SIMPLE FIELD BUS PROTOCOL

COPYRIGHT ©2003 Claudio H. Ghiotto and Paolo Marchetto
SOFT&MEDIA - HEXEL Electronic lab.

Version: 2.0.0 – November 2004

Reference: SFBP v.1 by Claudio H. Ghiotto and Paolo Marchetto.

SFBP - rev. 2.0.0, Nov. 2004 Review: Jul 2016 Edited for typos and rewording to improve clarity: May 2019

License

This protocol is open and it is released under the terms and condition of the General Public License (see http://www.gnu.org foundation for more info); for more details on copyright statements, please read the copyright section below.

Introduction

Reference

This paper describes the version 2 of the protocol which is a variation from SFBP version 1, written on September 2003. This revision implements a better checksum algorithm, and additional details and corrections over the previous paper.

This revision was necessary to correct the too simple checksum adopted in the earlier version of this protocol. The new algorithm is an actual CRC, though implemented with 8 bit math which features good performances and little footprint either in memory and over the network packets. Although designed to be as simple as possible, the new algorithm features good rejection to the typical noise on the intended transmission media. However, it is stressed that being an 8 bit CRC with a simple algorithm it remains less strong compared to the classical 16 bit CRC. So it is recommended to implement both the CRC check and packet type check in the Application layer to reduce possible errors.

Description

The Simple Field Bus Protocol was designed for communications among devices operating on an event-driven paradigm and shared computation over a single field bus or related field buses, in a multiple access configuration. This protocol was also designed to be as simple as possible allowing compact implementations that fit well in embedded systems.
Key points are:

- bi-directionality
- multi-master
- asynchronous access, which best suit event-driven systems
- application and physical layer agnostic
- small fixed packets, that best suit commands or small amount of data (such as updates of values).

The protocol may also transport streams of data, even though it is not optimized for this purpose.

Here are summarized the driving factors that led to the development of the protocol and may provide a guideline to understand if this protocol may suit your needs:

- reach a reasonable amount of devices
- be able to address all devices at a time (broadcast operations)
- both synchronous (connected) and asynchronous unconnected communication (datagrams)
- optimized for small amount of data, however providing also a way for larger data transfer (carrying fragmented stream of data using synchronous packets)
- multiple communications (master to master configuration), multiple access over a single carrier
- fault tolerant in the case of collisions and an optional method to reduce collisions
- provide a marker for easy parse of sequences of packets stored into a buffer
- independent from the physical layer
- fits the range of 8 bit of data per frame (to be able to run over a RS232 for point to point links, for instance)

Furthermore, some assumptions were made: the protocol relies on a field bus line capable of detecting line faults, namely collisions. Expansions can be done by adding repeaters, however this topic goes beyond the purposes of this document.

Although the protocol was designed having in mind the IEEE RS 485 standard as physical layer, the SFPB packets can be carried over others physical layers, including radio links.

All changes implemented into the versions two still meet the original purposes.

Definitions

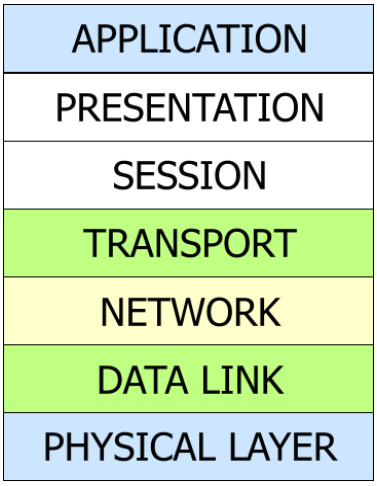
DU	Data Unit – Payload Data Unit
CSMA/CD or CS / CD	Carrier Sense Multiple Access / Carrier Detection
MAC	Media Access Control
ACK	ACKnowledge
SM	Start Marker
DA	Destination address
SA	Sender address
PI	Packet information
CS or CRC	Check sum or Cyclic Redundant Checksum

Characteristics

- Up to 127 addressable device (including stations and gateways)
- 1 broadcast address
- Unconnected datagram and connected packets for either fast messages or safe data transfer
- Fragment bit field for streaming data bigger than a DU (DU, in this document stands for the Payload Data Unit)
- Compact packet headers made up of three bytes
- DU (data unit) 6 bytes in length
- Packets formed with up to 11 bytes
- Seven possible type of packets: Data, Control, Time, Echo and System (other codes are reserved)
- CSMA/CD like collision detection
- Simple packet acknowledgment system with a single datagram packet formed with only 4 bytes in version 1, and 5 bytes in version 2
- Optional: Specific for RS485, RS422 and RS232 physical layers, frames formed with up to 8 bit data, 1 stop bit, 1 parity bit, and even parity checking.
  Remark: this is a specific typical application where SFBP is used over RS485, RS422 and RS232, but it is not a mandatory specification of SFBP.
- Optional: Failure detection at frame level (i.e., parity check bit), mandatory at packet level.

