The Time-Triggered Ethernet (TTE) Design

Hermann Kopetz Astrit Ademaj Petr Grillinger Klaus Steinhammer
Vienna University of Technology
Real-Time Systems Group
Treitlstr. 3/182-1, A-1040 Vienna, Austria
email:{hk,ademaj,grilling,klaus}@vmars.tuwien.ac.at

Abstract

This paper presents the rational for and an outline of the design of a time-triggered (TT) Ethernet that unifies real-time and non-real-time traffic into a single coherent communication architecture. TT Ethernet is intended to support all types of applications, from simple data acquisition systems, to multimedia systems up to the most demanding safety-critical real-time control systems which require a fault-tolerant communication service that must be certified. TT Ethernet distinguishes between two traffic categories: the standard event-triggered Ethernet traffic and the time-triggered traffic that is temporally guaranteed. The event triggered traffic in TT Ethernet is handled in conformance with the existing Ethernet standards of the IEEE. The design of TT Ethernet has been driven by the requirement of certification of safety-critical configurations and an uncompromising stand with respect to the integration of legacy applications and legacy Ethernet hardware.

Key Words*: real-time, safety-critical, multimedia, communication, Ethernet, fault-tolerant, communication architecture.*

1 Introduction

During the past decades, Ethernet has established itself as the most successful local area network of the world. Due to its open nature–which is one reason for its success–it is difficult to guarantee strict temporal properties in standard Ethernet-based systems. Therefore, many researchers have looked into the problem of extending Ethernet such that it can be deployed in applications where temporal properties are important. This paper presents the design of a time-triggered Ethernet that tries to achieve this ambitious goal.

More than ten years ago, our group has developed the time-triggered protocol TTP [17] that has had some impact on the real-time community. At present TTP/C proprietary controller chips support a bandwidth of up to 25 Mbits/second. During the research project NEXT-TTA we implemented TTP/C on a COTS Gigabit Ethernet. Due to the unpredictable nature of the COTS Ethernet switch we could only achieve a disappointing resource utilization [23, 6]. Based on the experience of this project and of industrial TTP applications in the aerospace domain, the railway domain, and the automotive domain we have looked into the possibility of providing a novel unified communication architecture based on Ethernet that meets the requirements of all types of non-real-time, real-time, and multimedia applications up to the most demanding safety-critical real-time products, while still maintaining full compatibility with the existing Ethernet standard. TT-Ethernet can be considered to be a unification of the best properties of *standard Ethernet* and *TTP/C*.

From the point of view of development of the semiconductor technology in recent years, the unification of the communication architectures for real-time and non-real-time applications is imperative. The ever-increasing costs of chip-mask development, software-tool development, maintenance, and training put an enormous economic pressure on specialized communication solutions for niche markets that require a proprietary hardware and software base. A generic communication architecture for real-time and non-real-time applications supported by mass-produced commercial-off-the-shelf hardware and software components –like the standardized Ethernet of today–would significantly reduce the cost of and provide a platform for a multitude of new applications.

In the past decade many designs [3, 10] have been published that try to extend Ethernet to the domain of real-time systems. There exists even a web-site [9] that informs about a number of these designs. It would be futile to develop just another design for a real-time Ethernet if we cannot eliminate some of the identified weak spots of the already published designs. In our opinion the most significant weak spot of the published versions of *real-time Ethernet* is their *missing determinism*. Without determinism it is difficult to support fault-tolerance for masking the failure of a component by triple modular redundancy (TMR)

in a safety-critical application. Deterministic operation is also a precondition for certification [4]. If the important class of safety-critical real-time applications is not covered by a design, then the vision of a unified communication architecture for all types of real-time applications cannot be maintained.

This paper is organized as follows. In the subsequent Section two we discuss the requirements that were at the starting point for our design. In Section three we refresh some basic concepts that are important for understanding time-triggered real-time systems. Section four gives an overview of our design for a time-triggered Ethernet. Section five presents the time-triggered (TT) Ethernet that is intended for non safety-critical applications, such as multimedia applications where a stream of data has to be transported predictably. Section six is devoted to the most demanding safety-critical configuration of TT Ethernet. The paper finishes with a conclusion in Section seven.

2 REQUIREMENTS

The *purpose* of TT Ethernet is to provide a seamless communication system for all types of distributed non-real-time and real-time applications, from very simple uncritical data acquisition tasks, to multimedia systems and up to safety-critical control applications, such as *fly-by-wire* or *drive-by wire*. It should be possible to upgrade an application from standard TT- Ethernet to a fault-tolerant configuration with minimal changes to the application software.

Certification

If TT Ethernet is to be deployed in all types of real-time applications, up to safety-critical applications, then there is the requirement for the *certification of the design*. This is the most demanding requirement from the point of view of design of any computer system. Certification entails that it must be possible to establish the correct operation of the communication system in all specified fault- and load scenarios [2]. It requires the careful partitioning of a system into independent fault-containment regions (FCR), the provision of a detailed fault hypothesis of each FCR and a set of convincing arguments that the design is capable to handle all specified faults. The independence of fault-containment regions [13] and the deterministic behavior of a system [4] are the central pillars of certification. Deterministic systems are also much easier to validate than non-deterministic systems.

Certifyability is the most important design driver of TT Ethernet. In our opinion, it is difficult to certify a system that has not been *designed for certification*.

Legacy Integration

It is the intention that TT Ethernet provides *predictable real-time capabilities* within the constraints imposed by the IEEE Ethernet standard. This requires that existing Ethernet applications can be ported to TT Ethernet without any modification in software and hardware, as long as the *additional services* provided by TT Ethernet, are not utilized. Uncompromising legacy integration is thus a further key requirement for the market acceptance of TT Ethernet.

Predictable and Deterministic Message Transfer

Dependable real-time control applications require a temporally predictable and deterministic (as defined in Section 3) communication system. It is most important that the delay of messages is small and the jitter of the transport system is minimized.

Fault-Tolerant Global Time

The operation of any time-triggered system depends on the availability of a global time base. Any loss of the time base is tantamount to system failure. In safety critical applications the establishment of a fault-tolerant time-base of *a priori* known precision is thus a definite requirement.

