Time-Triggered Systems for Safety Critical Applications

Kartikay Srivastava

Abstract

A Time-Triggered Architecture (Time-Triggered System) is used to create a computing mechanism for the design and development of a dependable distributed embedded system. The basic principle of a Time-Triggered Architecture is to decompose a large-real-time application into self-controlling clusters and nodes.

A fault-tolerant global time base of known precision is generated at each node. This global time is used to specify the interfaces, communication and agreement protocols among nodes to perform error detection and guarantee the timely working of a real life application.

This report presents the model of a Time-Triggered System (Time-Triggered Architecture), explains the design of a TTA, key factors which make a TTA suitable for a real-time application development and finally presents a real-time application development using Time-Triggered Architecture.

Contents

List of Figures

1 Introduction

A Time Time-Triggered System (also know as Time-Triggered Architecture) is a type of computer system which executes one or more set of tasks in accordance to a pre-determined and fixed schedule.

The implementation of such a system generally involves the use of a single interrupt that is associated to a periodic flow of a timer. This interrupt thus drives a task-scheduler. This scheduler then starts to release these tasks at pre-determined points in time.

Since these systems posses a highly-determinitic behaviour, Time-Triggered systems have been used for many years to develop safety-critical and related systems (like aerospace, automotive, nuclear engineering and many more). [2]

A Safety-critical system(or life-critical system) can be defined as a system whose failure or malfunction is highly likely to result into a lethal outcome like:

Death or serious injury to people.
Loss or damage to equipment.
Environmental harm.

Since safety critical systems require the development of a highy dependable computer systems, a Time-Triggered System is considered appropriate to develop such systems.

The major characteristic of a Time-Triggered System is handling of real-time as a primary quantity. A Time-Triggered System breaks down a very large embedded system into small groups and nodes. This in turn provides a fault-tolerant global time base of a known precision on each node.

A Time-Triggered System hence picks up on the advantage of availability of a globally synchronized time to determine interface among nodes, communication, error detection and therefore guarantees the timeliness of real-time applications.

This report on Time-Triggered Systems is organized as follows:

Section 2 deals with the architecture on which a Time-Triggered System is developed(this section covers some important concepts such as temporal predictability).

Section 3 covers the properties of a Time-Triggered System which make it suitable for developing safety-critical system.

Section 4 finally ends this report with a conclusion.

2 Architecture Model

The computational model that lies under a Time-Triggered System is a Time-Triggered(TT) model.

2.1 Model of Time

The model of time on which the TTA is developed is based on Newtonian physics. Real time progresses along a time-line, which consists of an infinite set of instants(from the past to future). An interval is a section in this time-line which is delimited by two instants. An event is a happening that occurs at an instant of the timeline. Therefore an event can also be known as an observation of the state of the world.

The time-stamp (time of occurrence of an event) is established by assigning the state of a node's local time to the event immediately after the occurrence of the event.

A fault-tolerant internal clock establishes global-time in a Time-Triggered System. In a Time-Triggered System it is highy probable that clocks are not synchronized perfectly due to denseness of real time, consider the following example: a clock in node j ticks, event e has occured, clock in node k ticks. Therefore in this situation the event e has been marked by two clocks j and k with a difference of one tick.

Therefore in a distributed system, due to the precision of global timebase it is highly impossible to order these events consistently on the basis of their global time-stamp. A Time-Triggered System tackles this scenario by introducing a sparse-time base (as can be visualized by the figure below). [2]

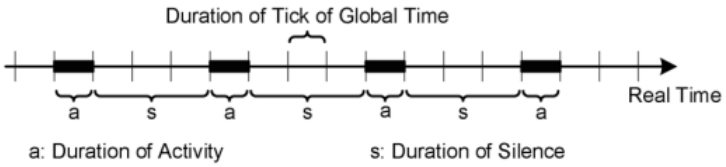

Figure 1: Sparse Time Base

Sparse-time model is a continuum of time partitioned into an infinite set of alernating durations of acitivity and non-activity durations as shown in the figure above. The length of an activity (section of global time) must be larger than the precision of the clock synchronization.