The ISO-OSI Model and SFBP

The ISO-OSI model



APPLICATION: Provides network services at the application level.

PRESENTATION: External data representation (e.g., byte ordering (coding), compression/decompression, protection...)

SESSION: Ensures data exchange between applications, e.g. Synchronization.

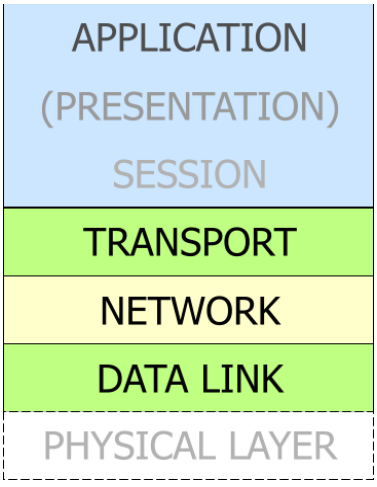
TRANSPORT: The transport layer is between the layers oriented to data processing (the upper ones) and the layers oriented to communication (the lower ones). Long messages are divided into packets on the sender side and they are collected on the receiver side.

NETWORK: Network layer is responsible of routing packets, higher layers are independent of routing.

DATA LINK: Data link layer controls media access (MAC: Media Access Control), data transfer, error detection and packet format.

PHYSICAL: The physical layer specifies the physical characteristics of the link therefore the type of the medium; such as voltage coding and modulation, for electric media, or light spectrum and intensity, for optical media.

The SFBP model against the ISO-OSI model



APPLICATION: SFBP partially implements the application layer with the packet types “Control”, “Data” and “Time”. It does not define the Presentation nor the Session layers that are implicitly left to the Application.

The Type of packet bits of the Packet Information section are related to the Application layer level, and in addition the NEXT bit of the Packet Information section may play a role also at the Session layer level. Byte ordering is ~~left to the application, though~~ Big Endian order is the preferred (where the most significative byte is the last one).

TRANSPORT: The ACK packets allow connected packets and mechanism for retransmission on fault of connected packets. The NEXT bit provides for a basic mechanism to deal with large messages. However further finer mechanisms for the transmission of large messages is left at the application level.

NETWORK: ACK packets and System packets works at Network layer level, they can be used also for routing purposes.

DATA LINK: This layer is covered by packets format and the CSMA/CD similar mechanism and timing rules.

PHYSICAL: SFBP does not specify any physical layer. Even though it was designed on top of a IEEE RS 485, it can run over any other physical layer such as IEEE RS 232, optic, radio and so forth.

Notes: Byte ordering was left to the application in version 1 of the protocol, in version 2 it is mandatory the Big Endian order.

SFBP
Simple Field Bus Protocol – Detailed information

Definitions

packet with-length	It is the length of a standard packet expressed in the unit of time defined by the size of data bits, times the clock time required for each bit (full bit in the case the physical layer uses a pulse width modulation to distinguish between 1 and 0).
frame width	Is the size of a frame of bits transmitted over the bus. It carries a single byte of information.
[t]	This symbol indicates a time wait of almost three frame width (see above).

Packets description

Standard Simple Field Bus Protocol typical packet

SFBP packets have a fixed size of 11bytes.

SM DA SA PI DU DU DU DU DU DU CS [t]

SM	start marker	1 byte	value: fixed, 254
DA	destination address	1 byte	value: 0...127 (0x00...0x7f)
SA	sender address	1 byte	value: 0...127 (0x00...0x7f)
PI	packet information	1 byte:	

```
7  6  5  4  3  2  1  0
L  L  L  A  N  T  T  T
```

L DU valid length, this field is 3 bits long, defines how many bytes are valid in the data unit.

