HISTORY OF THE DOCUMENT

Name	Date	Author	Type of change
Version1	28/11/96	J.DE AZEVEDO	WorldFIP update
Version 2	02/10/98	N. CRAVOISY	Update

Although great care has been taken in drawing up this document, **WorldFIP** cannot guarantee the exactness of all the information it contains, and cannot be held responsible for any errors it might contain or any damages that could result from its use.

The presentation, operation and use of the hardware, software and services presented in this document may change at any time, and their description in this document cannot be considered binding.

CONTENTS

1. INTRODUCTION

WorldFIP is a fieldbus network protocol designed to provide links between level zero (sensors/actuators) and level one (PLCs, controllers, etc.) in automation systems.

WorldFIP is a fieldbus whose requirements were set out by users. From the outset the definition of the protocol in three layers was based on:

- economic considerations:

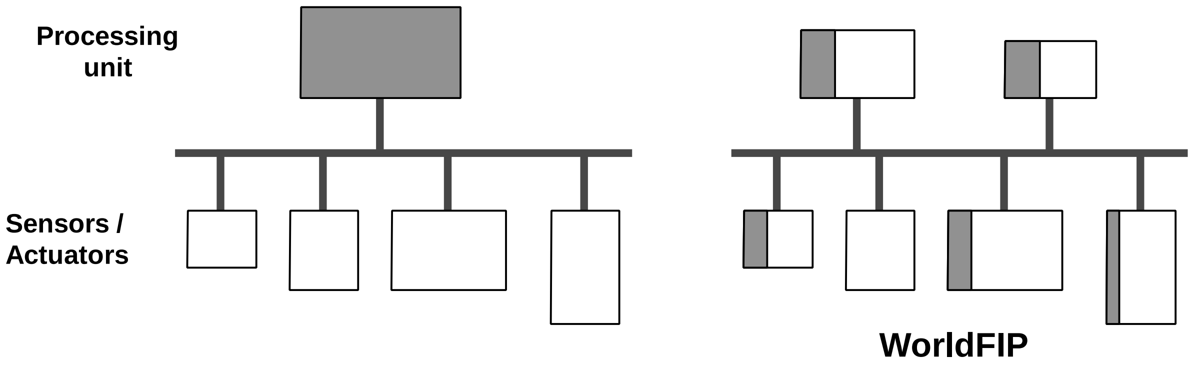
  - reduced cabling costs
  - savings in design, installation and commissionning

- technical considerations:

  - easy maintenance and modification
  - simplification of the traditional point-to-point wiring between sensors and processing units
  - guaranteed response time
  - security
  - accessibility to variables.

WorldFIP can be used with all types of application architectures: centralized, decentralized, master-slave. Distributed applications can be synchronous or asynchronous.

WorldFIP makes it possible to distribute intelligence, control and data:



An algorithm can be located in a single processing unit, or it can be completely distributed. The mechanism for broadcasting data, a basic WorldFIP mechanism, guarantees the spatial and temporal consistency of data for all subscribers that consume a set of variables.

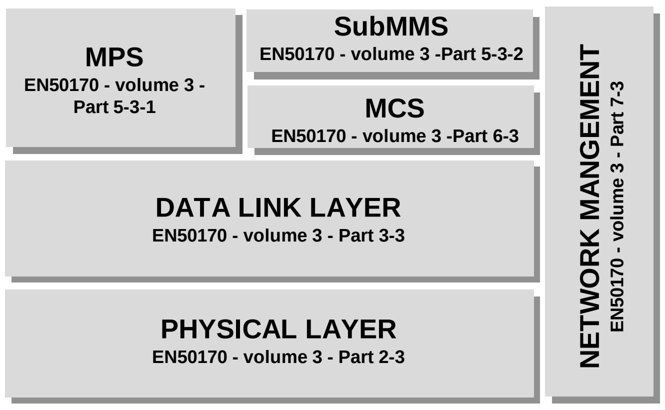
Heterogeneous systems can be designed using WorldFIP, because WorldFIP is an open system that makes it possible to interconnect devices from different constructors.

The WorldFIP protocol is completely specified and is part of European fieldbus standard EN50170.

This standard can be obtained from:
- l'**UTE** (Union Technique de l'Electricité - 33, Av du Générale Leclerc - BP 23 - 92262 Fontenay aux Roses CEDEX - Tél : 01 40 93 62 00 - Fax : 01 40 93 03 96)

as well as from:
- **AFNOR** (Tour Europe - Cedex 7 - 92049 Paris la Défense - tel.: (1) 42 91 55 55)

The **WorldFIP protocol** is made up of the three communications layers shown below:


MPS
EN50170 - volume 3 - Part 5-3-1
SubMMS
EN50170 - volume 3 -Part 5-3-2
MCS
EN50170 - volume 3 -Part 6-3
DATA LINK LAYER
EN50170 - volume 3 - Part 3-3
PHYSICAL LAYER
EN50170 - volume 3 - Part 2-3
NETWORK MANGEMENT
EN50170 - volume 3 - Part 7-3