In order to maintain temporal ordering, all events which have occured within an interval are considered to take place at the same time. Therefore events that have taken place in a distributed system at different nodes at the same global clock time are considered simultaneous.

Events that took place during different durations of activity and can be separated with the required interval of silence can be consistently ordered based on their global time-stamps. A Time-Triggered System must make sure that important events such as sending a message must take place only during an interval of the activity.

Therefore the timestamp of an event which lies outside the control of a distributed system must be assigned to an agreed interval of activity by an agreement protocol.

2.2 Time and State

The notion of state is introduced to distinguish the past from the future. Keeping this in mind it can be assumed that state and time in a system are inseparable entities.

For example if an event updates the state but it does not coincide with a well-defined tick of global clock on sparse time-base, the notion of system-wide state becomes irrelevant. It is not known whether the state of the system at a given clock tick includes this event or not.

Such a consistent view of time and state is very important if fault-tolerance is to be implemented, in such cases faults are masked by voting on replicated copies of the state residing in separate fault-containment regions. [2]

2.3 State Information vs Event Information

Any information that is being exchanged across an interface is either state or event information. A property of a real-time (RT) system (eg a relevant state variable) that can be observed by a node of a distributed real-time system at a particular instant is called state attribute and the accompanying information is called state information. Therefore the state observation holds the information about a state variable at a particular instant.

An event is a sudden change of state of a Real-Time entity that had occured at an instant. The information that describes such an event is called event information. Therefore Event Information can be defined as the difference between the state before the event and the state after the event.

Event information must be transmitted in event messages, this information an be used to rebuild the states and events of the system in a changing environment. Therefore starting from an initial stage, a complete sequence of (non-continuous) events can be used to reconstruct the complete sequence of states of a Real-Time Entity. [4]

2.4 Structure of Time Triggered Systems

A node is the building block of a Time-Triggered System.

A node can be defined as a self-contained unit which consists of a processor with memory, an input-output subsystem, a time-triggered communication controller, an operating system and the required application software (as can be referred from the figure below):

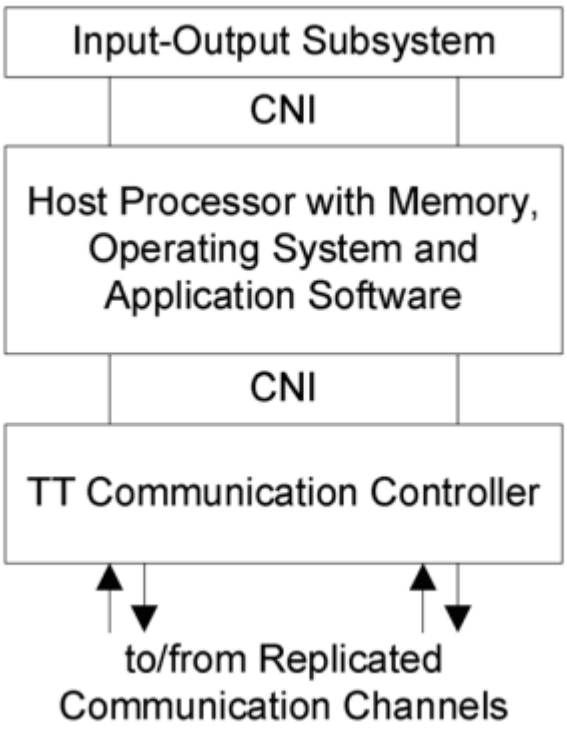

Figure 2: Node of TTA

Two replicated communication channels connect the nodes and thus form a cluster (grey shaded area in the figure below). The cluster communication system comprises the physical interconnection network and the communication controllers of all nodes of the cluster.

In a Time-Triggered System, the communication system works autonomously and executes periodically a pre-defined TDMA (Time-division multiple access) schedule.

