TTA-Group

TTP – Frequently Asked Questions

This document answers the most commonly asked questions about TTP. Some of these questions have arisen from the inherently complex nature of time-triggered real-time systems that must meet high safety and fault tolerance requirements. Other questions concern basic assumptions about, and design rules for, dependable computer systems.

Table of Contents

1. Application Domain, Migration Strategies 2
2. Safety, Availability, and Fault Tolerance 5
3. Membership and Consistent Agreement 9
4. Clock Synchronization 11
5. Flexibility 13
6. Configuration Data (MEDL) and CPU Interface (CNI) 14
7. Operation of Partial Networks and Start-Up 16
8. Physical Layer 16

Contact 18

1. Application Domain, Migration Strategies

Q: ***Is TTP suitable for a wide range of safety-critical applications?***

A: Yes. TTP has been designed for a broad range of applications. The focus, originally, was on backbone communication buses for automotive systems. For this kind of system, an important consideration is the ease with which existing as well as new, safety-relevant applications can be integrated into the system. This, therefore, was one of the main design considerations of TTP at the time. When the aerospace industry showed an interest in developing safety-critical applications using TTP, the high safety standards required for their applications were integrated into the design of TTP. Today TTP is used cross-industry-wise because it meets the highest safety requirements.

TTP can be used in different applications and industries. It has three important aspects.

- ***Safety:*** A high safety standard has to be an integral part of system design from the very beginning. Improved safety cannot be "tested into" the system or be retro-fitted at a later time. TTP, therefore, has the highest safety requirements built into it as an integral part of its design. This is a very important aspect of making an architecture future-proof. Migration can be started with a non-critical application and continued with safety-critical applications. This can be done without changing the architecture since both application types are supported by TTP.

- ***Composability:*** An increase in system complexity requires that the architecture support testing and integration. In a system having the property of composability, it is guaranteed that extensions which modify existing functions or have new functions will only affect specified subsystems. There can be no side-effects that call for a system-wide test. TTP provides composability for real-time communication.

- ***Data rate:*** Future applications will require considerably higher data rates. TTP has been designed to provide no limit to data rates and throughput. With TTP, therefore, new functions can be added even while reducing the number of gateways and the cost of the overall architecture.

Q: ***Does TTP meet the requirements of the industry?***

A: TTP was evaluated with respect to several requirements for next-generation automotive applications. Its suitability for these applications was established with regard to:

- Safety
- Cost-effectiveness
- High data rates
- Composability and ease of system integration
- Flexibility in terms of vehicle platforms and model variations
- Suitability for end-of-line programming (flash programming) and diagnosis
- Extensibility in the field

Furthermore, TTP also meets aerospace requirements for Level A safety-critical applications.

Q: ***What data rates does TTP support?***

A: The TDMA bus access scheme is collision-free and puts no limit on the data rate. The communication controllers available today support 25 Mbit/s synchronous and 5 Mbit/s asynchronous transmission (Asynchronous transmission is the method used over twisted pair wiring, synchronous transmission uses Ethernet-like wiring). Data frames can carry a payload of up to 240 bytes each. TTP has demonstrated a net data rate to gross data rate ratio of up to 85%. Prototype implementations have used 1 Gbit/s technology (lab experiment in 2002).

Q: ***Does TTP limit the frame size (size of data bytes per transmission)? Do all TDMA transmissions need to have the same frame size?***

A: TTP allows a free choice of the number of data bytes per transmission, which allows node configurations to be adapted to application requirements. As a result, configuration is highly flexible, and the bandwidth provided by the physical layer can be efficiently utilized.

Q: ***What network topologies are supported by TTP?***

A: TTP networks can contain up to 64 nodes. The cabling topology can be bus, star, or any combination of the two. Multiple stars or sub-buses on stars are also supported.

A redundant star topology with a bus guardian integrated into the star coupler has the advantage of combining the highest safety level with minimal cost. This approach offers a significant cost advantage compared to a node-local bus guardian architecture (about € 47 in a 10-node system).

For detailed information about TTP cost advantages please contact info@ttagroup.org.