A ACK bit field *

N NEXT bit field *

T Type of packet, this field is 3 bits long, defines the type of packet:

0 Echo
1 Control packet
2 Data packet
3 Time
4 reserved (was experimental *priority* in version 1, which has been removed)
5 reserved
6 System packet

System packets (with L set to any non-zero):

```
7    6    5
L1 - L2 - L3
```

T = 6

L1, L2, L3:

0	forbidden (must be never used)
1	Reset, device must reset communication subsystem
2	Stop device (this is meaningful for devices running processes)
3	peer is ready for communication
4	synchronization token
5	reserved
6	reserved
7	illegal value (see remarks below)

Remarks:
System packets never carries the DU, so they behave as ACK packets which are limited to 5 bytes (SM DA SA PI CS)

this field must never be combined with all the L1, L2, L3 bits set to prevent to be confused with a STARTMARKER.

7 reserved

* the combination of ACK and NEXT bit fields set to one, means an Unconnected datagram packet, only Connected packets may use NEXT.
ECHO packets can be Connected only. It means that both ACK and NEXT bits must be cleared, and instead of an ACK packet the remote peer sends back in return a DATA packet with the same data payload.
NEXT is set on even packets that belong to a stream of data packets. So a stream of data is a sequence of packets with the bit NEXT alternating set to 0 and 1.
ACK is set when a packet is sent as acknowledgment (see ACK Packets below) in response for a Connected packet.

Summary for all the combinations for ACK and NEXT fields

NEXT	*ACK*	*meaning*
0	0	standard connected packet
0	1	acknowledge packet
1	0	connected packet part of a stream of data (expected a further packet)
1	1	datagram non-connected packet

DU	data unit	6 bytes	invalid bytes should be zero padded with zeroes to the DU length (6 bytes).
CS	checksum	1 byte	value: cyclic sum of all bytes but SM and CS itself. (see addendum: CRC algorithm below)

[t] Physical layer : Delay time of 3 frames width, rounded in excess
This is not mandatory. It is suggested for physical layers such as the IEEE RS 485 or via radio links.
It helps reducing the likelihood of collisions and data corruption over the network.

ECHO packet

An echo request is performed by sending a packet of type ECHO: both bits ACK and NEXT must be cleared, and 6 bytes of data must follow (L must be 6).
On an ECHO packet request a user agent should respond with a DATA datagram (not connected) packet carrying the same data included into the original ECHO request packet.

Example:

A ==> ECHO ==> B ==> DATA ==> A

	SM	***DA***	***SA***	***Packet info***	***data0***	***data1***	***data2***	***data3***	***data4***	***data5***	***crc***
sender (1)	254	2	1	T=ECHO L=6 ACK=0; NEXT=0	1	2	3	4	5	6	CRC
receiver (2)	254	1	2	T=DATA L=6 ACK=1; NEXT=1	1	2	3	4	5	6	CRC

ACK packet (acknowledgement packet)

ACK packet have a fixed size of 5 bytes.

Remark*: This is in contrast with the v.1 of the protocol where only 4 bytes were used.*

SM DA SA PI CS [t]

SM, DA, SA, PI and CS are the same as in a normal packet, but PI must have the ACK bit field set.

PI of a ACK packet:

L	zero
L	zero
L	zero
A	ACK bit = 1
N	NEXT bit = 0
T	Type = 0

(L = DU payload size)

System packet

System packets have a fixed size of 5 bytes, they are sent for urgent system reasons and they must be processed even while waiting for NEXT packets in a stream of data.

***Remark**: This is in contrast with the v.1 of the protocol where only 4 bytes were used.*

SM DA SA PI CS [t]

SM, DA, SA, PI and CS are the same as in a normal packet, but PI must have both ACK and NEXT bit fields set (as in a datagram packet).
The first higher three L bits field hold the type of system packet (zero is forbidden since it is used for ACK packets, 7 is illegal because conflicts with STARTMARKER), and the T field must contain the value 6.

System packets may be either broadcast or directly sent to a targeted address.