Strong Fault Isolation

The physical structure of a safety-critical computer system is determined, to a significant degree, by the requirement of strong fault isolation. If *physical proximity faults* (e.g. the physical destruction of a component in an accident) are to be tolerated, then the computer system must be distributed in space and it must be assured that the destruction of any one site will not cause the other sites that have not been directly impacted by the fault to suffer.

Another requirement in this category is the elimination of error propagation. If a fault at a site causes the site to produce erroneous messages it must be assured that these erroneous message will not propagate to correct nodes and corrupt their state.

Consistent Diagnosis

In a distributed safety-critical system it is important that all correct nodes agree at all times on which node is functional and which node has failed. This is important from the point of view of consistent operation, reconfiguration, and recovery. An asymmetric failure of a node can lead to differing views of the failing node by the other nodes. An appropriate architecture inherent membership algorithm [22] must reconcile these differing views.

Scalability

TT Ethernet must be scalable. There should be no design decision in TT Ethernet that makes it difficult to extend TT Ethernet to higher speeds–e.g. 10 Gigabit/sec, or restricts the number of the controllers in the system.

3 BASIC CONCEPTS

Agreement on concepts is a prerequisite for achieving *understanding*. In this section we therefore introduce the basic concepts that are needed to describe the world of time-triggered (TT) systems in comparison to the world of event-triggered (ET) systems, such as the world of the familiar Ethernet.

In an event-triggered (ET) system an action (e.g., the sending of a message or the execution of a task) is started whenever a *significant non-temporal event* happens either *outside* of the computer (in the environment) or *inside* the computer. Examples of significant non-temporal events are: the occurrence of a *noteworthy state change in the environment* that is communicated to the computer by the *interrupt mechanism* or the termination of a preceding task within the computer.

In a time-triggered (TT) system an action is started whenever a *significant temporal event* happens *inside* the (distributed) computer system. A significant temporal event occurs whenever the monotonically progressing real time reaches a predefined instant, we call it a *trigger value*. Central to the understanding of a TT system is the concept of a *sequence of instants* (for short a *period*), i.e., a *sequence of recurring trigger values* where the next trigger value *appears regularly* after the elapse of a constant interval from the previous trigger value. A period is characterized by its *constant duration* (the interval between any two trigger values of the sequence) and by the *phase* with respect to a reference time. An example for the specification a period is: the train arrives every hour (*period*) ten minutes after the full hour (*phase*).

The fundamental difference between an event-triggered and a time-triggered system relates to the *origin* of the control signals [5]. In a *time-triggered system* control always resides inside the distributed computer system. From the point of view of control, a time-triggered system is a *physically closed deterministic system*. In an event-triggered system control signals can originate *within the computer system* and *external to the computer system* from the environment (e.g., relayed by the interrupt mechanism). An unpredictable environment will thus result in a *non-deterministic behavior* of the computer system. For a more philosophical treatment of the issues of determinism, the reader is referred to *Of Clouds and Clocks* by K. Popper [21].

3.1 Determinism of a Transmission Channel

In safety critical real-time applications, the distributed computer system must deliver its dependable service despite the failure of a node or the loss of a communication channel. Such a dependable service requires the provision of replicated (redundant) resources and a voting mechanisms. *Transparent replication* will only work, if all replicated resources, the nodes and the communication system, exhibit *timely and deterministic* behavior. A *timely and deterministic* multicast transmission channel for real-time applications is defined by the following three properties [14]:

1. Given that a message is sent at the *send instant* t_{send} then the *receive instants* $t_{receive}$ at all receivers of the (multicast) message will be in the interval $(t_{send} + d_{min}, t_{send} + d_{max})$, where d_{min} is called the *minimum delay* and d_{max} is called the *maximum delay*. The difference $d_{max} - d_{min}$ is called the *jitter* of the transmission channel. d_{max} and d_{min} are *a priori* known characteristic parameters of the given transmission channel.

2. The *receive order* of the messages is same as the send order. The send order among all messages is established by the temporal order of the send instants of the messages as observed by an omniscient outside observer.

3. If the *send instants* of n (where $n > 1$) messages are the same, then an order of the n messages will be established in an a priori known manner.

Property (1) assures the timeliness, i.e., that all receivers will receive a message within a bounded latency. Property (2) assures that all receivers will receive messages in the same temporal order, the *send order*. This is important in a TMR system, where multiple receivers must receive the same ordered sequence of messages from multiple senders over different *independent* channels. Property (3) assures that conflicts, which result from *simultaneity*, are resolved in an *a priori* predictable way and that messages that are transported by independent channels (needed to make the system tolerant with respect to a channel failure) will be received in the same order on all channels at all the receivers. It is most difficult to satisfy property (3).

From the above properties it follows that two *independent*, *deterministic* and *timely transmission* channels that operate correctly will always deliver the presented messages at all receivers in the same *receive order* within predefined intervals. Such a timely and identical receive order of the messages is required for the transparent implementation of triple-modular-redundancy (TMR).

3.2 Open-world versus Closed-world Systems

We define an *open-world system* as a system where an *(unknown) number of uncoordinated* clients *compete* for the services of a server. The *critical instant* [18] in an open-world system occurs, when all clients request the services of the server simultaneously. Guaranteed real-time performance cannot be achieved in an open-world system. An example of an open-world system is the Internet.

In a *closed-world system* the number of clients is limited and known *a priori*. The clients can *cooperate* with each other or with a central scheduler in order to establish a coordinated schedule, such that the server is in the position to meet the requests of all clients within specified temporal bounds. Temporal guarantees can only be given in a closed-world system.

Standard Ethernet [7] is an *open world system*. It is thus impossible to establish temporal guarantees in standard Ethernet without restricting the access pattern that is characteristic for open world systems. A number of *real-time Ethernet* proposals try achieve an improvement in the real-time performance by limiting the traffic that may be generated by the Ethernet controllers, e.g., by *traffic policing* as in AFDX [3].

The TT Ethernet design proposed in this paper follows a different route. TT Ethernet distinguishes between two classes of *fundamentally different* traffic categories: *standard Ethernet (ET) traffic* and *TT Ethernet traffic*. Temporal guarantees are only given for the TT Ethernet traffic, but not for the ET Ethernet traffic. Since the TT Ethernet traffic is planned by a scheduler, no conflict between TT Ethernet messages will arise in a fault-free TT Ethernet system. If an (uncoordinated) standard Ethernet message comes into conflict with a TT Ethernet message, then the TT Ethernet switch will preempt the transmission of the standard message in order to be able to transmit the TT Ethernet message within an *a priori* established constant delay. After the transmission of the TT Ethernet message has been completed, the switch will retransmit the preempted standard Ethernet message autonomously.