ECU: Electronic Control Unit

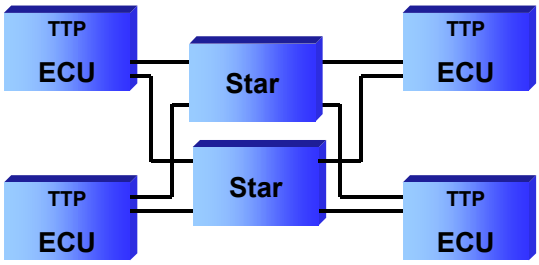

Figure 1: TTP in star topology

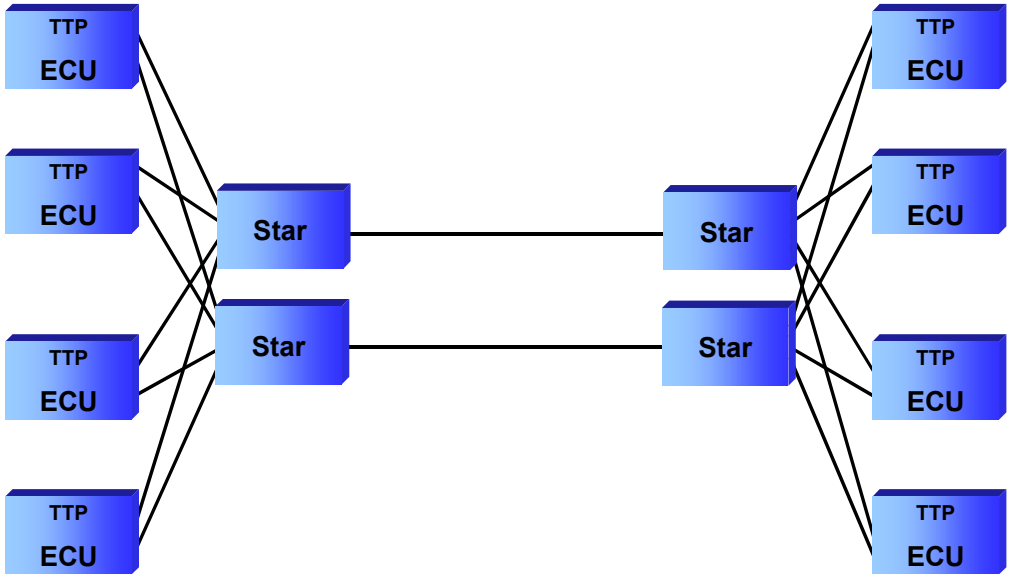

Figure 2: TTP in cascaded star topology

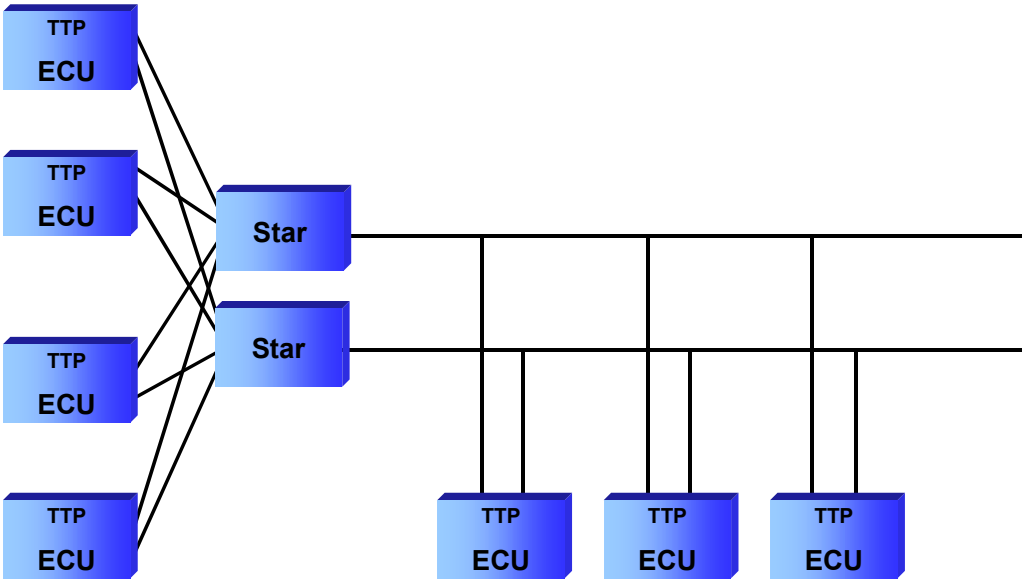

Figure 3: TTP in star topology with bus branch

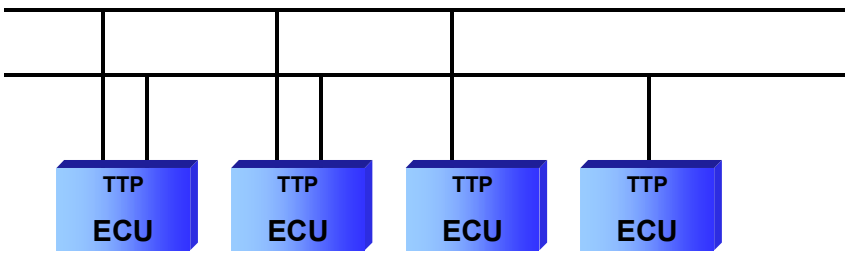

Figure 4: TTP in single channel topology

Q: ***How can existing CAN applications be migrated to TTP?***

A: An efficient migration strategy is needed to optimally bring existing CAN-based software into a new generation of products. Apart from CAN-TTP-Gateway solutions, a CAN emulation layer can be used on top of TTP. In the emulation layer, the registers of a CAN controller module are emulated, and this allows for the re-use of existing CAN software. A small portion of the TTP bandwidth is reserved for CAN, and CAN messages themselves are transmitted within regular TTP frames (on this reserved portion of the bandwidth). On a TTP bus at 10 Mbit/s, approximately 5% of the net bandwidth needs to be reserved to emulate a high-speed CAN network at 500 kbit/s. An experimental hardware-based CAN emulation layer was implemented at the Vienna University of Technology in 2002.

Instead of a full CAN emulation hardware solution, a software solution is also possible. In this scenario, CAN signals and identifiers are transported over a middleware layer, which supports event channels on TTP.

Both options allow existing CAN-based software to be migrated onto a TTP-based system with minimal effort.

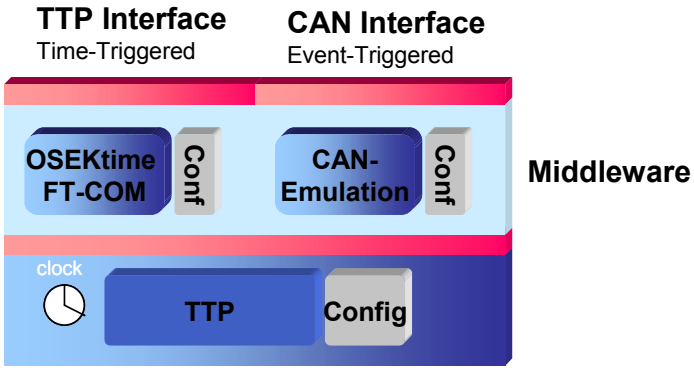

Figure 5: Event layer / CAN emulation with TTP

For detailed information about event data and migration strategy please contact info@ttagroup.org.

