are relayed to the computer system by the interrupt mechanism. However, every interrupt driven system is in danger of overload in case the minimum time between interrupts is not bounded. In order to exclude the possibility of overload by design, a number of discrete control systems sample the significant states cyclically and thus convert the discrete control system to a *quasi continuous* system. In such a design care must be taken that the sampling interval is smaller than the smallest duration between correct events that have to be sensed. Such a solution rejects events at the source that are closer together than the specified minimum interval between events and thus protects the control system from overload caused by sensor failure.

3. PULSED DATA STREAMS

Before introducing the pulsed data stream concept, we would like to elaborate on a fundamental conflict in the design of real-time communication protocols.

3.1 FUNDAMENTAL CONFLICTS IN REAL-TIME PROTOCOL DESIGN

Before designing a protocol for the transfer of real-time data between the components of a control system, one has to make a decision between the following two alternatives: (i) the components are assumed to be *competing* with each other for communication bandwidth or (ii) the components are assumed to be *cooperating*. The first alternative is the only alternative for *open system*, where the set of communicating components is not known *a priori*.

Independently of the characteristics of the detailed network protocol, there is always the possibility that in alternative (i)--*competition of the components*--two or even *n* components try to start sending messages to the same receiver at the same instant. Since only one message can be received at a sequential receiving port at an instant, there is the possibility that a conflict among the *n* components will occur. The two possible ways to resolve this conflict are: (a) either one message is sent immediately and the other *n-1* messages *are* ***stored*** *in the network* and sent to the receiver sometimes later when the link to the receiver is free again (this is the solution of *switched Ethernet*) or (b) the network *exercises* ***dynamic*** *back-pressure flow control* to the *n-1* components that did not succeed in sending their message (this is the solution of *CAN* or *bus-based Ethernet*). Both alternatives are *unsatisfactory* from the point of view of the real-time properties of the protocol, such as *delay* and *jitter*, because the *worst-case transmission time* for a message depends on the momentary traffic generated by *all* components and is much larger than the *minimal transmission time,* when no other component is active. Furthermore, the jitter becomes a *global property* that depends on the behavior of all components, violating some principles of composability [8]. Jitter-sensitive applications, such as control applications and high-quality multi-media applications, do have problems with a communication infra-structure that introduces significant jitter [9].

In alternative (ii), a *closed-world system,* the number of clients is known *a priori*. The clients *cooperate* with each other or with a central scheduler in order to establish a coordinated schedule, such that the communication system is in the position to meet the requests of all clients within specified temporal bounds. Temporal guarantees for *all* time-critical messages can only be given in a *closed-world system*.

3.2 THE PULSED DATA STREAM CONCEPT

The *pulsed data stream* is a new communication concept for closed-world distributed real-time systems. It assumes that a set of *a priori* known components that are aware of a global notion of time agree on a cooperative cyclic communication schedule, such that all transmission requests of all partners can be satisfied without conflict.

A *pulsed data stream* is a cyclic data stream that transports data uni-directionally in *pulses* from *one* sender to *n a priori* identified receivers at a specified phase of the cycle for a specified duration. We call the duration between the *start of transmission* and the *termination of transmission* at the sender, i.e., the duration of the *pulse* at the sender, the *active phase at the sender*. We call the duration between the *start of reception* and the *termination of reception* at a receiver the *active phase at the receiver*. Pulsed data streams are *bursty*, but the bursts are not sporadic, but regular and known in advance. This common knowledge is instrumental for the scheduling of the pulsed data stream by the network.

The *pulse* at the sender is characterized by the following parameters

- the set of receivers
- the duration of the cycle
- the start-instant of the *pulse*, expressed in the metric of global time
- the termination-instant of the pulse
- the reliability class
- security class

We call this data set the *pulse characterization*. The amount of data that can be transmitted during a pulse at the sender depends on the transmission bandwidth of the sender to the network.