3.3 ET Messages vs. TT Messages

In our work on distributed real-time systems we have found it useful to make a fundamental distinction between event-triggered (TT) messages and time-triggered (TT) messages or state messages. Table 1 summarizes relevant properties of event messages versus state messages. For a more detailed discussion see [12]. Normal Ethernet messages are event-messages, while TT Ethernet messages are state messages. This has far reaching consequences for the handling of these messages at the sender and receiver and on the temporal properties of these messages.

3.4 Global Sparse Time

A time-triggered system must contain a global time base which is available at all nodes of the distributed computer system. Such a global time base can be established by the periodic synchronization of the local clocks of the nodes by an internal synchronization algorithm [12].

Assume a set of significant events that are of interest in a particular context. This set could be the events of sending and receiving messages. If these events are allowed to occur at any instant of the timeline, then we call the time base *dense*. If the occurrence of these events is restricted to some active intervals of duration π, with an interval of silence of duration Δ between any *two active intervals*, then we call the time base π/Δ *-sparse*, or simply *sparse* for short (Figure 1) [11]. If a system is based on a sparse time base, there are time intervals during which no significant event is allowed to occur. In a system with a sparse time base it is possible to map the active intervals, and thus the time values, into the set of integers and establish the temporal relationship between events by integer arithmetic on the time values. The decision, whether two events that occur on a sparse time base are simultaneous is thus reduced to the comparison of integer number (not real-numbers, as in a *dense* time-base) which gives the same result at all nodes of the distributed system.

It is evident that the occurrences of events can only be restricted if the given system has the authority to control these events, i.e., these events are in the *sphere of control* of the computer system [5]. The send-instants of TT messages in TT-Ethernet are within the control of the distributed computer system and are must occur during the active intervals of a sparse time base.

The occurrence of events outside the sphere of control of the computer system cannot be restricted. In general it must be assumed that these external events are based on a dense time base. In a distributed system with a dense time-base the question of whether two events occur *simultaneously*

Characteristic:	Event Messsage:	State Messsage:
Information Type	Event Information	State Information
Temporal Pattern	Sporadic	Periodic
Semantics	Exactly Once	At least Once
Sender Access	Add to queue	Overwrite memory
Receiver Access	Take from queue	Read from memory
Synchronization	Tight	Loose
Message Rate	Unknown	Constant
Flow Control	Explicit	Implicit
Error Detection	At Sender	At Receiver
Loss of Message	Loss of State Sync	Loss of Period
Control Pattern	Bidirectional	Unidirectional
Multicast Topology	Diffi cult	Easy
Jitter	Signifi cant	Minimal

Table 1. Event vs State Messages

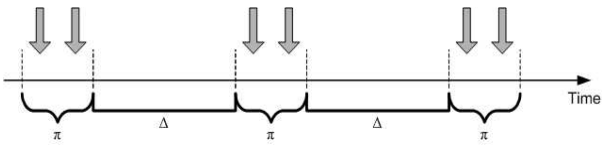

Figure 1. Sparse time-base

cannot be answered consistently (i.e. that all nodes always come to the same conclusion) without the execution of an *agreement protocol*. The execution of such an agreement protocol consumes real-time, requires additional bandwidth and compromises the independence of the involved entities. Nevertheless, the execution of an agreement protocol is required at the boundary between a subsystem with a dense time base and a subsystem with a sparse time base.

3.4.1 Time Format

TT Ethernet is based on a *uniform time format* for all configurations. A digital time format can be characterized by the two parameters, *granularity* and *horizon*. The *granularity* determines the minimum interval between two adjacent ticks of a digital clock, i.e., the smallest interval that can be measured with this time format. The *horizon* determines the instant when the time will wrap around. The time format of TT Ethernet (see Figure 2) is a 64-bit binary time-format that is based on the physical second. Fractions of a second are represented as 24 negative powers of two (down to about 60 nanoseconds), and full seconds are presented in 40 positive powers of two (up to about 30 000 years).

This time format is closely related to the time-format of the GPS (General Positioning System) time, but has a much wider horizon (about 30 000 years) in order to avoid the *wrap around problem* in the foreseeable future. This time-format has been standardized by the OMG in the small transducer interface standard [20]. Since the representation of an instant is exactly eight bytes long (5 bytes denoting the full second, and three bytes denoting the fraction of a second), every instant from January 1980 to about 30 000 years in the future can be represented uniquely within this time format with a granularity of about 60 nanoseconds. The transformation of this time-format to the local wall-clock time can be realized by a Gregorian Calendar Function.

3.4.2 Periods

Time-triggered traffic is *inherently periodic*. A static or dynamic scheduler must plan the conflict-free periods for each message a priori. The TT Ethernet design restricts the period durations of most time-triggered messages to the positive and negative powers of two of the second, i.e. a period

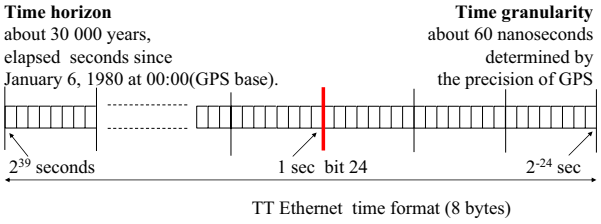

Figure 2. TT Ethernet Time Format

can be either 1 second, 2 seconds, 4 seconds, and so forth, or 1/2 second, 1/4 second, 1/8 second an so forth. Given the time format depicted in Figure 2, there are theoretically at most 64 different period durations in TT Ethernet–corresponding to the 64 different bits in the time format. The duration of each period can thus be characterized by the corresponding bit in the binary time format of Figure 3. We call this bit the *period bit*. In the prototype implementation of TT Ethernet we provide a choice out of 16 different periods, e.g, from 2 seconds to about 60 μseconds and thus require 4 bits to encode the *period bit*. The *phase* of a *period*, i.e. the *offset* to the start instant of the selected duration in the global time format, is designated by the specification of a pattern of twelve bits to the right of the *period bit* (the bits to the right of these twelve *phase bits*, if any, are always *not set*). TT Ethernet thus needs two bytes for the specification of the period of a periodic time-triggered message. We call these two bytes the *period identifier* (or *period-ID*, for short).