2. Safety, Availability, and Fault Tolerance

Q: ***What methods were used to verify correct functionality and safety of TTP?***

A: A large part of TTP has been formally verified. As a result, TTP is considered the most comprehensively verified communication protocol today—at least in the automotive industry. The following procedures were used:

- ***Formal verification:*** The protocol's core algorithms for consistency, stability, and safety were checked for correctness with formal mathematical proofs. This kind of verification guarantees that all system states are checked, and it is in contrast to testing, which checks only a subset of all possible system states. Formal verification was, and is being, carried out in cooperation with SRI International and other leading universities such as University of Ulm, University of York, University of Paris, and Vienna University of Technology. NASA has also supported this effort.

- ***Millions of fault injection experiments:*** In two large projects funded by the European Commission (PDCS and FIT), different methods of physical and software-based fault injection were used on TTP systems to evaluate TTP's fault tolerance behavior and error detection properties. The experience gained during these projects was extremely helpful in improving the existing technology; it has also demonstrated how cost-optimal safe solutions can be realized. The FIT project showed that a new concept of a bus guardian must meet highest safety requirements. Based on these experiences, the intelligent star coupler bus guardian was developed, which guarantees optimal safety at minimal cost. In 2003 the SP Swedish National Testing and Research Institute conducted heavy-ion fault injection experiments to validate the Time-Triggered Architecture (TTA) with TTP-C2 communication controllers. According to their report, the fact that no fail silence violations were observed with TTP-C2 communication controllers in the star and the bus topologies makes TTA one of the most reliable distributed computer architectures for highly dependable real-time systems.

  If you would like to receive a copy of the SP report, please contact info@ttagroup.org.

- ***Certification:*** The use of TTP in applications with the highest safety level requires that the protocol be certified. The development process and the design of the TTP-C2 chip model are both fully documented as required by aerospace standards. The documentation and processes of TTTech Computertechnik AG, combined with the relevant documentation and processes of the respective semiconductor manufacturer are the basis to certify products according to the RTCA software standards used by FAA (Federal Aviation Agency) or JAA (European Joint Aviation Authorities). The OSEKtime-based operating system ^{TTP}OS is developed according to the RTCA software standard DO-178B Level A. The firmware providing the protocol functionality of the AS8202NF communication controller (based on TTP-C2NF) is certifiable in compliance with the DO-178B standard for Level A applications. ^{TTP}Verify has been designed as a software verification tool in compliance with the software development standard RTCA DO-178B and supports the verification of safety-critical distributed control systems developed under RTCA DO-178B Level A. This design directive is one of the most stringent certification standards for developing safety-critical software.

- ***Project experience and use in commercial applications:*** TTP has been used in projects and prototypes for more than 10 years. FPGA-based systems were first used in 1995, and TTP ASICs have been available since 1998. Also, several development projects have used TTP and gained experience in using the system. The first TTP-based systems were deployed in the field in 2002. The first TTP-based fly-by-wire systems will be in commercial production in 2004. Honeywell deploys TTP in full authority engine control systems on the Lockheed Martin F-16 and the Aermacchi M-346 trainer fighter. Honeywell's APEX integrated cockpit using TTP is applied in single-engine turboprop aircraft such as GROB Ranger G 160, EXTRA EA-500, and IBIS Ae270 and several other airplanes. TTP is used in the Airbus A380 Mega-Airliner. Nord-Micro has selected TTP as communication protocol for the Airbus A380 cabin pressure control system. Since 2002, Alcatel has been using TTP as field bus protocol in its commercially produced railway signaling system ELEKTRA 2. The system is used in Switzerland, Austria and Hungary.

A total of more than 170 man years of development have been invested in the safety of TTP.

Q: ***Do the formal verification methods verify the actual algorithms in the chip?***

A: No, they verify the algorithms as they are specified in the current protocol specification, which is also the basis for chip implementations. Additional verification activities in the form of peer-reviews and conformance testing verify that the algorithms in the protocol specification are implemented correctly in the chip design.

Q: ***Is the proof of correctness at the protocol level of any use if application-level checks and end-to-end checks have to be performed in addition anyway?***

A: Consistency checks at the protocol level guarantee that all communicating nodes have identical information, or that an error signal is raised by the communication layer. The consistency checks at the protocol level do not make the application-level checks obsolete, they support them. This assurance can be used as a proven argument in a system safety case—which makes the proof of safety at the system level much simpler. If the protocol does not provide such mechanisms as an acknowledgment and consistency check, each application must provide these mechanisms separately if they are required for safety or integrity reasons. Doing this can mean more effort and cost for each application. Furthermore, if the proof of correctness is carried out, for the most part, at the software level, then every change in the software may require an additional round of verification. For safety-critical functions, the costs could become very high. Proof of consistency at the protocol level, therefore, can simplify the safety case at the system level.

Q: ***If a communication controller and a bus guardian are integrated on a single chip, would the requirements for safety-critical applications be met?***

A: No, fault injection experiments have shown that common mode faults with uncontained effects are all too likely in a single-chip solution, at least for highly safety-critical applications. What is needed is an external bus guardian that has higher levels of integrity than a single-chip solution. External, node-local bus guardians are currently not available for TTP, but a central guardian (i.e. star coupler) is. However, for non-safety-critical applications, a single-chip solution of communication controller with integrated bus guardian offers improved availability at minimal cost.

Q: ***What is the fault tolerance strategy of TTP?***

A: The so-called “single fault” hypothesis. A fault hypothesis is required to define and validate the fault tolerance properties of a fault-tolerant system. The fault hypothesis explains the fault model that is the basis for error detection and fault tolerance. The “single fault” hypothesis requires that any single fault—up to arbitrary failure modes resulting from an error of a complete node—is reliably detected and tolerated. Besides that, many multiple faults can be detected and tolerated. In addition, the protocol specifies higher-level error detection for faults that are outside of the “single fault” hypothesis and therefore cannot be tolerated by TTP itself.