Before the start of a pulsed data stream, the network must be informed about the *pulse characterization* in order that the required resources can be reserved. Depending on the characteristics of the network (size, bandwidth, internal protocol structure) and the current commitments to other pulsed data streams, the network will respond with the performance parameters that can be guaranteed for the transport of this newly requested pulsed data stream: the *transmission delay*, i.e., the interval between the *start of the active phase* at the sender and the start of the *active phase at a receiver* and the duration of the delivery phase. The *transmission delay* can be very small in an on-chip-network (in the order of nanoseconds) and substantial in a wide area network (in the range of hundreds of milliseconds, or even seconds). The *transmission delay* depends on the propagation delay of the channels and the scheduling strategies within the network.

In many cases the data transmitted within a pulse is *state data*, i.e., it informs about the current value of state variables. At the receiver, a new version of the state data replaces the previous version (update in place). State data is not consumed by a receiver (similar to a variable in computer memory). There are no queues needed to handle state data. In many control applications, the loss of a single pulse is not critical, because the receiver will automatically take the state at the previous cycle as a replacement for the lost pulse. The retransmission of a corrupted pulse by the network, which will impact the timing of the delivery, is thus not a standard option in pulsed data streams. The fact that corrupted pulse data does not have to be retransmitted has deep implications for the design of the network protocols.

Fig. 1 depicts two pulsed data streams within a typical control loop. Since a properly designed local area real-time communication network has a very small *transmission delay*, it is not depicted in Fig. 1. The most important property of the pulsed data stream concept concerns the *a priori* knowledge about the phase of the recurring send instants. This a *priori* knowledge must be used by the communication system to plan for the provision of the predictable transport service for the pulsed data streams within a network.

4. APPLICATION OF PULSED DATA STREAMS

In this Section we sketch a number of examples for the use of pulsed data streams in distributed control systems.

4.1 REAL-TIME MONITORING

Many real-time monitoring applications require the periodic transmission of sensor values to a central process monitoring facility. Two parameters are critical for the proper operation of this application: the synchronized sampling of the data and a small delay between the sense instant and the delivery instant of the sensor data at the monitoring facility. The *a priori knowledge* about the parameters of the pulsed data streams enables the communication system to plan and reserve the required transmission capacity for the pulsed data streams.

4.2 CLOSED LOOP CONTROL

A typical local control loop has the cycle structure of Figure 1. Two pulsed data streams, the *pulsed data stream* from the sensors to the component that executes the control algorithm and the *pulsed data stream* from the control-algorithm component to the actuators must be supported by the communication system.

In a wide area control scenario, such as power-grid control, a number of different networks are involved in the transport of the real-time data from the sensor components to the control center. The a-priori knowledge of the parameters of the pulsed data streams makes it possible to align the transmission phases of the different networks *a priori*, such that the overall latency of the real-time data transmission can be optimized.

4.3 TRIPLE MODULAR REDUNDANCY

In a TMR (triple modular redundant) system three components have to exchange *pulsed data streams* in order to vote on the results of the computations and the inner state of the components. These *pulsed data streams* can be coordinated *a priori* to minimize the latency and jitter that is introduced by the transport service.

4.4 MULTI-MEDIA SYSTEMS

In some multi-media applications a complete frame has to be transmitted from one component to another component before the frame processing can be initiated. This puts a high bursty load on the communication system that can be mitigated if the regularity of the *a priori* known parameters of the pulsed data stream is taken into account in scheduling the communication.

5. IMPLEMENTATION HINTS

The implementation of *pulsed data streams* requires the establishment of a system-wide global time base and a scheduler that coordinates the pulsed data streams requested by the involved components. The establishment of a global time of known precision is a well-understood problem that has been implemented in a number commercial applications [10-13]. Communication systems that are controlled by time-triggered communication protocols provide an ideal environment for the implementation of pulsed data streams. The scheduling problems is substantially simplified if the cycle durations are harmonic, e.g., positive powers of a smallest units. External synchronization can be eased if one of the cycle durations is exactly the duration of the physical second.