Clusters can be connected via gateway nodes. A gateway node is a member of two clusters and hence contains two connection interfaces. A gateway node restricts the view of one cluster to the other in order to reduce complexity. [2]

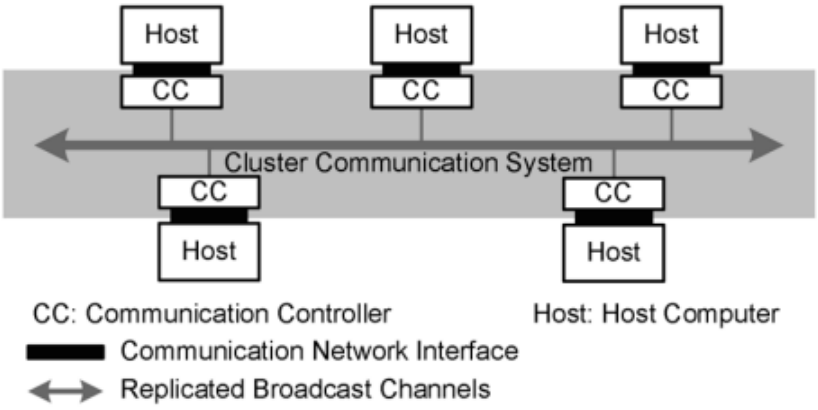

Figure 3: Structure of a TTA cluster

2.5 Interconnection Topology

A Time-Triggered System supports two different types of physical interconnection:

Bus Topology: In this kind of system, the physical interconnection consists of multiple passive busses (as can be depicted from the figure below). At every physical node site there are three subsystems (a node and two guardians). The guardians are independent units which observe a known temporal behaviour of an associated node.

If a node sends a message outside its determined time-slot, the guardian will cut off the physical transmission path and thus eliminates it. [2]

Figure 4: Bus Topology

Star Topology: In this kind of system, the guardians are integrated into two replicated central star couplers instead of sequence of linearly interconnected guardians (can be depicted from the below figure). [2]

The Star topology architecture boasts the following **advantages**:

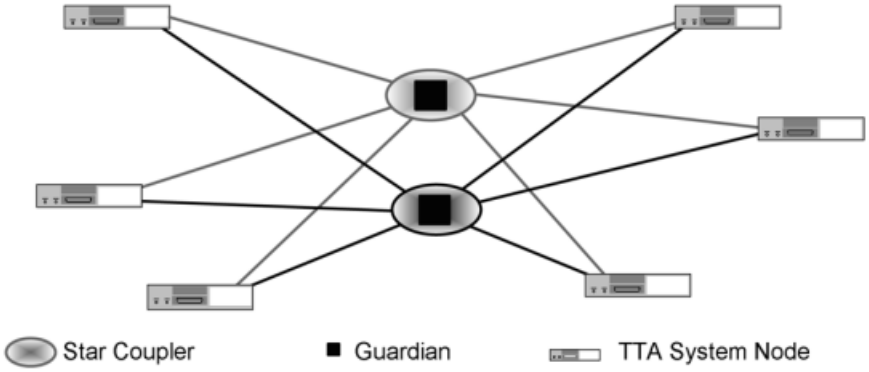

Figure 5: Star Topology

The guardians are fully independent.

The algorithms in the architecture can be extended to provide additional safety services such as: condition-based maintenance.

A bus system possesses better one on one communication properties than a linearly connected bus system.

3 Properties of Time Triggered Systems

The previous section of this paper introduced us to architectural properties of a Time-Triggered system.

The next sections will discuss the features of the Time-Triggered systems which make it feasible to develop Safety-Critical Systems.

3.1 Fault Tolerance

A Time-Triggered System can be used to develop safety-critical real-time applications like: control of an aircraft or cruise control of an automobile (due to their highly deterministic fault tolerant behaviour), one way to achieve this property is to use active redundancy by replication and voting.

Active replication requires a number of mechanisms such as replica coordination, voting, membership fusion, internal state alignment as well as reintegration of nodes after a transient failure. When such generic fault tolerance mechanisms are combined with application software of a node, there is an increase in the complexity of the underlying software and could prove to be the cause of additional design faults.