This leads to a three-level fault strategy, which can be adapted for each implementation and application level:

- Operation in a fault-free scenario;
- Error detection and integrated fault tolerance, for all single and some multiple faults;
- Error detection (without appropriate fault tolerance strategy in the protocol layer) for all multiple communication faults; such faults, e.g. a complete temporary loss of communication, are not handled by the integrated fault tolerance of TTP but need the attention of the application (never-give-up strategy).

Q: ***What are “SOS faults” and how are they dealt with in TTP?***

A: This class of rare faults is especially tricky to address since it brings the communication subsystem into a state of permanent disagreement about the actual fault. “SOS” stands for “slightly-off-specification” and indicates that the effect is marginal, i.e., the SOS fault is detected as a fault by some components (such as line drivers and communication controllers) but accepted as a non-fault by others.

Such faults, once they occur, can create errors so frequently that a distributed error detection mechanism cannot be relied upon to pick them all up and handle them appropriately. The only

way to address this kind of fault is to prevent SOS signals and remove them completely from the system. Such prevention must be performed by a dedicated central unit, such as a star coupler.

Q: What is a "babbling idiot" fault?

A: If a communication participant (node) in a distributed system does not communicate in the intended way but tries to monopolize the shared communication link by sending "more than allowed," this node is called a "babbling idiot." There are many different faults that can lead to a babbling idiot, and include faults in the application software of event-driven communication systems, but can occur in any kind of communication system.

In fail-safe systems, which can shut down if a babbling idiot fault were to occur, only error detection of the babbling idiot is required. In fail-operational systems, babbling idiot faults must, additionally, be tolerated. The only reliable way of detecting and tolerating this fault is to have a separate guarding unit, often called a bus guardian, which protects the bus from any illegal access.

Q: What happens in a TTP system when a communication error, or any other type of error, occurs?

A: In case of single faults—which are always tolerated—there is usually no reason for any kind of recovery. TTP detects and reports any single fault so that a specific system response can be initiated if necessary.

In case of multiple communication faults—especially so-called clique faults—a system restart may be the only option available to restore consistency in the system. This is because in such cases the communication system is unable to provide all nodes with correct and consistent data. When this happens, the application is notified of the fault and has to decide upon a reasonable course of action. For some applications, it may be useful to configure the system so as to continue operation with inconsistent data; in other cases, stand-alone operation in the safe state is more advisable. In whatever way the system chooses to proceed, it is the decision of the application—and not of TTP—to perform or not to perform a restart.

It should be noted, however, that multiple faults occur rarely if at all in a well-designed communication system. However, the possibility must be provided for, and a detection mechanism set up.

Q: What is the "never-give-up" strategy of TTP?

A: TTP provides mechanisms to detect communication faults that cannot be tolerated by the communication layer (multiple communication faults, SOS faults). The never-give-up strategy is based on this error-detection capability and mandates a clean restart and recovery in all cases that cannot be handled by more efficient or faster recovery mechanisms.

Q: Does TTP fault tolerance require whole nodes to be redundant, or can fault tolerance be made scalable?

A: This question refers rather to the system architecture than to TTP. TTP facilitates strictly synchronous redundant functions and therefore enables "replica-deterministic" redundant components. Components can be complete nodes or individual software modules. Such redundant functions (subsystems) can be dual-, triple-, or even higher-level-redundant. It is also possible to design different modes of operation on a node where some important redundant functions are executed (and some non-important ones are left out) only if another node executing the important function fails. In this way, a very fine scale of fault tolerance can be designed (see Figure 6).

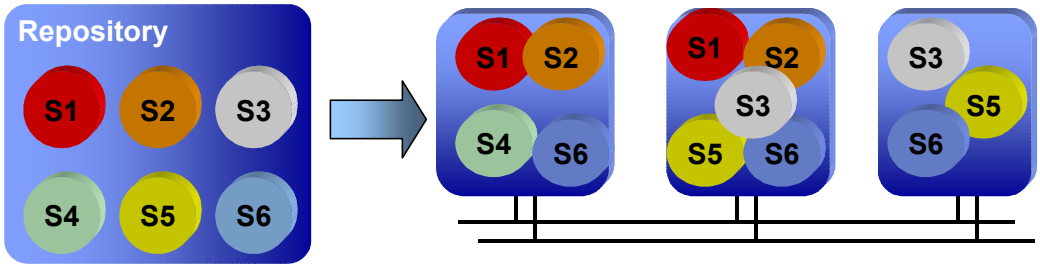

Figure 6: Replication of subsystems with TTP

3. Membership and Consistent Agreement

Q: ***What are the benefits provided by the TTP membership service protocol?***

A: The purpose of the TTP membership protocol is to provide reliable error detection of all communication faults. Such faults include transmission faults, reception faults, and even some that occur outside the scope of the communication layer fault hypothesis (such as clique faults during transmission). Faults in the application software are not covered by the membership service, even though some application faults may lead to a membership loss (for example, the setting of a faulty life sign, and the requesting of an invalid mode change). The purpose of communication-level membership is to provide fast and reliable error detection. Using communication-level membership, an application can detect the source of a problem more efficiently and reliably, and can perform fast and correct error handling and recovery.

A correct membership flag of a node M in the local communication controller thus indicates “no errors in the communication between node M and the other node.” In combination with the consistency checking, it also indicates “every node in the membership has detected correct communication with node M.” A node, which is not in the membership, has been diagnosed as “not communicating correctly,” which is quite different from “the CPU on that node has not set its life sign flag.” A communication error is diagnosed within a short time interval, short being defined as two TDMA slots for consistent faults, less than two TDMA rounds for arbitrary communication faults.

The decision to introduce a life sign in TTP and not let a communication controller transmit if this life sign is not serviced correctly is not a primary design element for the membership, but allows intentional phases of receive-without-transmission by the application, e.g. for reconfiguration or reintegration.