EN 50170 -volume 3- Part 1-3: General Purpose Field Communication System

EN 50170 -volume 3- Part 2-3: **PHYSICAL LAYER** Specification and Service Definitions
Sub-Part 2-3-1: IEC Twisted Pair
Sub-Part 2-3-2: IEC Twisted Pair Amendment
Sub-Part 2-3-3: IEC Fiber Optic

EN 50170 -volume 3- Part 3-3: **DATA LINK LAYER** Service Definition
Sub-Part 3-3-1: Data Link Layer Definitions
Sub-Part 3-3-2: FCS Definition
Sub-Part 3-3-3: Bridge Definition

EN 50170 -volume 3- Part 5-3: Application Layer Service Definition
Sub-Part 5-3-1: **MPS** Defintion
Sub-Part 5-3-2: **SubMMS** Definition

EN 50170 -volume 3- Part 6-3: Application Layer Protocol Specification(**MCS**)

EN 50170 -volume 3- Part 7-3: **Network Management**

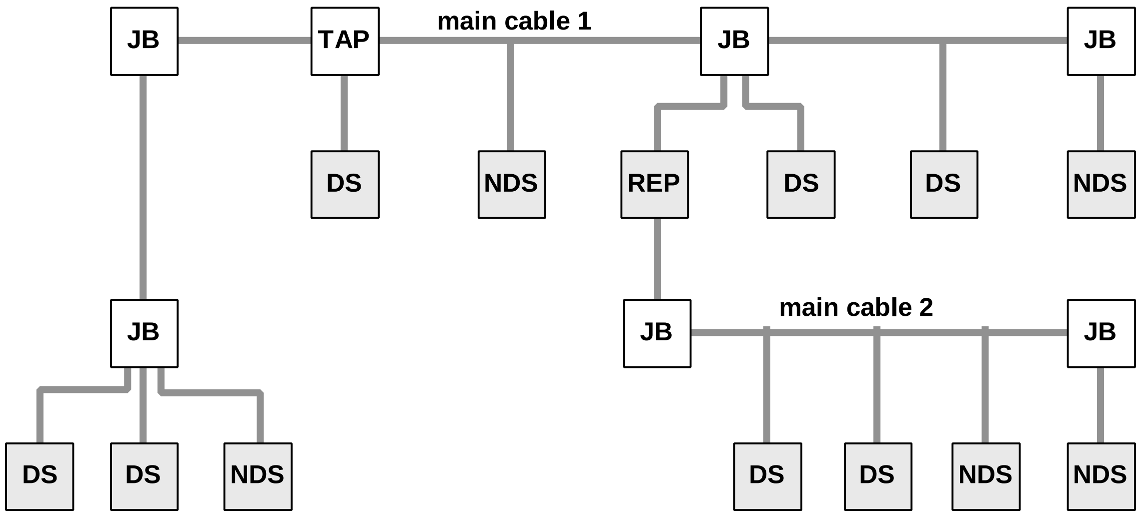
Note that European standard WorldFIP EN50170 -volume 3- replaces French standards FIP C46 601 to C46 607. The essential difference between the new European standard and the French standards is the former's adoption of the international IEC standard for the physical layer (1158-2).

2. PHYSICAL LAYER

WorldFIP's physical layer ensures the transfer of bits of information from one device to all other devices connected to the bus. The transmission medium can be shielded twisted pair wire or optical fiber. (This section deals only with shielded twisted pair wire.)

2.1 Overall configuration

The figure below shows a WorldFIP network with two main cables and a repeater that guarantees the link between them.



The following devices are located on the main cables:

JB	Junction Box A junction box is a passive multiple tap. It provides at least two pull-out accesses for derivations.
TAP	The main purpose of a tap is to provide a connection point on the main cable.
Repeater	A repeater is a special kind of active star. It brings together two main cables to form the fieldbus.
DB	Diffusion Box A diffusion or broadcasting box is a special kind of active star. It brings together several terminal segments on a main cable.
DS	Locally disconnectable subscriber
NDS	Non-locally disconnectable subscriber

2.2 Transmission speeds

Three transmission speeds have been defined for the copper wire physical layer:

- S1: 31.25 kb/s (low speed)
- **S2: 1 Mb/s (high speed)**
- S3: 2.5 Mb/s (high speed)

S2 is the standard speed.

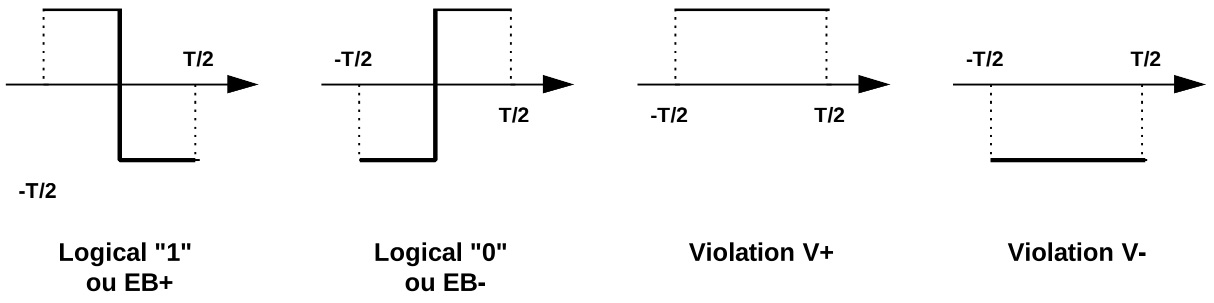
S1 and S3 are used only for special applications.