Figure 3 specifies a period of $1/2^4$ (i.e 1/16)second with a phase (i.e. the offset from the periodic 1/16 second instant) of $1/2^6 + 1/2^{11} = 16113$ μseconds.

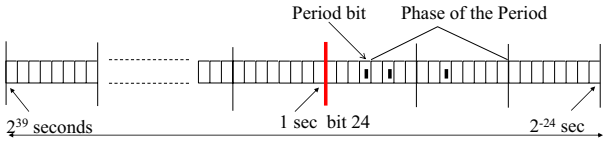

Figure 3. Example of a period specification

4 OVERVIEW OF TT ETHERNET

100 Mbit Ethernet assumes a switched architecture (Figure 4), where every node consists of a host computer and an Ethernet controller that is connected to a *store-and-forward* switch by a bidirectional point-to-point link. Such a structure–we call it a cluster–is typical for present-day Ethernet systems. We distinguish between two types of TT-Ethernet configurations:

standard configuration with standard Ethernet controllers, TT Ethernet controllers, and a single switch (left side of Figure 4)

fault-tolerant configuration with a safety-critical TT Ethernet controller containing two ports to two independent switches (right side of Figure 4).

In the fault-tolerant configuration the operation of the two switches is monitored by two independent guardians (see Section 6.2). We can build a *fault tolerant configuration* by using additional *TT Ethernet* (non safety-critical) and *standard Ethernet controllers*. We note that in such

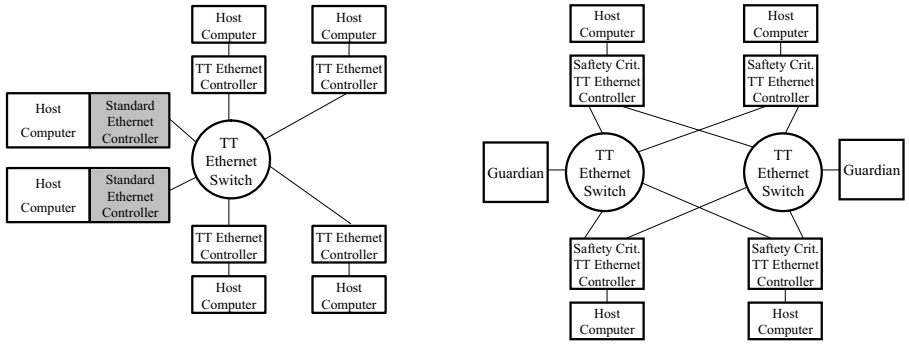

Figure 4. Standard TT Ethernet (left) and safety-critical TT Ethernet (right) configuration

configuration the non safety-critical *TT Ethernet controllers* and *standard Ethernet controllers* communicate using only one channel. In this configuration any component (controller, switch, guardian, links) can fail in a arbitrary way without an impact on the operation of the safety-critical part of the system (safety-critical TT Ethernet controllers, links, switch and guardian).

4.1 Design Rational

4.1.1 Seamless Communication Architecture

TT Ethernet is intended to provide a single communication architecture that covers the whole spectrum of real-time applications, from simple non-critical data acquisition or multimedia systems to demanding safety-critical control applications. Since error detection depends on the availability of independent redundant information about the state of the system, a safety critical application will require more resources than a non-safety critical application, where the cost of these additional resources cannot be justified easily. We introduce the fault-tolerance mechanism in a modular fashion, such that a prospective user has the choice to select those fault-tolerant mechanisms that are best suited to meet the requirements of her/his application in a cost-effective manner.

If a user starts with a standard TT Ethernet configuration she/he should have the assurance that the system can be upgraded to a fully fault-tolerant configuration without any major change in the application software.

4.1.2 Preemption of ET Messages

The key idea of TT Ethernet concerns the distinction between two different classes of traffic:

- The flexible standard event-triggered (ET) Ethernet traffic that originates in the *uncontrollable open world* (see Section 3) and cannot meet strict temporal requirements. The switch handles this traffic according to the store-and-forward paradigm with a best-effort delay as specified by the Ethernet standard.
- The *closed-world* time-triggered (TT) traffic that is coordinated by a scheduler and therefore free of conflicts. This traffic is handled by the switch according to the *cut-through* paradigm and is relayed to all receivers within an *a priori* known constant delay and minimal jitter.

Whenever there is a run-time conflict between an ET message and a TT message the switch will preempt the ET message and will transmit the TT message with the guaranteed constant delay. The switch will autonomously retransmit any preempted ET message immediately after the transmission of the TT message has terminated.

The Ethernet standards contains a two-byte *Type Field* that specifies the type of an Ethernet message (Figure 5). The *Type Field* provides a context for interpretation of the data field of the frame (protocol identification). In order to assure an ordered development of Ethernet, the contents of this Type Field are administered by the Ethernet standard authority of the IEEE [8]. This standard authority has assigned the value $0x88d7$ as the content of the *Type Field* in order to identify uniquely a TT Ethernet message and the associated message protocol.

4.1.3 Minimal State

The TT Ethernet design has been driven by the goal to maintain *minimal state* in the controllers and in the switches. In case the state in the controller or in the switch gets corrupted by transient fault, the time interval until the state is repaired again should be determinable *a priori*. Since this *state recovery* time is a critical parameter in the dependability analysis of safety-critical systems, this parameter should be configurable during the set-up of a concrete TT Ethernet installation.

4.1.4 No Single Point of Failure

The most demanding fault-tolerant TT Ethernet configurations should tolerate an arbitrary failure of any of its components, i.e., nodes, switches and communication links. There

must not be any single point of failure in such a configuration.

4.1.5 Naming

In TT Ethernet we introduce a naming scheme for messages that supports the identification of a message type (in the literature this is often called a message name) and a message instance. The message type name denotes a sequence of messages of the same type. In TT Ethernet we introduce the two byte period ID (see Section 3) as the message type name. A particular message instance can be identified by the concatenation of the message type name (the period ID) with the send instant of the message.