Q: ***Membership information can be read incorrectly at a node, and this would lead to inconsistent activity on that node. How can you protect against this?***

A: There is no guarantee that information from a communication controller will be read correctly by the CPU. This applies not only to membership information, but also to all data. This issue is therefore neither a membership issue nor a TTP issue. It can be addressed by specific hardware mechanisms that can reliably detect or prevent such a fault, or by not making the fail silence assumption for a node. Both approaches can be used with TTP.

Q: ***How much of the membership mechanism is used in the formal verifications of TTP? What is the semantics of the membership in these verifications?***

A: All formal verifications of TTP take account of the membership mechanism with regard to how it provides consistency of communication. (For a definition of consistency, see the discussion below.)

This means that a case for the safeness of a distributed system using the membership mechanism can assume the system’s consistency properties to be verified, provided that the

mechanism has been correctly implemented (in hardware) and the implementation has been independently verified. Otherwise, each software implementation has to be verified individually, and for each change made to the software.

Q: ***How long does the agreement of the TTP membership service take in the worst case? Is this fast enough for any known application?***

A: For consistent faults (i.e., faults which are seen in the same way by all nodes) two successive nodes (i.e., two sending slots with a typical combined length of 200 microseconds) are needed in the worst case. In the best case, which is the fault-free case, only one node is needed.

For inconsistent faults, the worst case is 1.5 rounds, with round being defined as the shortest time interval between any two transmissions of a node. Nodes may have a longer interval between two successive transmissions if multiplexing is used.

Both implementations are optimal with respect to timing. There is no faster way to provide single fault-tolerant acknowledgment and agreement over the bus, regardless of the application or communication protocol.

It is of special importance that the acknowledgment logic is implemented in dedicated hardware. In a high-speed system with 16 nodes and a round duration of 1 millisecond, error detection and acknowledgment must be calculated every 66 microseconds. When performing acknowledgment and consistency services in the application, different methods and slower acknowledgment cycles are used.

For detailed information about membership and consistent agreement please contact info@ttagroup.org.

Q: ***What does "consistent transmission" mean? Does it mean that all nodes connected to the same medium receive data practically simultaneously?***

A: The common definition of "consistency" includes much more than just the concept of "simultaneous reception." Most prominently, it includes the notion of "common knowledge," which is taken to mean that all information from a node is present on all other nodes that consider this information relevant. Also, that the information is present on "all other nodes" is taken to be guaranteed—not just assumed. The existence of such a guarantee would require two conditions to be satisfied: first, a mutual confirmation of reception (acknowledgment); and second, global acceptance of this confirmation (agreement).

Assumed or guaranteed consistency is a core element in the design of a distributed system. Only if there is consistency in lower layers (especially communication layers) can a higher layer (such as the application layer) assume that the information the system processes and produces is commonly available. Often, this requirement is assumed and not validated—an improper design choice for safety relevant systems.

Q: ***Why does TTP perform communication-level acknowledgment and consistency checks when it is necessary to perform application-level acknowledgment anyway?***

A: The question can also be put like this: if a distributed application has to ensure that it is in a consistent state, why must the communication system also check that it is in a consistent state? Has the application-level checking not ensured that already?

The error detection mechanisms used by TTP to ensure consistency at the communication layer (membership, acknowledgment, clique detection) complement application-level agreement/acknowledgment. Checking if a specific sensor value was correctly sent and received is a different issue than checking if the control function using that sensor value has computed correct outputs, and if these outputs were correctly received by the actuation unit.

It is arguably correct that a higher-level error detection like “is the output of the control function correct?” will also detect a missing transmission of the sensor value, which was an input to that control function. But it will take longer to detect this error, and it may be more difficult to find out what went wrong in the first place (it was not the control function!). So for some functions the application-level agreement may be quite adequate, while for others the faster and more specific communication-level mechanisms may be required. This is especially true for fast control functions, where responses to input data have to be given with the same rate as the communication cycle.

Q: ***Does membership reduce system availability? Does it force a node to restart after a single communication fault?***

A: No, the membership service does not reduce system availability. Membership serves to detect all communication faults, including transmission and reception faults. If a node fails to transmit (typically due to noise during the transmission) and is therefore removed from the membership, every node, including this node, detects it. The node can retry transmission in the next round (not immediate retransmission), and, if it succeeds, is included in the membership again. Faults which cannot be tolerated at the communication level (some multiple faults, clique faults), are reported and have to be handled at a higher layer. If the system decides that a restart is the best action to take in the event of a non-tolerable fault being reported, it must be understood that the restart is a system decision; the restart cannot be attributed to the membership service. The membership service simply detects the fault.

4. Clock Synchronization

Q: ***TTP is designed for systems with four or more nodes. How robust is the clock synchronization with only two or three nodes? Does it meet the requirements for safety-critical applications?***

A: TTP systems with two or three nodes (as active members in the clock synchronization) work reasonably well; the precision is less accurate than for larger systems. However, one of the core characteristics of a TTP system, single fault tolerance, is not met if there are less than four nodes. Single fault tolerance for asymmetric faults requires at least four nodes.

The safety of the TTP clock synchronization has been verified formally and experimentally. It can be guaranteed that: (i) the synchronization condition is fulfilled, and (ii) that any violation of this condition can be detected for all faults within the fault hypothesis. This guarantee is based on a formal proof.

Q: ***What is the difference between rate and offset correction?***

A: The clocks in a distributed system inevitably drift away from each other. The amount of drift is unpredictable and varies with temperature, voltage, and age of the crystals. Consequently, periodical resynchronizations of the clocks are necessary if a notion of common time—synchronization—is to be established.

There are two ways to resynchronize clocks. One way is to periodically check the difference and adjust each clock so as to reduce the offset among them as much as possible. This method is called ‘state’ or ‘offset’ correction, and has to be repeated periodically. An example of this type of correction is the way we correct our wristwatches every so often. This kind of correction mechanism is a simple and robust one.