PI of a system packet:

L, L, L	system statement: zero is ***forbidden***, other possible values: 1 reset communication subsystem 2 stop device (meaningful for devices perform processes only) 3 peer is ready for communication 4 synchronization token 5 reserved 6 reserved *7* ***illegal***
A	ACK bit set = 1
N	NEXT bit set = 1
T	Type set = 6

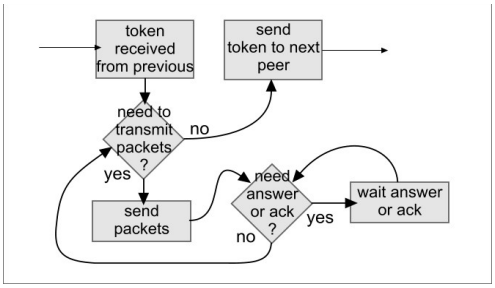
System packets as a mean for synchronization (Optional implementation, still experimental)

Two system packets are provided as a mean for synchronization among peers for accessing the line: one (L=3) is a master free, collaborative, mechanism; the other one (L=4, Synchronization Token) is master-oriented.
Both are not mandatory and each one of them can be implemented as an option.

Collaborative

Value 3 for L bits indicates that the peer is ready to receive data or free the line. This is a multi-peer synchronization mechanism where a token (the system packet with L set to 3) is bounced among peers as soon as they are ready to yield the transmission line, and they will obey to not take the line until the next token is sent to them. This way each unit connected at the transmission line will continuously receive the token, and re-transmit it upon reception to the next peer in the chain of addresses as soon as the unit has no more packets to send over the transmission line. This obviously requires that each unit has contiguous addresses starting from 1 and each unit must know the last address. Alternatively each unit must be chained by setting the “address before” and “address after”.

The scheme on the right summarizes the process.



DA address is from previous peer in the chain, SA address indicates the recipient peer to which the token is yield.

Caution: this mechanism can lead to a deadlock if one unit in the chain does not respond at all. In cases where this condition must be avoided the Synchronization Token is a better solution (see below).

Synchronization Token

Value 4 for L bits indicates that a unit (typically acting as master) has offered to a peer an exclusive access to the serial line. The peer that receives the token can send one packet and expect one answer from a peer. Anyway the transmission must be completed within the duration of 2.5 full-length packets (11 bytes each). In other words the master reserves equal slots of time per each peer for the exclusive access of the serial line. This mechanism is meant for real-time systems.

Remarks

The synchronization mechanism change the multi-master nature of SFBP into a master-driven architecture. At the moment of writing this paper it is unknown whether anybody has implemented the synchronization mechanism, with the exception of the Networked Shared Control gateway routers.

Priority Packet

Remark: Experimental.

*** REMOVED ***

Connected packets

Connected packets have the ACK bit field not set (= 0). In the case the packet is part of a data stream, and a next packet is expected, then the NEXT bit field set (= 1).
Each connected packet is sent to the remote peer, then the sender set up a timeout and waits until the receiver returns an ACK packet (see ACK packets), or a timeout occurs.

Single ACK

Because this protocol rely on the CS-CD information (returned by the lower level of the OSI model, the physical line status, detecting collisions and faults), to keep the protocol simple no further reply is expected after an ACK packet is sent or received.

Progressive Wait Time

The sender should stop immediately if a collision or line fault is detected, and should wait for some time before attempting again. Such a wait time is progressively increased by a random constant which should be set to be different for each device on the network. Each time a transmission attempt fails, the wait time increases.
The number of attempts is finally limited by the RETRYSEND limit (see diagram).

Therefore if the ACKnowledge is not received within the timeout, then packet shall be deemed to be as not delivered and a new attempt should be made afterward, for a number of times up to the RETRYSEND limit is reached.

SENDER	RECEIVER
packet - - - - - - - - - - - - - - - - - ->	packet ok?
ACK received, <- - - - - - - - - - - - -	yes, reply ACK
end of transmission	

Fragmented packets of a data stream.

Because other devices may attempt to send packets to a recipient peer that is already receiving a stream of data, while receiving a packet with NEXT bit set (which means the packet is part of a fragmented stream), the receiver will set up a status so that it will not accept further packets from any other sender than the original one who sent the first packet having the NEXT bit set. This status should last until a packet with the NEXT bit=0 is received or the receiver unit times out.

At the application level a further mechanism can be implemented to prevent interruptions of a data stream sending an application-specific packet to initiate the transmission and terminate the stream. Though, this goes beyond the scope of the protocol.

Unconnected Datagrams

Unconnected datagram packets have both the ACK and NEXT bit fields set (=1). They are sent over the network without expecting any reply from the remote peer. This type of packets are not granted to be delivered but because the transmitting device can get information about the state of the line, those information could tell whether the device should perform a new attempt. The number of maximum attempts is finally limited by the SENDRETRY constant.