4.2 Message Format

The TT-Ethernet message format is based on the format of standardized raw Ethernet messages (see Figure 5). As mentioned before a TT Ethernet message contains the value $0x88d7$ in the Ethernet *Type Field*. The first few bytes of the data field of a TT Ethernet message contain an additional TT Message header.

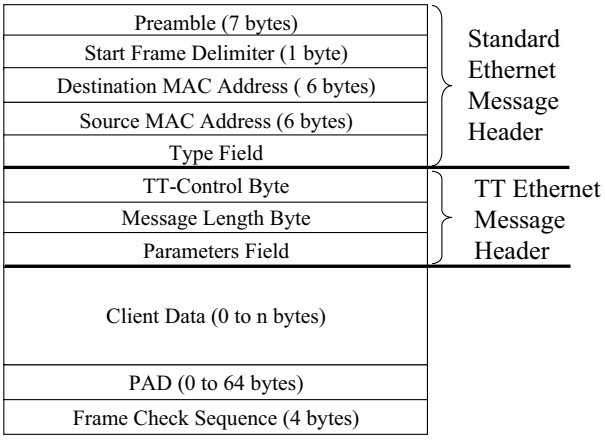

Figure 5. Format of a TT Ethernet Message

4.3 Principles of Operation

In the following subsections, we describe the principles of operation of the TT Ethernet switch and controller.

4.3.1 The TT Ethernet Switch

Based on the contents of the *Type Field* of an incoming message, the switch decides whether an incoming message is a standard Ethernet (ET) message or TT Ethernet message. ET Ethernet messages and TT Ethernet messages are handled differently by the switch.

Arriving standard Ethernet (ET) messages are stored in an ET-message queue of the switch. The message, which is at the end of the message queue, is forwarded to the specified receiver address whenever the outgoing channel to this receiver is free. If an outgoing ET message is in the way of an incoming TT message, then the switch immediately clears the channel (preempts the ET message) for the pending TT message. Immediately after the TT message has terminated, the switch retransmits the (previously preempted) ET message, and, if the transmission was successful, releases the message buffer occupied by this message for a new incoming ET message. This autonomous TT-Ethernet protocol mechanism uses any bandwidth that becomes free for the immediate transmission of ET-messages. It optimizes the throughput without any need for an explicit scheduling action and thus simplifies the schedule design and the system operation.

Arriving TT Ethernet messages are not stored in the switch. They are delayed by the switch for a defined number of μseconds (we call this delay the clearance delay, which is the time needed to clear the transmission path in the case of a preempted ET message) and then forwarded in to the addressed receivers. The clearance delay is a characteristic parameter of a particular switch implementation. In our prototype implementation of a 100 Mbit Ethernet switch it is 4 μseconds.

We recommend to use a broadcast address for TT Ethernet message for the following two reasons:

- In many real-time scenarios the natural topology is broadcast (e.g., a clock synchronization message) or multicast (e.g., a message to three TMR replicas), not point-to-point.
- TT message are sent to the TT-controller and not directly to the host computer. Since COTS (commercial-off-the-shelf) Ethernet controllers do not differentiate between a controller address and the host address, COTS controllers could not read a TT Ethernet message if it were addressed to the controller.

4.3.2 The TT Ethernet Controller

The TT Ethernet controller also treats ET messages and TT messages differently.

Event-triggered (ET) Ethernet messages conform the *Ethernet message format* and are handled according to the *IEEE Ethernet standard*. If a node does not intend to send any TT messages, but only ET messages, a standard off-the-shelf Ethernet controller can be used. In this case the node can send ET messages and receive all incoming ET messages and all incoming TT messages directed to this controller. Since the *message type name* (the *period ID*) is part of every TT message, even a host that is connected

to a COTS Ethernet controller obtains all the information needed to process an incoming TT message.

TT messages have, in addition to the standard message header, a TT message header (see Figure 5) in the data field of the Ethernet message. TT messages are handled by the controller in the following way: For every *new message type* (i.e. a new *period ID* that is not yet known to the controller) received by the controller from the *host* or the *switch*, the controller allocates a *local message-buffer* that will always contain the *current version* of this state message. The received message is *then* stored as the current version in this local message-buffer.

An *outgoing* message, that is a message that is sent from the host to the other nodes, remains in the message-buffer until the instant when the *phase bits* of the global time are identical to the phase bits in the *period ID*. Whenever such an instant occurs the controller sends a copy of the *local message buffer* to the other nodes in the cluster. The contents of the local message buffer remain unchanged until a new version of the message with the same period ID arrives from the *host*. This new version *overwrites* the contents of the *local message buffer* and thus becomes the new current version.

An *incoming* TT message, that is a message that is received by the controller from one of the other nodes of the cluster, overwrites the current contents of the controller local message buffer. The message in the local message buffer of the controller is not destroyed by the reading operation and remains in the buffer until a new incoming message overwrites the current version of this message.

4.3.3 Establishment of the Global Time

A subset of the TT controllers, we call them the *time-keeping controllers*, is responsible for the initial establishment of the global time after start-up and the maintenance of this global time during the operation of the TT cluster. Depending on the required dependability (and cost) of the global time base, there can be between *one* and *five* time-keeping controllers in a cluster.

In a low-cost configuration, there is only one *time-keeping controller*. It acts a time-master that sends periodically a *clock synchronization message* to all other controllers in the cluster. If a *fail-stop failure* of the time-master is to be tolerated, a shadow master can be configured that takes over whenever the current time master has disappeared. In the most dependable configuration, five time-keeping controllers execute a distributed Byzantine resilient clock synchronization algorithm that will tolerate the arbitrary failure of any controller. The configuration of the time-keeping controllers is specified in the data field of the start-up message and the synchronization message. The time-keeping controllers also perform the initial startup.

4.3.4 Host Controller Interaction

In TT Ethernet the host-controller interactions are performed based on standard Ethernet messages. In a TT Ethernet configuration, the host and the controller have two distinct Ethernet addresses. ET messages are sent to the Ethernet address of the host, while TT messages are sent to the Ethernet address of the controller.

Whenever the host intends to send a standard (ET) message, it sends the message in the standard Ethernet way to the Ethernet address of the selected host.