Another way to resynchronize clocks is to periodically check how fast the clocks drift away from each other, and adjust the rate of drift so that the times displayed stay “closer together.” This method, called ‘rate’ correction, is akin to cruising behind another vehicle and periodically adjusting one’s own speed to that of the vehicle in front so that the distance in between remains approximately constant.

These two mechanisms can be combined or alternated to suit different situations.

In TTP, the clocks of the communication controllers are synchronized during runtime by offset correction only. The reason is that for this mechanism—in the specific form that is employed in TTP—a formal-mathematically proven assertion exists that no single fault in the system can bring the synchronization algorithm into an instable state.

Offset correction is available in TTP in two ways, indicated below. It should be noted that neither of these is part of the runtime synchronization mechanism inside the TTP protocol, but needs support from the application.

1. External Rate Correction: Adjustment of the whole TTP cluster drift to an external clock, such as a GPS receiver. This does not improve synchronization among clocks within the TTP network; it keeps the network synchronized with reference to the external time source.

2. Clock Calibration: Static calibration of each TTP controller in the network. The drift rate measurement is performed before system runtime, and an appropriate adjustment value is stored in the TTP configuration data (MEDL). This minimizes the static amount of drift between the clocks, and is a suitable mechanism for dealing with drift that results from the aging of clock crystals. It does not, however, account for changes in drift during the drive cycle (e.g. temperature).

Q: ***How much efficiency does TTP lose by not using internal rate correction for clock synchronization?***

A: Rate correction offers optimized inter-frame gaps, i.e., the smallest possible gaps between any two subsequent transmissions in a TDMA network. TTP does not utilize rate correction – how much efficiency is lost due to the resulting longer inter-frame gaps? This question is best answered with an example.

Assume a network of 25 ECUs, each of which sends 50 bytes per round. The duration of a round is 2 milliseconds, and the data rate is 10 Mbit/s. The busload for the network is 50%. Further, assume a worst-case drift of the individual clocks of +/- 1000 ppm. This could lead to drifts of as much as 4 microseconds per round if no rate correction is performed. With a safety margin for measurement and propagation delay jitter of over 100%, we can assume that 10 microseconds would make a reasonable precision requirement for this network.

10 microseconds represent half a percent of the round. That would be far worse than the value that could be achieved with rate correction (e.g. 1 microsecond).

The overall loss of efficiency depends on how often this interval of 1 or 10 microseconds occurs during the round. In a network where each node sends exactly once, we would get 25 intervals for 25 nodes: 250 microseconds or 12.5% of the round. We could take four times as much “gap” (so as, say, to prevent noise on the line that lasts longer than 10 microseconds from affecting two subsequent transmissions) and we would still have 50% of the round for data transmission: 25 ECUs, each of which sends 50 bytes.

The loss of efficiency therefore depends on the number of transmissions per time, which again corresponds to the data length of these transmissions. As shown in the above example, even a rather large precision still allows for a high data efficiency.

Q: ***Does the TTP clock synchronization tolerate multiple faults?***

A: The membership mechanism of TTP is capable of detecting any kind of communication fault that is not already detected and handled by other means. This makes the TTP clock synchronization very robust. The membership mechanism removes faulty nodes from the clock synchronization algorithm, thus ensuring that only consistent nodes are used for clock

synchronization. In this respect, TTP tolerates even multiple faults, as long as they do not create other system-level problems. That is very likely because most functions do not have triple redundancy.

5. Flexibility

Q: ***Can TTP send event-triggered messages?***

A: TTP can send only time-triggered messages, but an application using TTP can send both time-triggered and event-triggered messages. The transmission of event-triggered messages is performed over an event channel (bandwidth is reserved for event transmissions inside the TDMA slots, and the messages use identifiers). A typical event channel mechanism is a CAN emulation, in which a CAN-compatible interface is provided and the CAN messages are transmitted inside TTP data frames. For a 500 kbit/s CAN, such an emulation would require about 5% of a 10 Mbit/s TTP system.

Composability of the TTP system is maintained when using event transmission in this way because bandwidth is not arbitrated among different nodes (as in CAN or Byteflight)—but only among different functions within a node. Timing and bandwidth analysis for event transmissions is therefore done on a per-node basis and does not need system-level design.

Q: ***How can typical bandwidth-intense event transmissions like diagnosis and download be performed in TTP?***

A: Diagnosis data, like diagnosis requests and answers, are typically transmitted over event channels. A diagnostic channel for common diagnosis protocols requires about 1% per node of the net bandwidth of a 5 Mbit/s TTP system. This bandwidth is protected by the bus guardian, just like every other transmission.

For download and end-of-line programming, TTP offers a dedicated download mode. This mode requires a special programming device, a Download Master, and is performed in a simple master-slave fashion, utilizing the complete bus bandwidth for flash programming or similar. This is a maintenance functionality, which does not affect, or require, time-triggered operation.

Q: ***Is the length and data size of each TDMA slot individually configurable?***

A: Yes, the duration of slots and the number of bytes per slot are configured individually in the MEDL. The duration of a node's slot in relation to the length of the TDMA round represents the share of bandwidth that this node "owns" in the network. This allows flexible configuration and optimization of bandwidth usage.

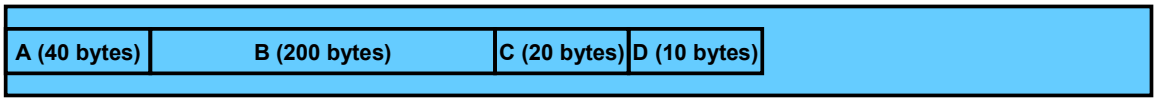

Figure 7: Configurable slot size with TTP;
unused frame space reserved for future expansion

Q: ***Can the two channels be used for different data?***

A: Yes, TTP allows sending data on both channels (redundantly), or just on one channel (in which case more bandwidth is available). Any amount of "mixing" of redundant and non-redundant data is possible. Of course, fault tolerance is not provided for non-redundant data.

