

Performance Analysis of Byteflight Networks

Gianluca Cena
IEIIT-CNR
C.so Duca degli Abruzzi, 24
10129 Torino – ITALY
gianluca.cena@polito.it

Adriano Valenzano
IEIIT-CNR
C.so Duca degli Abruzzi, 24
10129 Torino – ITALY
adriano.valenzano@polito.it

Abstract

Byteflight is a high performance communication network conceived for the use in passenger vehicles, which features the advantages of both the synchronous and asynchronous transmission schemes. Its development was driven by the new needs of automotive systems, which could not be satisfied by the existing solutions.

In this paper the main features of Byteflight are analysed – pointing out its advantages and drawbacks – and its performance is evaluated for the different kinds of communications that have to be supported in today's vehicles as well as in advanced automated manufacturing systems.

1. Introduction

At present, *controller area network* (CAN) [1] is by far the preferred technology for interconnecting devices in passenger vehicles. This solution has a number of advantages. Besides its simplicity and good degree of efficiency, it is very stable and CAN chips and development tools are widely available at (relatively) low costs. Because of these appealing features, it is also very popular in automated factory environments, where it is used as a field network.

However, CAN suffers from a number of drawbacks that will likely rule out this network for the use in sophisticated control systems of future passenger vehicles – initially in luxury cars and, in the following years, even in cheaper “budget” models. An example is given by the so called *x-by-wire* systems, where the driver does not interact directly with the physical components of the car any longer, by means of mechanical or hydraulic paths for steering, braking or gear shifting. Instead, all the information coming from the steering wheel, brake pedal or gear shift lever will be encoded in a stream of bits and managed by wires and chips.

CAN networks have serious performance limitations: the maximum allowed transmission speed, in fact, is 1Mbit/s at most (for bus lengths up to about 40m), and cannot be increased further because of the particular

medium access technique, which is based on bitwise arbitration [2]. Moreover, the CAN access protocol is not considered deterministic enough to be used in those systems that require high dependability. Even though the latter issue could be tackled by means of a modified version of CAN known as TTCAN [3], the available data rate makes it very hard to adopt CAN-based solutions in bandwidth-demanding applications.

Several communication protocols were developed in the recent past so as to provide solutions that fit in well with all the requirements of tomorrow sophisticated automotive systems. The *time-triggered protocol* [4], for example, exhibits a fully deterministic behaviour and is suitable for those systems that require very high dependability. Even though TTP was defined about one decade ago [5], chips featuring the required level of performance have appeared on the market only recently. The *Byteflight* protocol [6], instead, was first introduced by BMW and its development started in 1996. At present, communication controllers that comply with the Byteflight specifications can be found off-the-shelf and have already been used in production vehicles. More recently, the *FlexRay* protocol [7] has appeared, which is basically an evolution of Byteflight and also includes concepts borrowed from the time-triggered approaches. Once FlexRay products will be available off-the-shelf, it looks like Byteflight will hardly have a chance to survive commercially as a stand-alone product. However, FlexRay isn't considered explicitly in this paper because its final specifications are not yet publicly available at the time of writing.

As TTP, Byteflight features a throughput one order of magnitude greater than CAN and ensures a higher degree of determinism. However, Byteflight was conceived to provide a high level of flexibility, which resembles closely that achieved by CAN. TTP relies basically on a *time division multiple access* (TDMA) approach. It assigns the network capacity to the different nodes in a static and permanent way (allocation of the slots cannot be changed during the network operation). Thus, the addition of new nodes or new functions (i.e., new messages) requires reconfiguration of the whole system. By contrast, Byteflight enables sending messages whenever required, so no static slot allocation is needed.

It is worth noting that, besides automotive applications, the solutions described above are perfectly suitable to be used as high-performance fieldbuses too. While many papers appeared in the literature describing the features and the advantages of TTP, few of them dealt with Byteflight. In the following the main features of this protocol will be analysed, in the light of both their use in automotive systems and their possible adoption in the factory communications scenario.

The paper is structured as follows: section 2 describes the Byteflight protocol basics whereas in section 3 its behaviour is taken into account for several classes of data exchanges that can be found in automotive systems and in automated factory plants as well. Finally, in section 4 the real-time performance achievable in Byteflight is evaluated for some kinds of communications, and its advantages and drawbacks are pointed out.

2. Byteflight basics

Byteflight is conceived so as to provide the advantages of both the synchronous and asynchronous transmission schemes. In particular, very low jitters are ensured for the high priority messages, whereas the bandwidth allocation is highly flexible for the low priority data exchanges. The latter was felt as a key requirement by the protocol designers, in that the development period of automotive systems shortens as the time elapses, hence the ability is required to perform changes to the original design quickly and safely. For example, it should be possible to add new functions to the body electronics on-the-fly (maybe concerning some optional features of the vehicle) without affecting those sub-systems that require a high level of safety and have already been tested.

Byteflight relies on the so called *flexible time division multiple access* (FTDMA) technique. Transmissions take place cyclically, and each cycle is started by a *synchronization pulse* (*sync*) sent periodically by a special node known as the *sync master*. Sync pulses make available a common time-base to all the nodes in the network, and to the application tasks as well. Any Byteflight node can be configured to operate as the sync master. In current implementations, the interval between subsequent sync pulses is set to 250 μ s.