An outgoing TT Ethernet message from the host is addressed to the Ethernet address of the local controller and stored in the *local message-buffer* of the controller as outlined above until the global time reaches the *trigger value* for this message.

An incoming TT message is stored in the *local message buffer* of the controller.

4.4 Message Categories

TT Ethernet distinguishes between the following message categories:

- ET message
- Free-Form TT message
- TT startup message
- TT synchronization message
- Unprotected TT message (UTTM)
- Protected TT message (PTTM)

The bit-pattern $0x88d7$ in the *Type Field* of the standard Ethernet message identifies a message as a *TT Ethernet message*. TT Ethernet messages have an additional header, the TT message header (see Figure 5) that is contained, as far as the Ethernet standard is concerned, in the data field of the standard message. The first byte of the TT message header, the control byte indicates the message category of a TT Ethernet message.

4.4.1 ET Message

ET messages are handled in full conformance with the IEEE Ethernet standard in structure and behavior. The contents of the Type Field identify a message as a standard (ET) Ethernet message in full accordance with the Ethernet standard.

4.4.2 Free Form TT Message

Free Form TT messages (FFTT-messages) are handled exactly like ET message with the only difference that the switch will transport a FFTT message with a constant a priori known delay and minimal jitter in a *cut through* fashion. An ET message that is in the way of this FFTT message will be preempted.

FFTT message are intended for use in systems that do not support clock synchronization and establish the (conflict free) periods for the FFTT messages of a cluster at the application level. FFTT message can use any period durations and are not restricted by the *power-of-two rule* for the period durations.

4.4.3 TT Startup Message

The TT startup messages establish an initial synchronization immediately after startup of the *time-keeping controllers*. After this initial synchronization has been achieved, the *startup phase* is terminated and the clock synchronization algorithm switches to the operational phase, where the established synchronization is maintained by synchronization messages.

4.4.4 TT Synchronization Message

TT synchronization messages maintain the clock synchronization during the operational phase of the TT Ethernet system. A *synchronization block*, i.e., a tight sequence of one to five synchronization messages–corresponding to the one to five time-keeping controllers-is sent periodically. The length of the period between synchronization blocks (*the resynchronization period*) is determined by the quality of the available oscillators and the required precision of the global time and is a parameter of the configuration contained in the start-up message.

4.4.5 Unprotected TT Message (UTTM)

Unprotected TT messages transport user data from a sender to one or a set of receivers within a cluster. As the name implies, unprotected messages are not protected by a guardian. In case of a *non-silent failure* of a node, an UTTM of one node might be corrupted by a TT message from another node. Unprotected messages are intended, among others, for multimedia applications.

Different UTTMs may have different periods, which are conveyed to the controller in the *period ID* that is part of the TT message header. In TT Ethernet, we distinguish between two kinds of UTTMs, the *periodic UTTMs* and the *sporadic UTTMs message*.

4.4.6 Periodic UTTM

The periodic UTTM is a periodic message that is sent over a finite interval, we call it the *period life*, from a sender to the receivers. The *period life* starts with the sending of the first UTTM with this period ID from the host to its controller and terminates after the host has sent an UTTM with the *last-message bit* set (the *last-message bit* is a designated bit in the control byte of the TT header) to the controller, indicating the end of the *period life*. During the period life the controller will send at every instant when the *phase bits* of the global time agree with the *phase bits* of the *message ID* (we call these instants the *send instants* of the message) a copy of the UTTM to its receivers, irrespective of whether the message has been updated by the host between message transmissions.

4.4.7 Sporadic UTTM

The sporadic UTTM is similar to the periodic UTTM with one important difference: a copy of the message is only sent to the receivers if the local message buffer in the controller has been updated by the host since the last *send instant*. Sporadic UTTMs are delivered to the host in the *information push mode*. Sporadic UTTMs have to be scheduled by the scheduler in exactly the same way as periodic UTTMs. Sporadic TT messages consume communication bandwidth only when a new *message-content* must be transmitted. The TT Ethernet switch will use the unconsumed allocated bandwidth of such a message to transmit ET messages out of its local queue.

Sporadic UTTMs are well suited to implement scenarios where a guaranteed temporal response is required in rarely occurring situations, e.g., a *control signal* to activate or stop a service. The period of the unprotected UTTM determines the worst-case guaranteed response time of this message. If a short guaranteed response time, i.e., a short period is selected, then the design space for finding schedules for other TT messages will shrink, but the unused bandwidth will be used by the controller for the transmission of ET messages. The preemptive nature of TT messages makes this reuse possible without any compromise in the timeliness of the sporadic TT message. In a non-preemptive system such a reuse of the bandwidth is not possible, as demonstrated by the adversary argument from Mok [19].

Sporadic UTTMs can also be used to implement an event-message service with implicit flow control. Flow-controlled event messages are needed from the point of view of encapsulation when a federated architecture subsystem is becoming a part of an integrated architecture.

4.4.8 Protected TT Time-Triggered Message (PTTM)

Protected time-triggered messages (PTTM) are introduced in order to provide a communication infrastructure for safety-critical applications. In these applications fault-tolerance must be provided in the communication system such that the arbitrary failure of any single node, the failure of any single switch or the failure of any single channel can be masked without any effect on the availability of the communication service. The services of PTTMs are closely related to the field-proven services of the TTP/C protocol.

Two independent *guardians* (see Figure 4, left side) monitor the operation of the replicated switches of a fault-tolerant TT Ethernet system. The guardians have full information about the schedule of the PTTMs. A guardian

enables the input from a controller to a switch only if the controller is allowed to send a TT message during this time interval. Special mechanism in the guardian are in place to detect and mask slightly-out-of specification (SOS) failures [16].

4.5 The TT Scheduler

The TT scheduler is an *off-line* or *on-line* program at one or more nodes that calculates conflict free schedules for all time-triggered messages of a cluster. The event-triggered messages do not have to be considered, because the logic of the switch ensures that any available bandwidth that is not consumed by the TT messages will be used for the transmission of the ET messages.

In the protected mode, a unique version number is assigned to every new schedule and signed by the scheduler. The version of the PTTMs may not change between the *power-up* and the *shutdown* of a cluster. The guardian checks dynamically whether all PTTMs have the same version number.