Q: ***Can nodes be connected to only one channel?***

A: In principle, this is possible, even for consistent TTP slots. It is, however, not recommended because it causes application-specific restraints to the single fault hypothesis at the protocol level. Spatial proximity faults can be more efficiently handled by using a TTP star architecture. Consistency between nodes not connected to the same channel is not supported as it contradicts the principle of consistent communication among all nodes.

Q: ***Why can a node not send several times during one round?***

A: A TTP round is defined as the shortest interval between two sending slots of any node. A simple and thus safer bus guardian design and the verifications of safety mechanisms are the most prominent reasons for this definition of a round: it must be guaranteed that each node can make an influence on acknowledgment and clock synchronization at most once per round in order to validate the "single fault" hypothesis. If nodes need a slower rate of retransmission, this is possible using the "multiplexing" feature. Since the size of each TDMA slot in TTP can be configured individually with up to 240 bytes per slot and nodes with slower retransmission requirements can be defined as "multiplexed," there is no reason for multiple sending slots in TTP.

6. Configuration Data (MEDL) and CPU Interface (CNI)

Q: ***Does the MEDL (message descriptor list) have to be changed on all ECUs in the network when the communication design is changed?***

A: No, not generally. Since the MEDL contains only information about slot sizes, and not data, it is only when a slot size is changed that the communication tables need to be updated—and therefore the MEDL. Not-yet-allocated, and therefore empty, slots are reserved for future ECUs. If an ECU is added, then only nodes that receive data from the new ECU have to be updated. As a result, only component testing is required before a new ECU is integrated into the system. The overall communication behavior is unchanged because the MEDL remains the same. Changing the MEDL, however, would necessitate a system retest.

A change in the MEDL would also be necessary if there is insufficient or no bandwidth left for new functionality. In such a situation, the remaining bandwidth at existing nodes must be combined to provide the additionally required bandwidth. This requires a complete, but probably quite small change in the system communication timing. Obviously this problem of "updating a nearly 100% full system" is a general issue and not directly related to TTP.

Q: ***Does the use of a MEDL address the necessity of providing identical communication interfaces between platforms and suppliers and support for changes without reprogramming of nodes that are unaffected by a change?***

A: Yes. This is because the MEDL defines only the length of slots, not the contents of slots. The contents of a slot (its size, position, and the packing of signals inside data frames) are defined by a separate middleware layer (OSEKtime FT-COM). A suitable platform strategy can be specified accordingly.

Q: ***Do all communicating nodes have to have all communication information (the complete communication matrix) in their MEDLs?***

A: No, the MEDL contains only information about the slot lengths, not about the contents of the individual slots. Consequently, multi-vendor projects with restricted information about the communication interfaces are supported by this MEDL architecture.

Q: ***How much memory does the MEDL require?***

A: Experience has shown that the memory needed for the MEDL is not an issue. Very complex systems may have about 3 kB, but such systems will always have a large portion of software to pack and unpack the messages in a node, and the MEDL can be seen as being part of this software. For other communication systems, the information contained in the MEDL (communication controller configuration driver) is stored in the non-volatile memory of the CPU instead.

There is no restriction on the structure and size of the MEDL in the protocol specification, as well as no restriction on the implementation of MEDL memory (RAM, Flash). Silicon manufacturers are therefore free to optimize their MEDL design for best efficiency and cost.

Q: ***Does the CNI (communication network interface) require the CPU to access all communication data synchronously?***

A: No, asynchronous access is possible using the non-blocking write protocol. Typically, a separate middleware layer (OSEKtime FT-COM) performs the interfacing to the CNI, thus removing the need for the application software to do any direct CNI access.

Q: ***Why does TTP put received data into memory locations defined by the MEDL, and not into message buffers using identifiers, like CAN?***

A: For two reasons: memory efficiency and fault detection.

- ***Memory efficiency:*** If transmissions can be up to 8 bytes long, as they are in CAN, not very much memory is wasted if a single byte message is received in an 8-byte buffer. If transmissions can be up to 200 bytes long, buffers will have to be 200 bytes long. However, many transmissions contain only a couple of bytes.

  The available communication memory (CNI) in TTP can be allocated in any pattern, regardless of the length of transmission. This allocation is configured in the MEDL. In a 2 kB memory area, for instance, there can be as many short or as many long buffers as are needed, so long as they all fit within 2 kB. If they do not, overlapping structures can be defined, in which data is overwritten at known intervals. So a TTP CNI is never “overfull.” More CNI memory simply means fewer CPU activations to get the data out—and therefore more time for application tasks.

- ***Fault detection:*** When using identifiers, the problem of masquerading can occur: Some ECU in the system transmits an identifier that has not been assigned to that ECU. By adding check data into the transmission itself (and taking up some bandwidth as well as CPU time at the sender and receiver side), it can reliably be checked whether the message is from the sender that the sender claims to be. The problem is that it cannot usually be detected who the faulty sender is in the case of a masquerading fault. Any ECU could be the faulty one.

  TTP does not use any identifiers at the protocol level. All transmissions are identified only by the time of transmission. Identifiers can be included in the transmissions to signal message content, but the sender cannot masquerade as another. It is always 100% certain which ECU sent a specific transmission, correct or faulty.

Q: ***Why does TTP not use buffer management?***

A: Buffers are feasible only for short data packets. With larger buffers (say 100 bytes), the buffer memory would be unused for most of the time since most data packets would be much shorter than the maximum length. In TTP, each buffer would need to have 240 bytes, especially if double buffering (for atomic read operation) and a reasonable number of buffers (at least 16 or

32) are to be supported. The TTP CNI uses address-mapped data frames instead, and therefore allows a perfect optimization of the memory usage of data packets.