In Byteflight the access to the bus is based on the message identifier. Messages are sent over the network in increasing identifier order (i.e., message 1 is sent first, followed by messages 2, 3 and so on). This is achieved because each node maintains a slot counter, which is reset every time a sync pulse is read from the network. Slot counters begin at 0 and count up to the highest allowed identifier (255) – unless the end of the cycle is reached (that is, a new sync pulse is detected). Whenever the slot counter reaches a value equal to the identifier of a message waiting for transmission (i.e., already queued

for transmission), the message itself is sent over the network by the relevant node. As in CAN, no two nodes in the network can produce the same message (i.e., labelled with the same identifier). This is required so that unsolvable collisions cannot occur on the bus.

In order for the access mechanism to operate correctly, the slot counters of the different nodes have to be updated in a coherent way. This is achieved by increasing them on a minislot basis. In practice, every time the bus stays idle for a given period of time (much lower than the transmission time of a message), a minislot is detected by all the nodes and all counters are increased by one. The count is temporarily stopped when a message is being exchanged over the bus, and is restarted as soon as the transmission ends. In order to achieve the required level of synchronisation among the nodes, the timing requirements of the local oscillators in Byteflight are quite tight.

The resulting access mechanism resembles to some extent a virtual token passing schemes, such as, for example, that adopted in P-Net. However, Byteflight allows a highly deterministic behaviour in that the pace for transmission cycles is not decided by a distributed algorithm (token circulation), instead it is driven precisely by a single station (the sync master).

As in CAN, the Byteflight access mechanism ensures that messages with lower identifiers get precedence over those labelled with numerically higher values. However, each message can be sent at most once per transmission cycle. This means that, unlike in CAN, no Byteflight node is allowed to hog the whole available bandwidth, ruling out all the other transmitters. Such a feature greatly limits potential damages due to the “babbling idiot” problem – a faulty node that transmits high priority messages repeatedly.

2.1. Physical layer

Current implementations of Byteflight rely on optical links. Nodes are connected one to another by means of a star coupler, which merges the signals coming from the nodes by performing a logical “AND” function and distributes the resulting signal back to the nodes on its outgoing links. Optionally, it is possible to attach a device directly to the star coupler by exploiting electrical connections. This helps reducing propagation delays. In the same way, a more conventional bus topology based on copper wires can be adopted.

The signal can assume two complementary logical values, that is dominant (0) and recessive (1), respectively. At present a single transmission speed of 10Mbit/s is foreseen in the specifications.

2.2. Frame format

Each frame (called a *telegram* in the Byteflight specification) is preceded by a *start sequence*, consisting of 6 bits at the dominant level. Due to effects in the currently used optical transmissions, receivers must

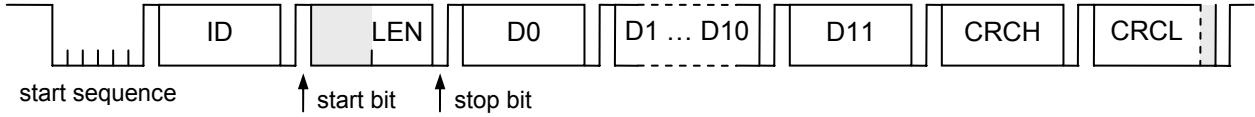


Fig. 1: Byteflight frame format.

recognise as valid start sequences consisting in up to 9 dominant bits.

Then a number of fields follow, each one encoded on one or more bytes. As depicted in Fig. 1, the identifier field (ID) is located just after the start sequence and encodes the message identifier in the range from 1 to 255. The length field (LEN) is next to the identifier and specifies the size in bytes of the data field in the range from 0 to 12. Only the 4 least significant bits of the LEN field are effectively used, so that the remaining 4 bits are available for other purposes (including future protocol extensions). Then there is the data field, made up of 0 to 12 bytes (D0 to D11). The frame ends with the *cyclic redundancy check* field (CRC), which takes the last two bytes (CRCH and CRCL, respectively). As a 15-bit CRC sequence is used, the least significant bit of CRCL is left unused and is set to the dominant value.

Each byte of the fields from ID to CRCL is preceded by a start bit at the recessive level and is followed by a stop bit at dominant level, in order to enable the synchronisation of receivers with the bit stream read on the bus. Unlike the bit stuffing mechanism adopted in CAN, this achieves fixed-length frame encoding and helps in reducing jitters. Nominally, the size of Byteflight frames ranges from 46 to 166 bits (corresponding to 0 and 12 bytes in the data field, respectively). Hence, at the nominal bit rate of 10Mbit/s the transmission time of a message varies from 4.6 μ s up to 16.6 μ s. This means that, depending on the size of the exchanged data and the message identifiers, Byteflight ensures several messages (roughly up to one dozen or more) to be sent in every transmission cycle.

This introduces the first major difference between CAN and Byteflight: CAN is a truly asynchronous protocol that implements a distributed non-preemptive priority-based communication scheme. In this case, the message with the highest priority can obtain, in theory,

all the available bandwidth, and thus it is the only transmission which is always granted to occur within a predefined maximum time. On the contrary, Byteflight is basically a synchronous protocol that allows prioritised asynchronous network accesses. Hence, it is able to ensure a privileged treatment to a number of messages at the same time.

