Time-Triggered Technology
TTTech Computertechnik AG

Comparison
CAN vs. Byteflight vs. TTP/C

Edition 2-Apr-2001

TTTech Computertechnik AG
Schönbrunner Straße 7
A–1040 Vienna
Austria

voice: + 43 1 585 34 34–0
fax: + 43 1 585 34 34–90
web: http://www.tttech.com
email: office@tttech.com

1. Basic Protocol Properties

	CAN (specification version 2.0 B) www.can-cia.de	**Byteflight** (specification version 0.5) www.byteflight.com	**TTP/C** (specification version 0.5+) www.tttech.com
1.1 Media Access Strategy	Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)	Minislotting	Time Division Multiple Access (TDMA)
1.2 Mechanism to resolve transmission concurrency between nodes	non-destructive bit arbitration on the physical layer, using unique priority identifiers for each message	Design-time determined, application-driven minislotting scheme (bus-sensed detection and preemption of empty slots), using unique slot IDs for each telegram	Design-time determined, time-driven slot scheme using a global time-base and a static global communication schedule, no collision or concurrency occurs during runtime.
1.3 Gross Data Rate and Physical Layer Requirements	up to 1 Mbit/s using inexpensive twisted pair. Due to bit arbitration mechanism, high speeds/long distances are difficult to achieve, and signal shape must be well specified.	10 Mbit/s on a specially-designed optical layer; other speeds and physical layers allowed, but not required and not implemented.	arbitrary speeds, on arbitrary physical layers that allow edge detection (already in use: RS485, highspeed CAN, optical); data rates of up to 2 Mbit/s required, higher speeds allowed, but not required by specification. Expected max. rate supported by the next generation of controllers (avail. 2001) is 5 and 25 Mbit/s.
1.4 Network topology and size	Bus, number of participants and length of bus limited by arbitration mechanism and electrical limitations, long propagation delays not allowed. Typ. 2-20 nodes.	Star (bus suggested but not detailed), limit of participants defined by star coupler technology (current implementation allows up to 22). Long propagation delays allowed, but must be known and decrease performance. Typ. size unknown.	Bus, star, or any combination thereof, limit of 64 active participants (= max. size of membership); long propagation delays allowed, but must be known to some extent (in microsecond range), and decrease performance. Typ. 4-40 nodes.

2. Features

2.1 Message transmission/ buffers	Controller-dependant amount of data (current implementations: ca. 120 bytes), transmitted and received according to application request and possibly delayed by bus access.	Controller-dependant amount of data (current implementations: ca. 240 bytes), selectable as state message buffer or queue. Synchronous (state) messages are transmitted and received according to the ID scheme at the beginning of the cycle (currently 250 µs), asynchronous (queue) messages according to the ID scheme in the rest of the cycle.	Controller-dependant amount of data (current implementations: ca. 2 kbytes), transmitted and received according to a static communication schedule and stored in state message buffers ('temporal firewalls'). The sequence of message buffer transmission and reception repeats with each cluster cycle in a fully predictable fashion, controlled by the global time.
2.2 Global timebase (How are the clocks of the individual nodes in the cluster synchronized?)	None	By SYNC pulse from bus master node, granularity t_cyc (in current implementation fixed with 250 µs). No explicit time value provided by protocol controller.	By fault-tolerant distributed algorithm, granularity configurable in 1-10 microsecond range. Explicit global 16 bit time value provided by protocol controller.
2.3 Redundant channels	No	No	Two, replica determinate

2.4	Membership service (Which of the nodes in the cluster are currently alive?), max. age of membership information, consistency in a malicious (a.k.a. byzantine or asymmetric) fault scenario (typically introduced by faulty line drivers or clocks)	Not supported by protocol	Not supported by protocol	Explicit membership bitvector, max. age one TDMA round (except for multiplexed nodes). Consistency enforced after two TDMA rounds in a worst-case malicious fault scenario by removing the minority of disagreeing nodes.
2.5	Message acknowledgement (How does the sender know if the message was correctly received?)	Explicit by retrying transmission and error frames, communication timing is affected by acknowledgement mechanism.	None, strategy is frequent retransmission without acknowledgement.	Implicit by including the C-state in the frame CRC calculation, communication timing is not affected by acknowledgement mechanism.
2.6	Composability	None	Application dependant	Always