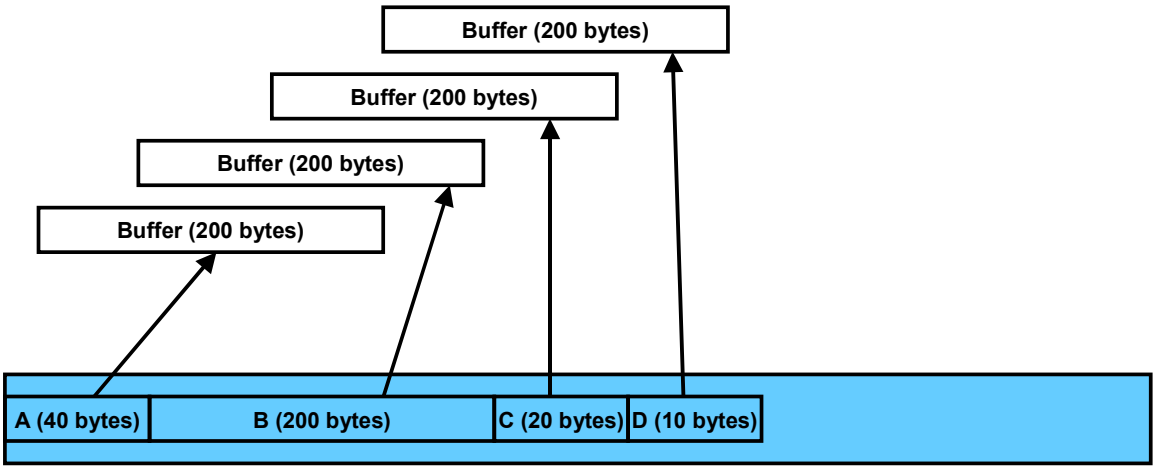

Figure 8: Buffer management

Furthermore, the update rate for buffers may vary strongly, while the update rate of CNI frames is constant and known. If a buffer has not been read when a new matching message is received, the buffer contents have to be overwritten or the new data will have to be discarded. In TTP, the newest data for any location in the CNI DPRAM memory is always available, the CPU is not affected by a large interrupt load (especially on fast communication systems with short rounds), and no data will be lost even in the worst-case load scenario.

7. Operation of Partial Networks and Start-Up

Q: ***Do the fault tolerance mechanisms of TTP cause problems if only parts of a network are present, as is often the case during tests?***

A: Typically, they do not cause any problem. The distributed clock synchronization is the only mechanism that might be susceptible to problems for this reason. That is because it requires a working subset (at least two nodes) of the “actively synchronizing” set of nodes in the full network to be present. This requirement of the distributed clock synchronization can be fulfilled with rest bus simulation hardware (TTP nodes sending in the slots of the missing nodes) if necessary.

Q: ***Can TTP start up with noise on a channel?***

A: Yes, this is possible on a bus. When an intelligent star architecture is used, both channels can be noisy. Startup can still be performed, however, provided the noise does not destroy all communication.

8. Physical Layer

Q: ***What dedicated TTP physical layers are available?***

A: TTP has been tested on several existing physical layers. Most systems in use today have high-speed CAN transceivers or RS485. Except for MOST, no physical layer is currently available that supports automotive requirements at data rates higher than 1 Mbit/s. This applies to all communication protocols in use today. The physical layer specification for TTP has now been finalized. This specification is a result of the TTA-Group Physical Layer Working Group.

Q: ***Does TTP support electrical and optical data transmission?***

A: Yes, a system has been tested in which one channel was made of copper and the other of glass.

Q: ***What is the difference between a dedicated TTP physical layer and RS485?***

A: The dedicated TTP physical layer differs from RS485 in the following features:

- Defined bus idle behavior without bus biasing
- Excellent common mode behavior (-12 / +20V) and high speed
- Fail-silent behavior when exceeding the permissible common-mode range
- Wake-up function and sleep mode
- Built-in diagnostics for electric bus faults
- Compatible with 42V power supply (PowerNet)
- Adaption to communication controller supply voltage (2.5V, 3.3V, 5V)

Q: ***What are the requirements for a dedicated TTP physical layer?***

A: The finalized specification takes into account all timing and level requirements for a dedicated TTP physical layer. As to the environment requirements, the same guidelines obtain as for existing CAN networks. Additionally, higher safety and support of 42V power supply are included in the specification.

Q: ***Was any testing done under close-to-real conditions?***

A: Yes, e.g., the simulation and test setup of an aircraft bus architecture. This test involved various cable lengths (50 and 100 m) and investigated the aging effect on connectors. All functional and EMI/EMC tests, including lightning upset tests, were passed in accordance with RTCA DO-160D.

Q: ***What EMI/EMC requirements were included in the design of a dedicated TTP physical layer?***

A: The EMI/EMC requirements included in the design of a dedicated TTP physical layer were as follows: ISO-pulse-test (ISO7637-1, ISO7637-2, ISO7637-3), ESD protection, EMI, and EMC. Additionally, many environment requirements from the automotive industry were also taken into consideration.

Q: ***Does TTP support energy management like shutdown and wake-up?***

A: Yes, the physical layer supports these functions.

Q: ***Is wake-up also supported by standard drivers?***

A: Yes, in principle. Standard drivers such as high-speed CAN use dominant bits in the data stream for detecting wake-up. Due to MFM encoding, TTP supports wake-up only up to a certain data rate, depending on the component used.

TTA-Group

Contact

TTA-Group Secretary
Schoenbrunner Strasse 7
A-1040 Vienna, Austria
Tel.: +43 1 585 34 34-74
Fax: +43 1 585 34 34-90
E-mail: secretary@ttagroup.org
Web: www.ttagroup.org