2.3. Medium access technique

Byteflight relies on a FTDMA mechanism that uses counters for determining when messages have to be sent. A device can start transmitting a frame when there is a request pending at the beginning of the current transmission cycle and no activity on the bus is detected during the *frame waiting time* (T_{wx}). In order for the frame to be sent there should be enough time left before the end of the cycle – this is to avoid collisions with the next sync pulse. In practice, this implies that the time between the end of the frame to be sent and the rising edge of the next sync pulse should be at least T_{w0} .

The waiting time for a message (T_{wx}) depends on its identifier and the other messages already exchanged in the current cycle, as depicted in Fig. 2. For the first message after the sync pulse T_{wx} is equal to

$$T_{wx} = T_{wx0} + T_{wx\Delta} \cdot ID \quad (1)$$

whereas for the other messages it is computed as

$$T_{wx} = T_{wx0} + T_{wx\Delta} \cdot (ID - ID_{prev}) \quad (2)$$

where ID_{prev} is the identifier of the message that was previously exchanged over the network. From a practical point of view it is possible to use (2) in both cases by assuming that ID_{prev} is equal to 0 for the very first message.

The value of $T_{wx\Delta}$ should be the same for every node in the system, and is selected equal to the maximum propagation delay between any two nodes in the network, plus a safety margin to take tolerances into

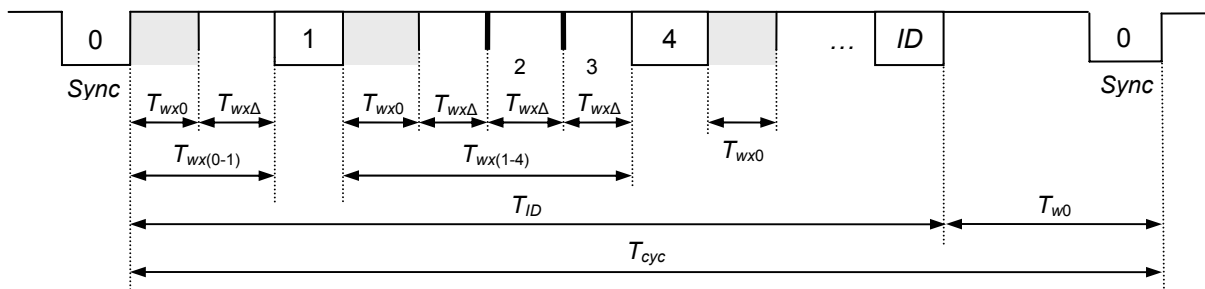


Fig. 2: Details on the timings of the FTDMA technique.

account. T_{wx0} , instead, is selected on a per-node basis and depends on the propagation delay on the link connecting the node to the active star coupler (i.e., on the distance from the centre of the network).

Actually, two different values are defined for T_{wx0} , depending on the fact that the last frame exchanged over the network was either sent or received by the node, that are indicated as T_{wx0}^{tx} and T_{wx0}^{rx} , respectively. This is aimed at ensuring that the counting of the multiplying component of T_{wx} starts at the same time for all the nodes.

2.4. Assignment of identifiers

Even though Byteflight doesn't specify how identifiers should be assigned to the different messages, there are a few simple design rules leading to the best results. In particular, we have to distinguish between time-critical messages, which must be sent within a well-defined maximum time and need very low jitters (strictly deterministic transmission), and messages that have not such tight requirements.

If messages have to be exchanged regularly in every transmission cycle (*synchronous* messages, such as, for example, those involved in time-triggered operations) there is no need to fully exploit the FTDMA technique. In this case the best solution is to assign messages consecutive identifiers starting from 1, so that the overhead due to the transmission of minislots is minimised. In the following we will refer to the Byteflight access mechanism used together with such an assignment scheme as *quasi-TDMA* technique (QTDMA).

The resulting behaviour of QTDMA is quite similar to that achieved by the TDMA technique, where each message is pre-assigned a specific (fixed) slot. As long as high priority messages are sent regularly in every cycle, they suffer from virtually no jitters – in fact, synchronous messages in QTDMA are sent back to back over the bus and are interleaved by a minimal fixed-length idle time. However, unlike pure TDMA networks, should it happen that a message is not exchanged in a given cycle the following ones are sent a bit early – that is, they experience a (small) jitter. This is because each

missing message is “replaced” by a minislot, which is certainly shorter than the message itself.

On the contrary, the exchange of messages whose schedule is not known in advance or, at least, that don't take place in sync with the common pace given by the network transmission cycle (*asynchronous* messages, such as, for example, those involved in event-driven interactions), relies entirely on the peculiar features of the FTDMA scheme. In this case messages are selected for transmission according to their priority – which is the same as the identifier, as in CAN.

As synchronous messages are usually characterised by more tight requirements than the asynchronous ones, they are assigned lower identifiers. The resulting identifiers assignment in a typical Byteflight network is depicted in Fig. 3.

3. Communications in complex control systems

Networks conceived for complex control systems, such as those involved in today's passenger vehicles, have to support several kinds of data exchanges at the same time. Even though solutions based on some interconnected networks can be used for satisfying the different needs in the best way, relying on fewer networks (hopefully, only one) may reduce both the design and production costs. In this case, in fact, there is no need for intermediate gateways, and all information are made directly available everywhere in the system.

In the following, the typical communication needs of such a kind of systems are briefly analysed, and the way they can be satisfied in Byteflight is described.

3.1. Hard-real-time exchanges

Controlling time-critical operations, such as those involved in x-by-wire systems, requires hard-real-time communications. In this case, in fact, the lack of determinism in the communication sub-system may lead to damages to the vehicle and even injuries to the passengers.