5.1 ON-CHIP NETWORKS

The communication network within a deeply embedded multi-computer SoC, such as the Cell chip [14], can be implemented as a time-triggered network. Such an on-chip time-triggered network can be expected to support a very high bandwidth (the bandwidth of the interconnect on the cell chip is more than 100 Gbits/second). The implementation of pulsed data streams is trivial if such a time-triggered network is available as a network-on-chip (NoC). In order to be able to synchronize the pulsed data streams within a chip with off-chip networks, an external synchronization of the chip-internal time base must be provided.

5.2 LOCAL AREA NETWORKS

TT-(time-triggered) Ethernet [13] provides an ideal environment for the implementation of pulsed-data stream in a local area context. TT Ethernet distinguishes between two traffic categories, i.e. open-world ET (event-triggered) Ethernet traffic and the closed-world time-triggered traffic. ET traffic is handled in full compliance with the Ethernet standard [15], while TT traffic is coordinated by a scheduler and transported with a-priori known constant latency and minimal jitter. For a detailed description of TT Ethernet refer to [13].

5.3 WIDE AREA NETWORKS

The implementation of *pulsed data streams* in wide-area networks, such as the Internet is a research challenge, since the present Internet is driven by flexible dynamic routing algorithms that are optimized for average performance, but do not support real-time guarantees that are required in real-time control systems.

6. CONCLUSION

Pulsed data-streams are a new communication primitive for the exchange of real-time data among the components of a distributed control system. A pulsed data stream is a powerful abstraction that is an exact fit with the requirements of many real-time control applications.

7. ACKNOWLEDGEMENTS

This work has been supported in part by the EU integrated research project DECOS and by the NoE Artist II.

REFERENCES

[1] Decotignie, J., D., *Which Network for Which Application*, in *The Industrial Communication Technology Handbook*, R. Zuwarski, Editor. 2005, Taylor and Francis: Boca Raton. p. 19/1-19/15.

[2] CAN, *Controller Area Network CAN, an In-Vehicle Serial Communication Protocol*, in *SAE Handbook 1992*. 1990, SAE Press. p. 20.341-20.355.

[3] Furrer, F., J, *Industrieautomation mit Ethernet TCP/IP and Web -Technologie*. 2003, Heidelberg: Hüthig Verlag.

[4] Birman, K.P., et al., *Overcoming Communication Challenges in Software for Monitoring and Controlling Power Systems*. Proc. of the IEEE, 2005. **93**(5): p. 1028-1041.

[5] Kopetz, H., *Real-Time Systems, Design Principles for Distributed Embedded Applications; ISBN: 0-7923-9894-7, Seventh printing 2003*. 1997, Boston: Kluwer Academic Publishers.

[6] Force, U.-C.T., *Final Report on the August 14, 2003 Blackout in the United States and Canada*. 2004.

[7] Winfree, A.T., *The Geometry of Biological Time*. 2001: Springer Verlag New York.

[8] Kopetz, H. and N. Suri. *Compositional Design of Real-Time System: A Conceptual Basis for the Specification of Linking Interfaces*. in *ISORC 2003--The 6th International Symposium on Object Oriented Real-Time Computing*. 2003. Hakodate, Japan: IEEE Press.

[9] Reinder, J., et al. *Dynamic Behaviour of Consumer Multimedia Terminals: System Aspects*. in *IEEE International Conference on Multimedia*. 2001: IEEE Press.

[10] Kopetz, H. and W. Ochsenreiter, *Clock Synchronisation in Distributed Real-Time Systems*. IEEE Trans. Computers, 1987. **36**(8): p. 933-940.

[11] Kopetz, H., *Specification of the TTP/C Protocol*. 1999, TTTech, A 1040 Wien, Schönbrunnerstraße 13.

[12] IEEE, *1588 Standard for a Precision Clock Synchronization Protocol for Network Measurement and Control Systems*. 2002.

[13] Kopetz, H., et al. *The Design of TT Ethernet*. in *ISORC 2005*. 2005. Seattle: IEEE Press.

[14] Wang, D.T., Proc. of the *ISSCC 2005: The CELL Microprocessor*. 2005.

[15] Ethernet, *IEEE Ethernet Standard 802.3 at URL: http://standards.ieee.org*. 2002.