In a Time-Triggered System such fault tolerance mechanisms are implemented in a dedicated fault-tolerance layer (as can be depicted from the below

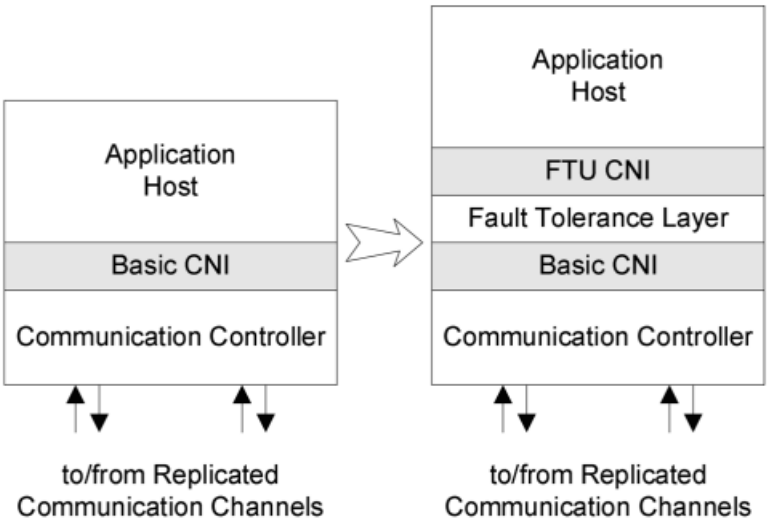

Figure 6: Expansion of a non-fault tolerant node into fault-tolerant node

figure) with its own processor (the structure of such a fault tolerant unit is similar to a non-fault tolerant unit). A properly structured application software can be used as fault-tolerant system or a non-fault-tolerant system without any special modifications. The fault-tolerance mechanism remains an integral part of a Time-Triggered System. [3]

3.1.1 Fault Hypothesis

A fault-tolerant design starts by the specification of its fault hypothesis. A Fault Hypothesis states the types as well as the amount of faults that a system must tolerate. In a Time-Triggered System a chip is a single fault-containment region, as all the functions of a chip share the same power supply, ground, oscillator, mask, manufacturing process and all work in the same proximity.

In a Time-Triggered System star configuration a cluster will tolerate an arbitrary failure of a Time-Triggered System node (chip). A faulty unit can thus be consistently detected by membership protocol (agreement) and can be separated. A Time-Triggered System therefore can mask an asymmetric fault of any node by architectural properties and guarantees an efficient temporal and spatial partitioning of the nodes at cluster level.

3.1.2 Fault-Tolerant Units

To handle an internal fault the preferred fault-tolerance strategy is active replication of the independent nodes. The partitioning of the nodes as well as masking of faults by a Time-Triggered System creates the prerequisites to form a

logical group of a set of nodes or software systems which compute the same function into a fault-tolerant unit (FTU). Thus FTU can tolerate the failure of any its independent parts without compromising any of the services provided by a Time-Triggered System. The nodes comprising a FTU must be positioned physically far apart from each other in order to tolerate physical proximity faults.

FTU comprises of three nodes (Triple Modular Redundancy, TMR) that operate in a replica deterministic manner, the FTU then represents these results to a voter (who is located at every consumer of the result) that makes a majority decision. TTA provides replica determinism at the CNI of a node. The host software runs in synchronization on three different host computers and produce the results simultaneously at the FTU node (can be referred from fig-6)

The FTU layer then starts distributing messages to other nodes in the cluster. Before receiving any messages, the FTU layer at the respective node vote on the incoming messages and present the results to the hosts at delivery instant. This process is repeated periodically and every host must share its internal state for the voting procedure.

As can be inferred from the above procedure an integration node must wait until it has received a voted internal state before it can start participating in an application. A Time-Triggered System supports self-checking pairs of nodes. A self-checking node will produce results only if it is correct in the temporal and value domain. A self-checking FTU will operate with two nodes to tolerate a single node failure. Therefore a host application must be designed in such a manner that the duration between the fetch and delivery instants at the communication units must be long enough to perform fault-tolerance features. If the system meets this requirement, the FTU layer will be able to perform appropriately.