Today it is felt that the most suitable way for dealing with such a kind of requirements is the time-triggered approach: all the operations in the system (task

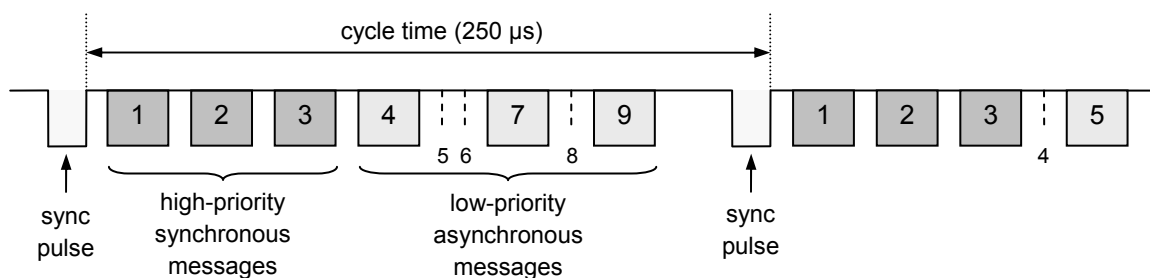


Fig. 3: Allocation of identifiers to synchronous and asynchronous messages.

activations, message exchanges, actuation and sensing on the physical system) take place at precise points in time, which are known in advance by the control algorithm(s). Hence, even the absence of a message at the time when it should have been received carries useful information.

Hard-real-time (HRT) messages involved in time-critical operations must be exchanged deterministically, with virtually no jitters. This means that they should rely on a synchronous communication scheme, which enforces strictly co-ordered data exchanges. The access mechanism best suited to the communications needed in a time-triggered system is with no doubt TDMA, such as that adopted by TTP.

Besides networks based on the TDMA approach, also the Byteflight protocol can provide such a kind of behaviour, by assigning HRT messages subsequent identifiers starting with the lowest value (this corresponds to the QTDMA synchronous transmission scheme described in the previous section).

In order to grant deterministic exchanges, all the scheduling details of HRTs have to be known in advance. In particular, they must fit in the transmission cycle. If this requirement is not satisfied, in fact, the transmission of a message could be delayed to the next cycle, and communications could no longer be deterministic.

It is worth noting that, in such a condition, the assignment of identifiers to the HRT messages has a limited influence on the transmission latencies. In fact, the key point here is that all HRT messages must always be sent within the current cycle.

Instead, the assignment scheme may affect transmission jitters in a sensible way. In theory, if all the HRT messages are transmitted in every cycle there can be no jitters. From a practical point of view, however, transmission errors may occur in the network and (faulty) nodes might occasionally fail in sending their messages. In this case, the Byteflight access scheme ensures that the lower the message identifier the shorter the jitters it may experience.

3.2. Soft-real-time exchanges

A second kind of communications concerns the so called *soft-real-time* (SRT) messages, used by body electronics, engine control, and so on. The timing requirements on SRT data exchanges are not so tight as for HRTs, in that they aren't strictly related to safety issues. Nonetheless, SRT messages are involved in controlling a physical system and so they have real-time requirements too.

As SRT messages can be created spontaneously by devices, the generation pattern is asynchronous and mostly unpredictable – this is the price that has to be paid for flexibility. Hence, from a general point of view, it is not possible to ensure that all of them be exchanged immediately as they are generated (i.e., right in the cycle

that follows the transmission request). The best approach in this case is to assign each message a specific priority and follow a best effort policy, which grants bus access to the pending message with the (network-wide) highest priority – as it happens in CAN.

When either asynchronous SRT messages are generated according to a periodical pattern (this is the behaviour of many electronic control units that are used in automotive systems) or the minimum inter-arrival time of each message is known in advance, a number of techniques exist [8, 9] for evaluating the worst-case transmission times. In such conditions, the rate monotonic and deadline monotonic approaches can be used for finding a suitable (fixed) assignment of priorities to the messages so that their deadlines are always met (provided that a feasible schedule of the message set exists). In all the other cases, however, the maximum latencies for SRT messages cannot be estimated a priori, and this has to be taken into account properly when designing the distributed control applications.

The FTDMA mechanism of Byteflight is very suited to support SRT exchanges. In fact, they can be assigned the identifiers just above those allocated to HRTs (i.e., they can be carried out by means of asynchronous data exchanges). As in CAN, no two messages are allowed to share the same priority (and thus, the same identifier). Whenever the time available in the current cycle is not sufficient for sending a particular SRT message, it is delayed to the next cycle and, in the case the overall offered load exceeds (for a short time) the available bandwidth, it might even experience a delay of several cycles.

It is worth noting that this approach is the most suitable when there is a (potentially) high number of SRT messages defined in the system, that are required to be sent seldom and sporadically. As long as the generation pattern of SRTs is spread evenly over time, low delays will be experienced on the average. On the contrary, when a temporary overload condition occurs (many SRT messages that are generated almost at the same time), they are scheduled for transmission according to their relative priority order so that the lower the priority the higher the delay.