3. Performance

3.1	Data Efficiency vs. Latency	Latency increases with load. Cannot prevent overload of the communication system. Typical max. efficiency ca. 30%, but this very much depends on the application. Predictability decreases strongly with increasing load.	Latency of asynchronous messages increases with load, synchronous messages are retransmitted every cycle of t_cyc, currently 250 µs. Max. efficiency ca. 54% mostly due to transmission overhead.	Latency is constant and precisely known at design time. Max. efficiency depends on controller implementation (so-called 'inter-frame gap'), theoret. limit around 95%, realistic systems will have around 60-80%. Note: Efficiency is higher at lower bit rate.
3.2	Net amount of application data transmitted per transmission	1-8 bytes per message	1-12 bytes per telegram	1-16 bytes per frame (will be 1-240 bytes per frame in next specification version)
3.3	Per-Transmission Overhead (headers, identifiers, CRC, start and stop patterns)	62 bit + variable amount of stuff bits	40 bit + 2 start-stop bits per data byte, plus propagation delay	23 bit (will be 31 bit in next specification version), plus inter-frame gap (including propagation delay)
3.4	Other overhead	Bus load must be kept reasonably low to avoid thrashing in a peak load scenario	Minislotting delay including propagation delay with each minislot, SYNC pulse	Currently unused bandwidth reserved for application (can range from 0 to 'very large')
3.5	Hamming distance of transmission (if less than this number of bits is disturbed during transmission, the error can definitely be detected by the CRC check, otherwise there is a small chance that the error goes unnoticed)	6	6	6

4. Flexibility and Extensibility

4.1 Distribution of bandwidth among the nodes and protection against misuse by a node during runtime	Priorities are distributed at design time by assigning unique identifiers. Within these, bandwidth use is fully controlled by application; therefore fully flexible, but no checks or protection possible by the communication system.	Priorities are distributed at design time by assigning unique telegram IDs (each allows to transmit up to 12 bytes) to each node. Within these, bandwidth is divided into 'synchronous' and 'asynchronous' telegrams, the latter allow flexible bandwidth use. Communication system cannot check if only assigned IDs are used by application, application faults can therefore result in excess bandwidth usage by a node.	Bandwidth is distributed at design time by assigning frames of specific length (1-240 bytes) to each node, communication system controls this. Application faults cannot result in excess bandwidth usage by a node as only the assigned amount of data is transmitted at the assigned times (this is autonomously controlled by the communication system and guarded against timing faults by the bus guardian unit).
4.2 Strategy to handle future expansion of the system and effect of expansion of a (safety-critical) system if this strategy is used.	Keep average bus load as low as possible. After expansion, system must be completely re-tested with regard to communication timing.	unknown (presumably a mixture of cluster-wide planning of synchronous telegrams plus subcluster-local allocation of asynchronous telegrams, then running a statistic analysis on the whole message set to determine if latency requirements for the asynchronous messages are fulfilled)	Define communication requirements in an off-line tool (TTPplan). This results in a TTP/C communication schedule which requires a certain percentage of the available net bandwidth TTP/C can deliver on the specific setup. The remaining bandwidth is statically assigned for future expansion of specific existing nodes, and/or nodes to be added at a later time.
4.3 Effects of adding a message that was NOT planned for in the original communication design (if bandwidth for it WAS allocated, e.g., by sending empty dummy messages, there are no changes to the communication in any of the protocols)	New message identifier is declared to sender(s) and receiver(s), other application nodes are unaffected. Effects of the added message on the communication timing are difficult to predict, system must be completely re-tested/re-validated.	New telegram ID or new length of existing telegram is declared to sender(s) and receiver(s). If it is a synchronous message, other application nodes may need to know about the changed latency of the existing synchronous messages (the lower-priority telegrams now arrive later). The statistical latency analysis for asynchronous messages must be re-run to establish that they can still be transmitted within the given latency limits.	New firewall information is declared to sender(s) and receiver(s). The communication (FT-COM) layer and the schedule data (MEDL) in each node must be updated. The off-line scheduler tells if the new system is feasible and what the new guaranteed message latencies are. In TTP/C applications, reserving bandwidth for messages (by allocating frames longer than necessary) is a common and recommended strategy.
4.4 Effects of adding a node that was NOT planned for in the original communication design	Same as above; adding a node in CAN is the same as adding new messages, maybe except regarding physical layer issues.	New telegram ID(s) are declared to the receiver(s), existing telegrams do not get longer. Otherwise same as above. Application level fault tolerance strategies (e.g., the intelligent star coupler shutting off faulty nodes) must be updated.	New firewall information is declared to the receiver(s), except if the new node only transmits firewall messages already present in the system (i.e., a redundant node). The communication (FT-COM) layer and the schedule data (MEDL) in each node must be updated. TTP/C allows to explicitly reserve slots for future expansion.
4.5 Regardless of the communication protocol used, an increase in the amount of transmitted data (within the physical limits of achievable net bandwidth) that was not anticipated by explicit means (e.g., dummy messages, reserved idle time on the bus) will change the communication timing, usually changing the *application timing* too. As such a change dictates a high amount of re-evaluation (typically tests and validation procedures) for safety-critical systems, explicitly anticipating future system expansion is a good idea regardless of the communication protocol used, as it can save a lot of money and effort.			