Unconnected datagrams are useful for fast data transfers and pulse or events such as I/O or status notifications.
However should be kept in mind that datagram packets are not guaranteed to be delivered.

Broadcast packets

As the name implies broadcast packets are received by all the devices connected to the line.
Broadcast packets can be of type datagram (unconnected) only, and sent to the address zero (DA = 0). No ACK is expected. Though remote peers are free to reply, and this can be defined at Application level.

Broadcast packets are useful for status notification addressed to all devices, for synchronization, for address discovery, routing and so forth.
Because broadcast packets have the same format of a regular packet with exception of the destination address, they may carry information as Control packets or Data packets.
Because broadcast packets carry the sender address, it may be used, for instance, to notify a device added to the line at run-time or as a plug-and-play mechanism. This is a suggestion left to the Application implementation.

Address Discovery Using Connected Packets

Sending a packet to any one of the 127 possible addresses may be used in special occasions for address discovery and auto-detection. Because each device who recognize itself as the destination address could reply with an ACK packet, which in turn contains the device's address, the sender can collect addresses from all the recipients that have replied.

Address ID and MAC Address

Devices could be have an undefined address ID, but marked with a MAC Address (Media Access Control Address), the address ID could then be set by a special broadcast packet carrying the new intended address ID and up to 5 byte MAC Address in the payload. The device that matches the MAC address then could set the given address ID.

Predictive Synchronous Carrier Sense Multiple Access / Collision Detection Mechanism

The SFBP provides a mechanism to optimize the multiple access to the common medium, allowing better performances, but still deems the media access as a non-deterministic model.
This behavior has been chosen because it is well suited to fit an event-driven model, since events are not predictable by definition.

The mechanism is based on the time required by the packet length (thereafter called “Packet width time”) plus a small amount of “silent” time. This time is synchronized when a STARTMARKER frame is received, even when the following frame carries a mismatched address.

This technique is a very efficient way to send random packets over a heavy trafficked network, limiting the collisions. Tests in our research laboratory have proven that a normal CSMA/CD versus the PS-CSMA/CD gives a reduction of collisions of about 30% under heavy traffic conditions while it does not show a significative difference under low traffic conditions.

Because collisions downgrade the performance of a network as the traffic raise, this explain the increased performances mentioned above only at high trafficked rates.
You are welcome to submit more test results sending the report to info@smartcontroller.org.

Please note that this mechanism requires timing that are almost as large as a “Packet width” or at least a “Frame width”, so they can easily implemented even on physical layers that don’t have any timing like RS232.

Timeouts

The “Frame receive” timeout is set to almost 2 frames width, rounded in excess. This guarantee that an un-synchronized packet will timeout if a fake start marker is erroneously detected. For example, while in the mid of receiving DU data.
“Packet width” timeouts after that all the sent frames are timed out, plus the time required to send almost 3 frames (the “silent” time). This is given by 11 **tf** + 3 **tf** where **tf** is a frame timeout; a new packet can't be sent by any device connected to the shared line before a packet width is timed out.

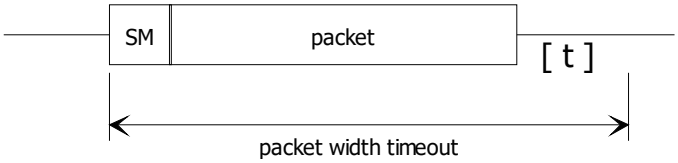
This strict timing is not mandatory and is reserved for tight-timed applications, a minimum implementation just requires to keep track the very basic time of the whole length of a packet and on a PC environment with loose timing this can be simply ignored.

Communications Among Peers

Basically SFBP has been thought as an asynchronous system based on the well known Carrier Sense, Multiple Access mechanism.
While by the mean of System packets a mechanism for the synchronous exclusive access of the line has been included for tight timed system and real-time applications (see System packet, L values 3 and 4, for details), the following describes how collisions can be avoided or managed with the basic asynchronous mechanism.

Predictive Synchronous Carrier Sense.

Carrier Sense is a well known mechanism where a unit before attempting to transmit over the medium, checks if it is free.
The *predictive* synchronous method instead tests for a timeout called Packet Width Timeout, as described above in the Timeouts paragraph.
Before sending a packet over the network a unit checks if the last Packet Width Timeout has expired, and only attempts to transmit the packet after that time is expired.
A Packet Width Timeout is set each time a STARTMARKER frame is detected over the media, by continuously listening the network (see diagram).