3.3. Non-real-time exchanges

The last kind of communications needed in complex systems is represented by *non-real-time* (NRT) messages. Because in today's vehicles there is an ever increasing need of high-level data exchanges for supporting applications related to information and entertainment (usually referred to as infotainment), new networks have to support this kind of communications in a proper way as well. Modern cars often include many devices (a dozen or more) that need to interact with the driver and the passengers by means of visual and audible interfaces. For managing infotainment facilities,

protocols such as *media oriented systems transport* (MOST) [10] were developed (incidentally, MOST is currently the “de facto” standard for such a kind of communications).

NRT data exchanges require high throughput and fair bandwidth share-out. While a bit rate in the order of 10Mbit/s seems to be adequate for the current needs of many infotainment applications (including web browsing, multimedia content delivery or voice transmission), Byteflight fails in ensuring the required level of fairness. Because of the prioritised approach used by the FTDMA technique, in fact, it is not possible to ensure the same quality of service to two or more connections at the same time.

This is clearly a problem in a number of cases. In fact, the communication requirements of infotainment applications inside vehicles closely resemble those found in office automation or home entertainment systems. For many high-level functionalities, such as internet access or streaming multimedia, if the traffic generated by the applications exceeds the available network bandwidth (because of a temporary overload condition, for instance), it is preferable to reduce the amount of bandwidth available to each connection rather than to completely rule out some of them.

3.4. Comparison with fieldbus networks

From the above considerations, it can be seen that the kinds of communications that are needed by modern automotive systems are roughly the same as those involved in controlling automated manufacturing systems in industrial plants. Nowadays, every field network has to support real-time process data exchanges for controlling the physical system as well as non-real-time communications concerning device configuration, diagnostics and management. Either a time-triggered or an event-driven approach is required for exchanging process data and, in some cases, both of them are needed at the same time.

For example, numerical control applications usually benefit from a synchronous TDMA transmission scheme, in that it ensures the lowest jitters and hence the best precision in controlling the mechanical parts of the

system (the SERCOS protocol is currently considered the solution that best satisfies these requirements).

On the other hand, supporting an asynchronous real-time transmission scheme (for notifying alarms or emergency conditions quickly) and efficient high-level communications (for carrying out device parameterisation easily) besides synchronous data exchanges, is a valuable improvement over the traditional solutions.

This fact will affect the future of field networks. In particular, new-generation automotive protocols such as Byteflight and FlexRay can be used quietly at the field and cell levels in many advanced high-performance automated manufacturing systems, in the same way as CAN was largely adopted in the past decade to interconnect field devices to PLCs or industrial PCs. In this case, the higher level of performance they achieve makes it possible to either shorten the cycle times (well below 1ms) or increase the number of process data that can be exchanged in the system.

On the one hand, synergic effects between factory automation and automotive scenarios can decrease noticeably the costs of hardware components while, on the other hand, they push further technological research, which implies the availability of both application-level standard protocols and sophisticated development tools.

4. Performance analysis of Byteflight

To better understand the performance that can be expected in a Byteflight system, real network parameters should be taken into account. In the following, we will assume that the relevant network parameters are selected according to the values specified in the rightmost column of Tab. 1.

For the sake of simplicity, we assumed that all the nodes are located at the maximum distance allowed from the star coupler. This implies that T_{wx0}^{tx} shouldn't last less than the *minimum idle time* (11 bits), that is the time for which the bus must remain idle between adjacent frames (actually, T_{wx0}^{tx} has been selected as the maximum between $11 T_{bit}$ and $T_{wx\Delta}$).

Tab. 1. Typical network parameters.

Symbol	Meaning	Requirements	Value
T_{bit}	Bit time (bit rate 10Mbit/s)		100 ns
T_{cyc}	Cycle time		$2500 T_{bit}$
T_{w0}	Sync pulse waiting time (min. idle time + max. sync pulse normal)	$< 60 T_{bit}$	$41 T_{bit}$
T_{wx}	Waiting time between frames	$\geq 11 T_{bit}$	
T_{wx0}^{tx}	Fixed component of T_{wx} (node was transmitter of the last frame)	$< 19 T_{bit}$	$11 T_{bit}$
$T_{wx\Delta}$	Multipl. component of T_{wx} (end-to-end delay + recognition of bus activity + tolerance)	< 1000 ns	500 ns

Because of the rules adopted in Byteflight for accessing the shared medium, each message (including the sync pulse) is followed by an idle period not shorter than $T_{wx0}^{tx} + T_{wx\Delta}$, as depicted in Fig. 2. Moreover, each identifier corresponding to a message that is not sent generates a minislot whose duration is equal to $T_{wx\Delta}$.

The time T_{ID} , elapsing from the beginning of the cycle (rising edge of the sync pulse) to the end of the transmission of the message characterised by the identifier ID , can be evaluated as

$$\begin{aligned} T_{ID} &= (ID - n_{tx}) \cdot T_{wx\Delta} + \sum_{k=1}^{n_{tx}} (T_m^k + T_{wx0}^{tx} + T_{wx\Delta}) \\ &= ID \cdot T_{wx\Delta} + \sum_{k=1}^{n_{tx}} (T_m^k + T_{wx0}^{tx}) \end{aligned} \quad (3)$$

where n_{tx} is the number of messages that are effectively exchanged in the transmission cycle (before message ID) and T_m^k is the time taken to send the k -th message over the bus (it is worth remarking that here k is not the message identifier). Equation (3) can be rewritten as

$$T_{ID} = ID \cdot T_{wx\Delta} + n_{tx} \cdot (\bar{T}_m + T_{wx0}^{tx}) \quad (4)$$

where \bar{T}_m is the average transmission time of the exchanged messages. In turn, T_{ID} must satisfy the following constraint

$$T_{ID} \leq T_{cyc} - T_{w0} \quad (5)$$

Let ID_{max} be the highest identifier that can be assigned to a message, so that it can be sent in the current cycle (or, better, it is not ruled out for sure). According to (4) and (5), ID_{max} and n_{tx} are related by the following equation

$$ID_{max} = \left\lfloor \frac{T_{cyc} - T_{w0} - n_{tx} \cdot (\bar{T}_m + T_{wx0}^{tx})}{T_{wx\Delta}} \right\rfloor \quad (6)$$