4.6 Download and Initialization

Neither the TT Ethernet controller, nor the switch or the guardian contain any initialization (i-state) [12]. These components will accumulate the necessary state immediately after power-up out of the synchronization messages (contains the start-up information) and during the operation out of the headers of the TT messages.

Download of the host software can be realized with standard TCP/IP based file-sharing protocols using the ET messages. The time-interval between power-up and the first transmission of a PTTM message is available for the transmission of ET messages. During this interval, the full bandwidth is offered, e.g., for the download of the core images to the host computers.

5 STANDARD TT ETHERNET SYSTEMS

In this Section, we present two extreme TT Ethernet configurations: (i) a standard TT Ethernet system on side and (ii) a safety-critical TT Ethernet system on the other side. The standard TT Ethernet system is a low-cost configuration that can be deployed in non-safety critical applications, e.g., for multimedia applications. The safety-critical TT Ethernet system provides all services needed to construct a TT Ethernet system that will tolerate an arbitrary failure of any of its components.

Because of the modular structure of the TT Ethernet services, it is possible to construct many intermediate configurations. For example, it is possible to augment a standard configuration with a fault-tolerant clock synchronization service that tolerates the failure of any single clock.

5.1 TT Ethernet Controller

On the hardware side, a TT Ethernet controller has two *ports*, one port to connect to the host computer and the other, an Ethernet port, to connect to the switch. The controller/host combination has two Ethernet addresses, one for ET messages directed to the host and the other for TT messages directed to the controller.

The TT Ethernet controller will support the periodic and sporadic UTTMs, but will not support the PTTMs. It contains sufficient memory to store a single version of every incoming or outgoing TT message type. It will handle ET messages according to the established Ethernet standard.

5.2 Standard TT Ethernet Switch

The standard TT-Ethernet switch stores the incoming ET messages and forwards them to their receiver as soon as the channel from the switch to the receiver becomes free. If, during the transmission of an ET message a TT message arrives, the switch will preempt the outgoing ET messages and will transmit the TT message (in many cases in *broadcast mode*) within an *a priori* known constant delay to receivers in the cluster. After the channels become free again, the switch will retransmit the previously preempted ET message.

5.3 Clock Synchronization and Startup

In its most basic form, clock synchronization is performed by the central master algorithm. In this case, the synchronization block consists of a single message only (the message from the clock synchronization master). The clock synchronization master performs also the startup. Immediately after power up the clock synchronization master starts transmitting start-up-synchronization messages. After a start-up period, the clock synchronization master switches to the transmission of resynchronization messages. The master also performs external clock synchronization.

In case a more dependable time service is desired, a shadow master can be configured that takes over in case the active master fails.

6 SAFETY-CRITICAL TT ETHERNET SYSTEMS

In a safety-critical application the communication system must deliver its service despite the occurrence of failures in the components and the communication channel with a probability of better than one failure in hundred thousand years (the magical 10^{-9} number [25]). In TT Ethernet the protected time-triggered message (PTTM) have been introduced to provide this fault-tolerant service. Those readers who are familiar with the TTP/C system [26] will find

many similarities between the PTTMs and the field-proven design decisions of TTP/C.

6.1 The Safety-Critical TT Ethernet Controller

The controller for a safety critical TT Ethernet system contains three ports, one to the host computer and two Ethernet ports to two independent switches. The TT message header is extended in order to contain a membership vector, the period counter and a signature of the scheduler. The membership vector is calculated according to the same algorithm as in TTP/C [26]. The signature testifies that the schedule has been calculated by an authorized scheduler and has not been mutilated by a malicious controller or host.

6.2 The Guardian

The TT Ethernet switch in a safety-critical configuration is the same as the TT Ethernet switch in standard TT Ethernet. However, the operation of the switch and the connected controllers is monitored by a guardian that forms an independent FCU. The connection between the guardian and the switch consists of an Ethernet port and a set parallel wires that allows the guardian to observe and disable the inputs and outputs of the switch. The guardian has its own fault-tolerant clock synchronization algorithm. The guardian can receive all TT-messages from the switch over the Ethernet port. It is, however only allowed to send ET messages to the switch over this port. An example of such an ET message from the guardian is a diagnostic message informing about the behavior of the controllers. The switch will not accept a TT message that is generated by the guardian. The guardian has knowledge about the schedule of PTTMs and disables the inputs of the point-to-point links to controllers that could possibly interfere with the PTTM schedule. The guardian can perform a semantic analysis of the TT message header and can disable the outputs of the switch in case a semantically incorrect message has been detected. Furthermore, the guardian will monitor that the switch delays an incoming TT messages precisely for the *a priori* established constant delay.

6.3 Fault-Tolerant Clock Synchronization and Startup

Clock synchronization is realized by a combination of a distributed fault-tolerant clock state synchronization algorithm with a central rate correction algorithm [15]. This combined algorithm is also used to perform external clock synchronization. The issues related to a fault-tolerant start-up of a time-triggered system have been thoroughly investigated in the PhD thesis by Steiner [24]. TT-Ethernet uses the fault-tolerant start-up algorithm described in this thesis.

Immediately after startup, a timekeeping controller will send a startup message, which is protected by a signature of the scheduler. The startup message also contains the startup information for the guardian. The guardian will accept this startup information and open the channels for all timekeeping nodes. As soon as the synchronization has been established, the guardian will wait for the first PTTM and store the protected schedule information, which is part of the extended TT message header to monitor the regularity of all succeeding message instances. In case a message does not conform to the established schedule, the guardian will disable the outputs of the switch such that the message is truncated and rejected by all receivers.

In case the guardian experiences a transient fault and loses its state information, it will restart and obtain the current state information from the incoming messages within few TDMA rounds. No explicit state recovery procedure needs to be established.