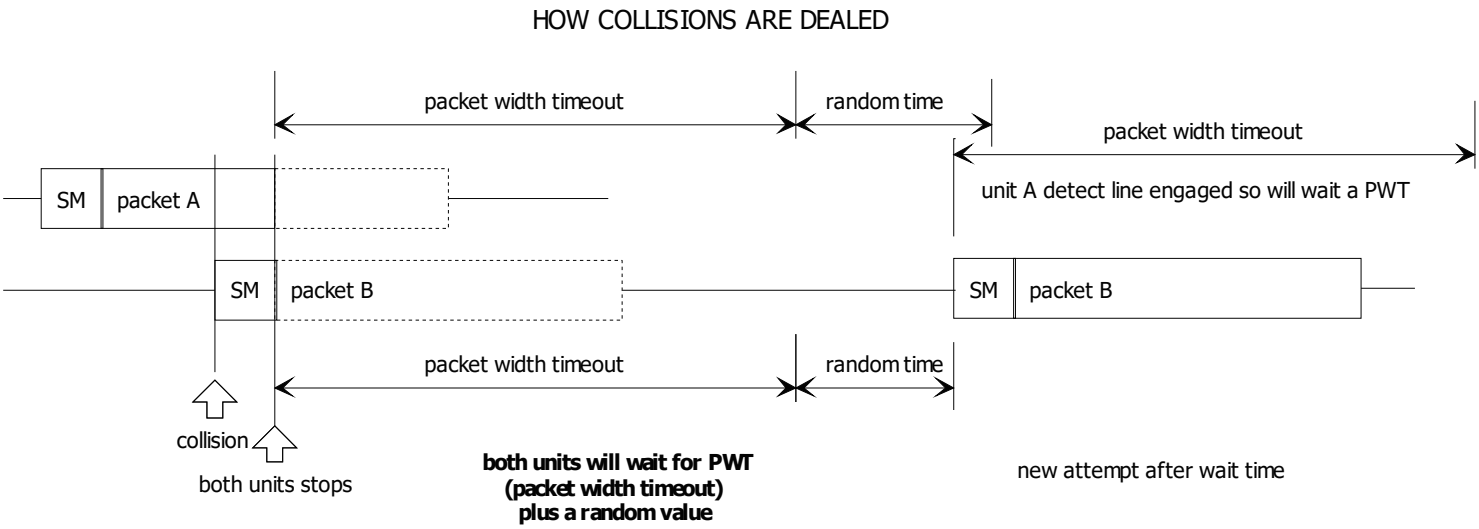


This way a synchronization is started from the last received packet (even if it is not addressed to the unit), reducing the probability of collisions with other units. If anyway two units still attempt to send packets over the network at the same time, a collision will occur causing a distortion on the line signals and causing the message to be mangled.
The sender unit should read the data sent over the network as soon as they are sent. Should the message be mangled, the read data would not match the sent one, allowing the unit to know a collision happened.
This is the Collision Detection mechanism.

When a sender detect a collision it must immediately stop sending frames, and should wait a whole “packet width time” (PWT) plus a random coefficient increased at each failed attempt. This induces a variation that helps avoiding synchronization between partners in attempt to send packets that would collide each time.