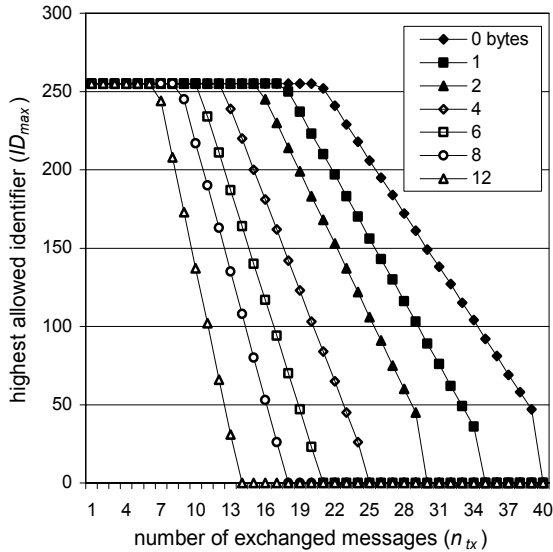


Fig. 4. Highest allowed message identifier vs. number and size of exchanged messages.

It is clear from (6) that the adoption of identifier values above ID_{max} is meaningless, in that they cannot be sent on the network.

In Fig. 4 ID_{max} is shown versus n_{tx} for several different (average) message sizes. As we can see ID_{max} decreases quickly as n_{tx} increases. This means that, even in average load conditions, messages associated to high identifiers (that is, close to 255) can hardly be sent. In other words, Byteflight trades efficiency for flexibility. If the applications rely on many asynchronous messages, which are sent sporadically by their producers, the bandwidth usage has to be kept low.

From the above considerations it follows that it is worth allocating identifiers in increasing order for asynchronous messages too, starting with the first available value and avoiding gaps (if this is possible).

4.1. Synchronous and asynchronous transmission schemes

The maximum number n_{max}^{FTDMA} of messages that can be included in each transmission cycle with a non-null transmission probability can be evaluated by means of the following equation

$$n_{max}^{FTDMA} = \left\lfloor \frac{T_{cyc} - T_{w0} - ID_{high} \cdot T_{wx\Delta}}{\bar{T}_m + T_{wx0}^{tx}} \right\rfloor \quad (7)$$

where ID_{high} is the highest identifier that is assigned to a message. Generally speaking, n_{max}^{FTDMA} includes all the synchronous messages and a number of pending asynchronous messages (that are served in increasing priority order). From a practical point of view, the number of synchronous messages included in every cycle should not exceed $n_{max}^{FTDMA} - 1$ so that asynchronous messages are not ruled out. For example, when 8-byte-sized messages are taken into account, no more than seven messages can be included in the transmission cycle so that the message with the highest identifier (255) can be sent.

In the same conditions, when the minislotting technique isn't fully exploited (that is, Byteflight is used according to the QTDMA plain scheduling, where the messages to be sent are assigned increasing identifiers in the range from 1 to n_{tx}), up to 17 messages can fit in the same cycle. This can be evaluated by means of the following equation, which provides the maximum number n_{max}^{QTDMA} of consecutive (synchronous) messages allowed in such operating conditions

$$n_{max}^{QTDMA} = \left\lfloor \frac{T_{cyc} - T_{w0}}{\bar{T}_m + T_{wx0}^{tx} + T_{wx\Delta}} \right\rfloor \quad (8)$$

The ratio between n_{max}^{FTDMA} and n_{max}^{QTDMA} can be used as an index to measure the loss in efficiency of the FTDMA mechanism with respect to a quasi-TDMA approach in order to ensure flexibility, that is for

enabling the transmissions of up to ID_{high} different messages (in general, ID_{high} is higher than both n_{max}^{FTDMA} and n_{max}^{QTDMA}).

The values of n_{max}^{QTDMA} and n_{max}^{FTDMA} versus the (average) message size are plotted in Fig. 5 for different values of ID_{high} . As expected, the diagrams show that when more identifiers are needed the resulting efficiency is lower. When the maximum flexibility is required (i.e., 255 identifiers), the achieved data rate is reduced to about a half with respect to a pure synchronous scheme.

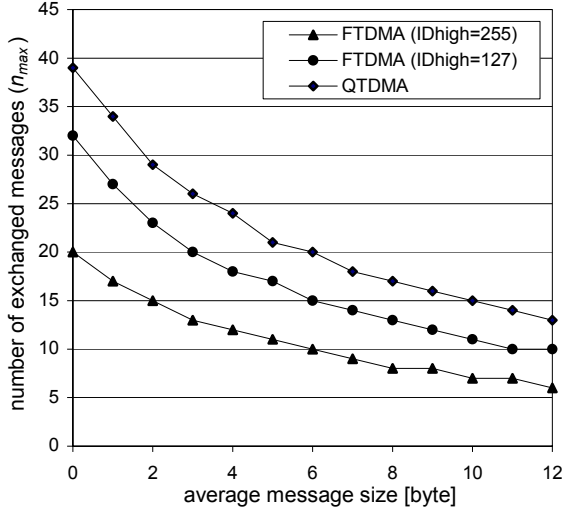


Fig. 5: Number of exchanged messages vs. average size and available identifiers.