4.6	Support for large systems and synchronization between clusters/to external clock, support for re-integration on a global time-base within or between clusters.	Arbitration mechanism restricts size. Gateways can easily exchange messages between clusters, but latency bounds for gatewayed messages are very difficult to establish and depend on the delays in both clusters. Replica determinate gateways (active or shadow) difficult to implement. Clock synchronization is not supported by protocol, must be handled completely on the application level if needed by the application.	Gateways can easily exchange messages between clusters. The SYNC pulse generated by the bus master node provides a global timebase with fixed granularity of t_cyc (currently 250 us). Synchronization between clusters currently not possible. Re-integration on a global timebase must be implemented by application messages.	External clock synchronization possible, with a precision in the microsecond range; gateway nodes act as time gateways. Replica determinate gateways (active or shadow) very simple to implement. Also feasible over a chain of clusters, overall precision decreases linearly. Re-integration on a global timebase supported by communication system.
4.7	Support for global change of system mode, consistency (how is it guaranteed that all the nodes change to the new mode) and latency of mode change (time from the detection of the need to change the mode until the system runs in the new mode, assuming that all relevant nodes can actually react so fast)	Not supported by protocol and not widely used. If used, system mode changes must be signaled by application messages, no consistency mechanism provided by protocol. Latency for mode change depends on system load and message priority (usually high), probably less than 100 µs.	Two global modes supported, by broadcast of ALARM state and removal of ALARM. Sent by bus master and detected by all nodes. On the BF-specific physical layer, two independent units (controller and optical driver) detect the mode. No global mode change feature on protocol level for non-bus master nodes (they must ask the bus master to raise the ALARM with an application message, presumably with high priority). Latency < t_cyc (currently 250 µs) for bus master, application-dependant for non-bus master nodes (if sent in a synchronous telegram, probably < 2*t_cyc)	Up to 30 global modes supported, by 'Mode change request' and 'Clear Mode change request'. Can be requested by any node out of a user-specified set of nodes. Execution of a mode change is globally synchronized by communication protocol. Up to 3 application-specific mode changes possible from current mode; static design exactly controls which node may request which mode change at which time. Latency < cluster cycle (typically some ms).