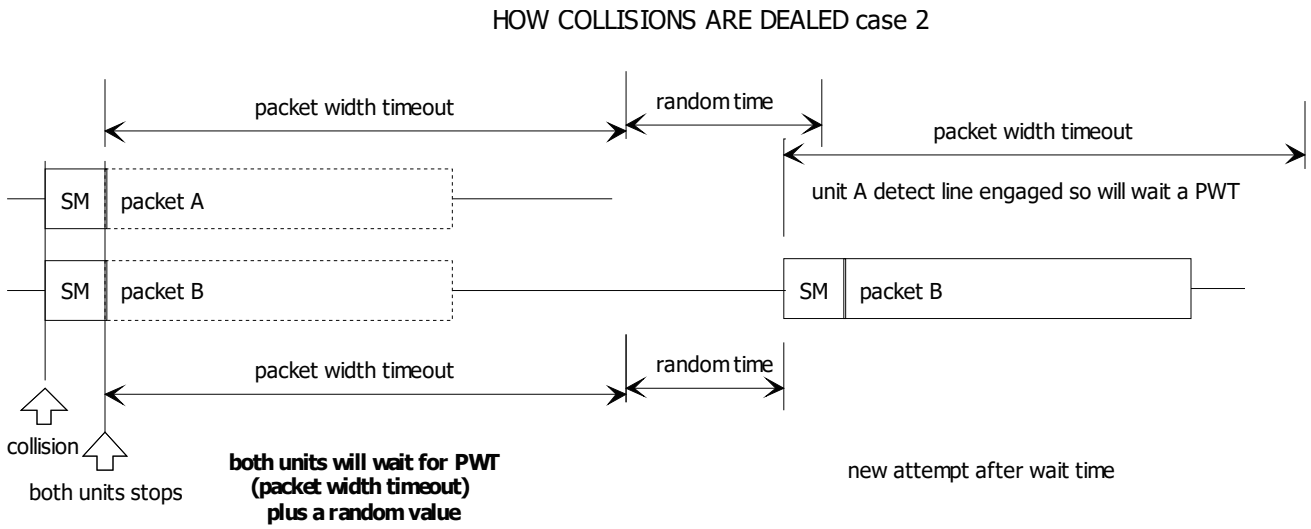
Because a collision may occur randomly, it could happens just in the middle of a packet while the sender is transmitting (for instance, due to a new device “hot-connected”). Because the receiver cannot recover this situation due to the fact that the last received frame was mangled because of the collision, and should produce a CRC error; it is required that a sender who detects a collision immediately stops to send frames and will re-transmit the last packet later.

How much later? Below a diagram shows a possible scenario:



This scenario should be rare because packet B starts too late related to packet A; unit B should detect the beginning of the packet A so that the unit B should have been set a PWT.

This is a more realistic scenario:



In this case both A and B units attempt to send a packet, because at the time they try to send they believes that the line is not yet busy. This will cause a distortion of the signals on the line and both cannot match the actual data on the line against the data sent making the collision detectable.

There are cases where this do not happens. If the two peers are far away each other, the attenuation introduced by the line and the propagation delay would create a local condition where the data appears not to be corrupted, in other words no collision is detected. This is an edge case where other mechanisms should come into play, like the ACK packet and the CRC, possibly checking other integrity factors.

Cyclic Checksum Algorithm

by Claudio Ghiotto

Several algorithms are available to make a cyclic redundant checksum, but almost everyone take significant space in memory, and in short it is cumbersome to implement in small device already packed with the whole stack of software to handle the communication and do other tasks.
So the following algorithm has been conceived. It is lightweight and have a small footprint, does not require to store a lookup table, and because of the small amount of operations involved it is also fast. It basically works processing the data with a polynomial expression so that when noise is applied, which offsets data, it cannot produce a false negative (a fake valid result).
Shift operations actually multiply or divide the number by two, adding then the new data simply makes a polynomial. However a simple shift causes losing information, namely the most or the least significative information, so the number is "rotated" to left by one bit. This preserves the whole information but increases the entropy (diffuses to the whole input information a variation applied to each one of its members), dramatically reducing the probability of false negative CRC results.

Follows a C example which implements the algorithm, the initialization number prevents starting from zero.

principle:

```
unsigned char cs = 23, d;

cs = (( cs << 1 ) | ( cs >> 7 )) + d;
```

full example:

```
unsigned char computeCheckSum(unsigned char *d)
{
	// d		NULL terminated string of data to send

	unsigned char cs = 23, i;
	for(i = 0; d[i]; i++) doCS(&cs, d[i]);
	return cs;
}

void inline doCS(unsigned char *cs, unsigned char d)
{
	*cs = ((*cs << 1) | (*cs >> 7)) + d;
}
```

Its representation is the following:

$$cs = 23$$

$$cs = \left| \left(\; | \, 2cs \, |_{2^8} + \frac{cs}{2^7} \right) + d \right|_{2^8}$$

Where *cs* is the checksum and *d* is the data processed by the algorithm. *cs* is initialized by a prime number (23 has been chosen) before beginning to process all data that will form the final checksum.
In System and ACK packets the members DA, SA, and PI are processed to compute the checksum, which is then appended to the packet.
In Datagram and Connected packets the DA, SA, PI and the 6 DU members are processed to compute the checksum, which is then appended to the packet.