It is worth noting that while this limit doesn't seem so severe for Byteflight, it might be a serious drawback for FlexRay. In fact, the FlexRay protocol is very similar to Byteflight to several extents, but foresees up to 4095 different message identifiers for increasing flexibility. This means that in FlexRay the transmission of messages with high identifier values might result in a noticeable loss of efficiency, because of the non-null size of minislots. At the limit, it could be impossible to select most of the highest-valued identifiers, even when a single asynchronous message should be exchanged over the network.

This can be a problem if FlexRay is used for emulating a CAN network (for example, one might conceive a "CANopen over FlexRay" solution to be used in factory automation systems). The only way to enable the transmissions of all available identifiers is by enlarging the cycle time. In this case, however, the responsiveness is reduced consequently.

4.2. Design guideline

Let n_{syn} be the number of synchronous messages that have to be sent in every transmission cycle, whereas n_{asy} represents the maximum number of asynchronous messages required in the system. If identifiers are

allocated according to the rules given in section 2.4, ID_{high} is equal to $n_{syn} + n_{asy}$. In order to exchange all the synchronous messages and at least one asynchronous message in every cycle, the condition $n_{tx} \geq n_{syn} + 1$ must hold. From (3) and (5) we obtain

$$T_{cyc} \geq (n_{syn} + n_{asy}) \cdot T_{wx\Delta} + \sum_{j=1}^{n_{syn}} T_m^{(j)} + \max_{1 \leq i \leq n_{asy}} (T_m^{(n_{syn}+i)}) + (n_{syn} + 1) \cdot T_{wx0}^{tx} + T_{w0} + T_{spare} \quad (9)$$

where $T_m^{(ID)}$ is the time taken to transmit the message with identifier ID serially over the bus.

The above condition must be satisfied to ensure that the requirements on the transmission of both the synchronous and asynchronous messages are met, thus it can be used as a design guideline. A spare time T_{spare} longer than 0 can be foreseen to provide additional room for asynchronous messages, in order to make their transmission times shorter.

4.3. Main limitations of Byteflight

In CAN there is a clear trade-off between bit rate (i.e., network performance) and bus length (i.e., network extension). In protocols based on the FTDMA approach, instead, the trade-off involves efficiency (i.e., performance) and flexibility (i.e., number of available identifiers).

It is worth noting that identifiers have never been a problem in CAN. The overhead introduced by the bitwise arbitration, in fact, depends on the number of available identifiers according to a logarithmic law. The 11-bit standard format is sufficient for 2048 different messages, while more than half a billion messages are permitted with the 29-bit extended format (this is much more most real applications will ever need).

At this point one might ask whether or not the network extension affects the Byteflight performance too. The answer is that larger networks imply higher propagation delays which, in turn, mean higher values for $T_{wx\Delta}$ and hence lower communication efficiency. This fact has one important consequence: if the transmission speed R of Byteflight is increased but the maximum network extension is left unchanged, we shouldn't expect that the cycle time can be shrunk in the same way.

In Fig. 6 the maximum number of messages that can be effectively sent within the transmission cycle is shown versus the average message size for different transmission speeds R ranging from 10Mbit/s to 35Mbit/s. The transmission cycle was selected equal to 2500 bit times (that is 250μs at the nominal bit rate of 10Mbit/s), whereas ID_{high} was set to 127. Fig. 6 shows that if we wish to grant a given level of flexibility when the bit rate increases the network traffic has to be reduced.

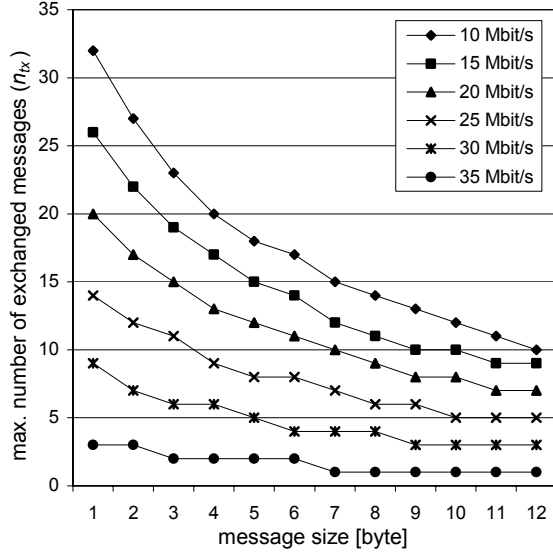


Fig. 6: Maximum number of exchanged messages vs. message size and transmission speed.