3.1.3 Never Give Strategy (NGU)

The fault tolerance services that have been mentioned previously can only be implemented if the environment co-operates with fault hypothesis. If the environment is non-compliant to fault hypothesis a Time-Triggered System activates Never-Give-Up (NGU) strategy. The NGU strategy is initiated by the communication protocol of a system in co-operation with the application as soon as it is evident that the available resources are not enough to provide minimum required services.

The NGU strategy is application specific eg. if the reason for an outage is a massive transient fault, then the NGU strategy will most likely consist of freezing the actuators in their current state until the cluster can be successfully restarted.

3.2 Composability

A Time-Triggered System has a predictable response time and an unrestricted length of the data field. When label information needed by a given protocol

is not available in the TTP-control field, it is possible to use some bytes of the TTP data field to adjust additional protocol information. This required protocol conversion can be calculated locally in the communication controller in such a manner that the system does not have to bother for use of additional resources.

Therefore it is highly compatible with all protocols that have an unpredictable response time and have a restricted data field, for example IEC 61508 (industrial systems), ISO 26262 (automotive systems), IEC 62304 (medical systems). [1]

3.3 Temporal Predictability

A Time-Triggered System is an integrated communication protocol. It therefore provides services required for implementing a fault-tolerant real-time system some of which are: predictable message transmission, message acknoledgement, clock synchronization and redundancy avoidance.

A Time-Triggered System must support temporal predictability: which means that the occurance of an event can be predicted in advance (without being concerned about the order in which event was developed) and hence the Time-Triggered System can check if there is a fault in the system.

The developers of a Time-Triggered System are able to fulfill temporal predictiability requirement by using appropriate system architectures, scheduling and implementation models and by also verifying the timeliness of the resulting system against the desired specification. The quality of temporal predictiability strongly depends upon the accuracy with which the time taken to complete an operation on the target system can be determined.

If the duration to complete an operation is variable then the applications running in such a system will have considerable jitter and the exact execution time will be difficult to calculate. Whereas if the time taken to complete the operations is invariable, such system will be able to meet the even the tightest of constraints and it will be easier to predict the time of completion of an activity.

3.4 Implicit Synchronization

A global time-base with a pre-defined precision is one of the most basic services that must be provided to a decentralized real-time application. A Time-Triggered System provides a fault-tolerant Internal Synchronization of the local clocks to create a global time-base of known precision.

Due to the availability of such a time-base the receiver knows in advance about the time at which each frame is being sent, the deviation between a predetermined sending time and the observed receiving time is an indication of the difference in clock between the sender's and receiver's clock.

Therefore it's not necessary to calculate the value of send time in the frame because continuous clock synchronization is being carried out without any overhead by applying the fault-tolerant algorithm periodically (mostly with hardware support)

This pre-defined schedule satisfies synchronization requirements based on a global time-base which includes precedence constraints and avoids race conditions. [1]

4 Conclusion

A real time safety critical system must respond to state changes in a very small interval of time. An event-driven system can be used to develop such systems, however these systems come up with the shortcomings of unreliability and determinism point of view, which prove to be critical aspects in development of such systems.

A time-driven architecture (Time-Triggered System) which boasts the advantages of being timely and fault tolerant with a lower rate of resource utilization prove to be quite beneficial to develop such systems.

References

[1] E Anbarasi, N Karthik, and R Prabakaran. Analysis of time triggered schedulers in embedded system. In *2011 3rd International Conference on Electronics Computer Technology*, volume 1, pages 134–137. IEEE, 2011.

[2] Hermann Kopetz. Event-triggered versus time-triggered real-time systems. In *Operating Systems of the 90s and Beyond*, pages 86–101. Springer, 1991.

[3] Roman Obermaisser, Hamidreza Ahmadian, Adele Maleki, Yosab Bebawy, Alina Lenz, and Babak Sorkhpour. Adaptive time-triggered multi-core architecture. *Designs*, 3(1):7, 2019.

[4] Naimish Thaker and SS Krishnamurthy Babu. Analysis of event-triggered and time-triggered architecture for a reliable embedded system. In *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–5. IEEE, 2017.