5. Error Containment and Fault-Tolerance

5.1	Communication channel redundancy	Not supported by protocol. If implemented on application level, no message or membership consistency provided.	Not supported by protocol. If implemented on application level, no message or membership consistency provided. Synchronous timebase over two channels currently not supported by protocol.	Two redundant channels, message and membership consistency between them guaranteed by protocol. Synchronous timebase over both channels is part of protocol.
5.1	Failure mode of communication system, scope of error containment	Nodes are fail-uncontrolled, babbling idiot and timing errors possible in the whole system.	Nodes are fail-uncontrolled, errors will propagate though system; if detected (requires star architecture and application level diagnosis functionality in the star coupler), they can be contained after propagating through the system for some time (fail-restrained).	Nodes have fail-silent timing and fail-consistent data transmission (enforced by bus guardian and implicit acknowledgement, respectively), error containment is performed at node level.
5.3	Support for replica determinism (several nodes perform the same computations in parallel and produce equal results unless a fault occurs in one of them)	None	None	Global timebase, consistent state information (includes time, mode, membership), consistent message acknowledgement, cliques (nodes with inconsistent state information or message sets, result of a byzantine fault) are removed within a bounded time interval

5.4 Support for transparent shadows (a node which takes over the job of a failed node without the rest of the system noting the difference)	Not supported by protocol. If implemented on application level, shadow application must figure out whether to become active (must prevent multiple activation of shadows to avoid double message transmission and overload of communication system)	Not supported by protocol. If implemented on application level, risk of telegram collisions (protocol requires unique telegram IDs for each node) which are not handled by protocol fault hypothesis.	Protocol provides mechanism to detect node failures fast (membership) and for collision-free integration of exactly one shadow in case more than one are ready to replace the failed node
5.5 Protection against Babbling Idiot (due to a fault, a node or line driver monopolizes the line, preventing any communication)	None	Not supported by protocol. If implemented on application level, star coupler can shut off a node (requires identification of failed node by application in star coupler); babbling star coupler not covered. No known protection when using a bus topology.	Independent bus guardian unit (bus topology: on each node, star topology: on each node or in star coupler), which cannot actively transmit but can prevent transmission.
5.6 Node-local error detection mechanisms for application or host CPU faults	None	None	Lifesign challenge-response protocol, concurrency check (NBW protocol), mode change request permission check, firewall status check
5.7 Node-local error detection mechanisms for controller faults	None (or implementation specific)	None (or implementation specific)	Controller built-in self test, Bus Guardian
5.8 Suggested methods for achieving application level fault tolerance	---	---	Active replication or shadowing (hot spares) of inexpensive, fail-silent nodes, protocol supports replica determinate programming with consistent state and message set information.

6. Pro and Contra

6.1 Pro	Highly flexible, widely available, inexpensive and easy to use. Available today as integrated module on several CPU cores.	High precision of central master clock synchronization. Flexible bandwidth distribution between non-critical messages.	Formally verified mechanisms for high-level fault tolerance (clock synchronization, membership, message agreement). Strong error containment on node level. Runs on several existing physical layers (high speed CAN, RS 485, optical). Off-line communication design yields guaranteed latency for all messages in the system.
6.2 Contra	Does not offer reliable mechanisms to build fault-tolerant, safety-critical systems. Systems are difficult to test for fault coverage and nearly impossible to certify. Very limited capability for higher data rates.	Single points of failure exist. Failure modes of bus master/backup concept and redundant channels not known. No acknowledgement mechanism provided. Fault tolerance strategies rely heavily on application.	Unplanned extensions require update of all nodes. Data efficiency limited by the IFG of the slowest controller in the cluster, which must be known at design time.

Your browser doesn't support JAVASCRIPT.

Click here: *byteflight* Homepage