Let τ_{cyc} , $\bar{\tau}_m$, τ_{w0} and τ_{wx0}^{tx} be the values of the corresponding timing parameters (T_{cyc} , \bar{T}_m , T_{w0} and T_{wx0}^{tx} , respectively) normalised to the bit time. From (4) and (5) the normalised cycle time has to satisfy the condition

$$\tau_{cyc} \geq n_{tx} \cdot (\bar{\tau}_m + \tau_{wx0}^{tx}) + \tau_{w0} + R \cdot (ID_{high} \cdot T_{wx\Delta}) \quad (10)$$

If we consider a conventional Byteflight system and an average message size of 4 bytes, no more than 20 messages can be exchanged in each transmission cycle so that up to 100 different messages can be defined. This means that n_{syn} is at most 19, in which case n_{asy} can be up to 81.

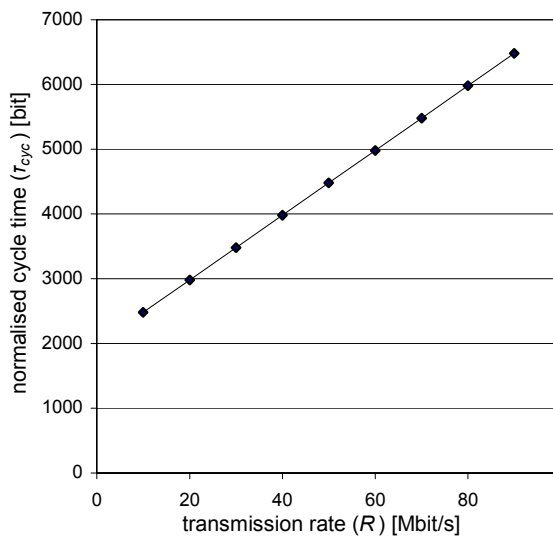


Fig. 7: Normalised minimum cycle time vs. bit rate.

By putting these values in (10), it is possible to evaluate how much the cycle time can be safely shortened (by increasing the bit rate R) without the overall network behaviour changes in any respect. Results are reported in Fig. 7, which shows that the minimum allowed (normalised) cycle time τ_{cyc} grows as the transmission speed increases.

4.4. Network extension

The Byteflight specifications don't define explicitly the maximum allowed network extension. However, it can be easily evaluated by means of the following reasoning. Transceiver ICs introduce propagation delays that are usually in the range from 60ns to 100ns, while the propagation speed is about 0.2m/ns. According to the data sheets, the time taken by the controller to recognise a bus activity is typically less than 100ns. As the typical worst-case propagation delays foreseen by the specification are about 400ns, the maximum length of each link is in the order of few tens of meters.

This implies that the overall network extension is roughly the same as a full speed (1Mbit/s) CAN network, whereas the data rate is about one order of magnitude higher. Increasing the network extension requires $T_{wx\Delta}$ to be increased consequently which, according to the previous section, implies lower communication efficiency.

5. Conclusions

Byteflight is a new-generation high-performance network conceived for automotive systems that demand high data rates and a deterministic behaviour, such as, for example, passive safety systems.

In this paper the main features of Byteflight have been analysed and its performance evaluated. From several points of view Byteflight can be considered as a high-performance version of CAN. However, a more detailed analysis shows that the main issue with Byteflight is the trade-off between performance and flexibility: in order to achieve a higher flexibility for asynchronous data exchanges, performance has to be somehow sacrificed.

As this paper pointed out, requirements on data exchanges in modern automotive systems are quite similar to those which are found at the field level of complex automated manufacturing systems. Hence, these results can be applied also when Byteflight is used as a fieldbus. It is worth noting that most of the analysis carried out here can be applied with very few changes also to FlexRay, a very promising solution which is likely becoming one of the main contenders as a possible successor of CAN for tomorrow's automotive systems.

References

- [1] International Organization for Standardization, Road vehicles -- Controller area network (CAN) -- Part 1: Data link layer and physical signalling, ISO IS 11898-1, 2003.
- [2] G. Cena and A. Valenzano, "A Multistage Hierarchical Distributed Arbitration Technique for Priority-Based Real-Time Communication Systems", *IEEE Transactions on Industrial Electronics*, Vol. 49, No. 6, December 2002, pp. 1227-1239.
- [3] International Organization for Standardization, Road vehicles -- Controller area network (CAN) -- Part 4: Time-triggered communication, ISO/DIS 11898-4, 2002.
- [4] TTTech, Time-Triggered Protocol TTP/C High-Level Specification Document, Protocol Version 1.1, Specification edition 1.4.3, November 19, 2003.
- [5] H. Kopetz and G. Grunsteidl, "TTP - A protocol for fault-tolerant real-time systems", *IEEE Computer*, Vol. 27, No. 1, January 1994, pp. 14-23.
- [6] M. Peller, J. Berwanger and R. Griebbach, Byteflight specification (draft), Version 0.5, BMW AG, October 29, 1999.
- [7] BMW AG, Daimler Chrysler AG, Robert Bosch GmbH, General Motors/Opel AG, FlexRay – Requirements Specification, Version 2.0.2, April 9, 2002.
- [8] K. Tindell, A. Burn, and J. Wellings, "Calculating Controller Area Network (CAN) message response times", *Control Engineering Practice*, Vol. 3, No. 8, pp. 1163–1169, August 1995.
- [9] M.L. Homer, "Handling event driven messaging in distributed flight critical systems", *proceedings of the 21st Digital Avionics Systems Conference*, Vol. 2, pp. 13.C.4.1-13.C.4.6, 2002.
- [10] MOST Cooperation, Media Oriented System Transport, Multimedia and Control Networking Technology, MOST Specification, Rev. 2.2, November 2002.