7 CONCLUSION

We have presented the design for a unified real-time Ethernet that is intended to serve all types of non-real-time and real-time applications, from simple data acquisition tasks, to multimedia application and up to the most demanding fault-tolerant control systems that require certification. The proposed design is fully compatible with the current Ethernet standard and can be characterized by the following principal design decisions: (1) the distinction between two classes of traffic, the open-world *competing traffic* of standard event-triggered Ethernet systems and the closed-world *cooperating traffic* of time-triggered systems and the decision to preempt the event-triggered traffic in case it is in conflict with the time-triggered traffic in order to guarantee the timeliness of the time-triggered traffic, (2) the introduction of a sparse global time base to solve the problem of simultaneity and to achieve a deterministic operation of the communication system, (3) the introduction of an efficient coding schema for *TT message type* names such that the periodic send-instants of the message can be derived from this name and the controller as well as the switch and the guardian can be kept free of *i-state*, (4) the introduction of a new class of time-triggered traffic, the *sporadic time-triggered traffic*, that helps to significantly improve the bandwidth utilization in mixed TT/ET systems, and (5) the support of different upwards compatible configurations such that the tradeoff between cost and dependability can be made in the context of the given application requirements. Because of these clear design principles, we consider TT-Ethernet to be a *simple* and *understandable* system that can be thoroughly analyzed.

The proposed design for a TT Ethernet meets all requirements established in Section two. We are currently implementing the TT Ethernet design on FPGA modules in order to test and evaluate the design decisions. These TT Ethernet

modules are planned to be used in the DECOS (DEpendable COmponents and Systems) project for the provision the communication infrastructure. Since our design does not require any application-specific i-state in any one of the TT Ethernet components, the components do not need any stable storage, which reduces their production cost and eliminates the need for software configuration tools to initialize the controller or the switch.

This TT-Ethernet design represents the latest result of our more than twenty-five yearlong research effort in the design of fault-tolerant distributed real-time systems. The iterative cycle of designing a new artifact, implementing the design, testing and evaluating the implementation, exposing the design and the implementation to critical comments form the scientific and industrial community, and finally improving the design based on the gained insights is in our opinion the only route to attain progress.

Acknowledgment: This work has been supported in part by the Austrian FIT-IT Program for Embedded Systems under Project Number 808197/1729-KA/HN, the European IST project DECOS under project No. IST-511764, and by the European IST Network of Excellence ARTIST2 under project No. IST-004527. The work on the clock synchronization has been supported in part by the Austrian Science Foundation under Project Nr. P 16638-N04.

References

[1] A. Ademaj, H. Sivencrona, G. Bauer, and J. Torin. Evaluation of Fault Handling of the Time-Triggered Architecture with Bus and Star Topology. In *IEEE International Conference on Dependable Systems and Networks (DSN 2003)*, San Francisco, Cal, USA, Jun. 2003.

[2] ARINC. Software Considerations in Airbone Systems and Equipment Certification. ARINC, Annapolis, Maryland Documents RTCA/DO-178B, December, 1992.

[3] ARINC. 664 Standard. http://www.arinc.com, 2003.

[4] ARINC. Minimal Operational Performance Standards for Avionics Computer Resource. ARINC RTCSA-SC-182/Eurocae WG-48, Washington D.C., June 4, 2004.

[5] C. Davis. Data Processing Spheres of Control. *IEEE Systems Journal*, 17:179–198, 1978.

[6] C. Eder. Designing a high-performant real-time architecture based on cots-components. Master's thesis, Vienna University of Technology, Real-Time Systems Group, Vienna, Austria, 2002.

[7] Ethernet. IEEE Etherent Standard 802.3. Available at: http://standards.ieee.org, 2002.

[8] Ethernet. EtherType Field Public Assignments. http://standards.ieee.org/regauth/ethertype/eth.txt, 2005.

[9] R.-T. Ethernet. Information about Real-Time Ethernet in Industry Automation. http://www.Real-Time-Ethernet.com, 2004.

[10] F. Furrer. *Industrieautomation mit Ethernet TCP IP und Web Technologie*. Heidelberg, Huethig Verlag, 2003.

[11] H. Kopetz. Sparse Time versus Dense Time in Distributed Real-Time Systems. In *Proceedings of the 12th International Conference on Distributed Computing Systems*, pages 460–467, Yokahama, Japan, June 1992.

[12] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.

[13] H. Kopetz. Fault Containment and Error Detection in the Time-Triggered Architecture. In *In Proc. of the he Sixth International Symposium on Autonomous Decentralized Systems*, volume ISADS, pages 139– 146, Pisa, Italy, Apr. 2003.

[14] H. Kopetz. On the Determinism of Transmission Channel. Research Report 48/2003, Vienna University of Technology, Real-Time Systems Group, Vienna, Austria, 2003.

[15] H. Kopetz, A. Ademaj, and A. Hanzlik. Clock-State and Clock-Rate Correction in Fault-Tolerant Distributed Systems. In *The 25th IEEE International Real-Time Systems Symposium, Lisbon, Portugal, Dec. 2004*, pages 415–425, Dec. 2004.

[16] H. Kopetz, G. Bauer, and S. Poledna. Tolerating Arbitrary Node Failures in the Time-Triggered Architecture. *SAE 2001 World Congress, Detroit, MI, USA*, Mar. 2001.

[17] H. Kopetz and G. Grunsteidl. TTP - A Time-Triggered Protocol for Fault-Tolerant Real-Time Systems. In *In Proc. 23rd International Symposium on Fault-Tolerant Computing*, volume FTCS-23, pages 524–533, Tolouse, France, June 1993.

[18] C. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20:46–61, 1973.

[19] A. Mok. *Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment*. PhD thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, May 1983.

[20] OMG. Smart Transducer Specification TTP/A. ftp://ftp.omg.org/pub/docs/orbos/01-10-02.pdf Object Management group, 2002.

[21] K. Popper. Of Clouds and Clocks. *Objective Knowledge, Oxford*, pages 206–255, 1971.

[22] J. Rushby. Formal Verification of Group Membership for Time-Triggered Architecture. SRI International, Menlo Park, Cal, September 2000.

[23] M. Schwarz. Implementation of a TTP/C Cluster Based on Commercial Gigabit Ethernet Components. Master's thesis, Vienna University of Technology, Real-Time Systems Group, Vienna, Austria, 2002.

[24] W. Steiner. *Startup and Recovery of Fault-Tolerant Time-Triggered Communication*. PhD thesis, Vienna University of Technology, Real-Time Systems Group, Vienna, Austria, 2004.

[25] N. Suri, C. J. Walter, and M. M. Hugue. *Advances in Ultra-Dependable Distributed Systems*. IEEE Computer Society Press, 1995.

[26] TTTech. Time-Triggered Protocol, High Level Specification Document. Vienna, Austia, D-032-S-10-28 Available at http://www.tttech.com, 2002.