An additional speed of 5 Mb/s has been defined for an optical fiber physical layer.

2.3 Coding

The physical layer codes the bits transmitted by the data link layer using the Manchester code. This code makes it possible to transmit simultaneously the temporal synchronization of signals and data. Each time interval used for coding a bit is split into two parts of equal duration.

Symbols are represented as follows:



2.4

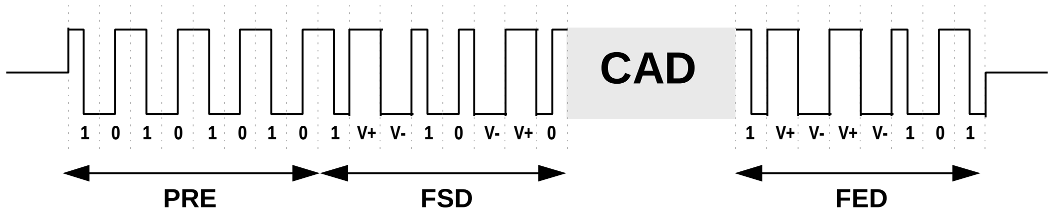
Coding a WorldFIP frame

All WorldFIP frames (frame_question, response, message, etc.) are composed of three parts:

- Frame start sequence (FSS)

- Data and check fields

- Frame end sequence (FES)



The frame start sequence (**FSS**) contains the following fields:

- Preamble (PRE)
  This series of 8 "1" and "0" bits is used by receivers to synchronize themselves with the transmitter's clock.
- Frame Start Delimiter (FSD)
  This series of bits indicates to the data link layer the beginning of useful information (CAD).

The **CAD** (Control and Data) field contains only the logical information ("0" and "1") from the data link layer.

The Frame End Delimiter (**FED**)
This series of bits is used by the data link layer to locate the end of the CAD field.

The physical layer adds 24 symbols to every frame transmitted.

3. DATA LINK LAYER

The data link layer provides two types of transmission service:

- exchanges of identified variables
- message transfers

These exchanges can take place:

- cyclically. When the system is configured the names of objects and their periodicity are set. Exchanges of these variables or messages take place automatically without the user requesting them.

- upon explicit user request. These explicit requests will cause the values of one or more variables, or one or more messages, to circulate on the bus.

3.1 Addressing

The WorldFIP addressing model has two distinct addressing spaces:

- variable addressing

  Each variable in the distributed system is associated with an identifier that uniquely characterizes the variable. Addressing is global.

  Identifiers are coded using 16-bit integers; theoretically 65536 variables can be named. Entities participating in the exchange of a variable do not physically address each other. Rather, they refer to identifiers that they recognize in production or consumption.

  For a given identifier there can be one and only one producer, but several consumers. Variable exchanges are accomplished through broadcasting.

- message addressing

  Exchanges of messages take place in point-to-point or in multipoint on a single segment. Each message transmitted contains the address of the transmitting entity and the address of the destination entity. These addresses are coded using 24 bits, and they indicate the network segment as well as the address of the station in the segment.

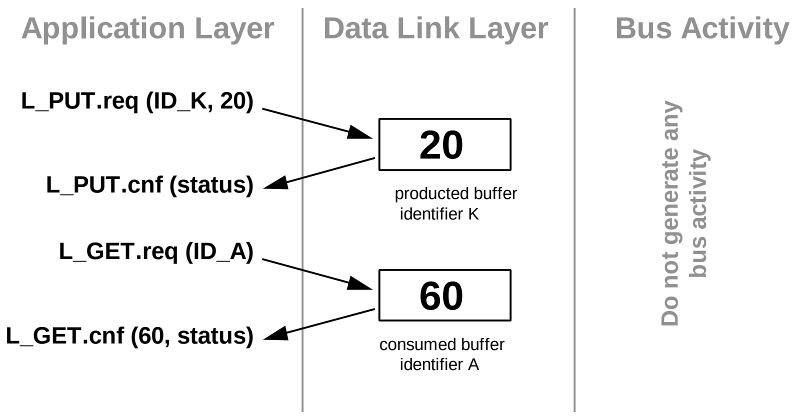
3.2 Application layer-Physical layer interfaces

The data link layer provides services to the application layer and uses the services of the physical layer.

The data link layer is made up of a set of produced and consumed buffers. These buffers contain the latest values updated by the user or by the network. When a new value is introduced in a produced or consumed buffer the previous value is overwritten.

Resources describing these buffers are allocated when the station is initially configured. Access to a buffer is through the intermediary of the produced or consumed identifier.

The drawing below shows the application layer - data link layer interface.



In this example the data link layer contains two buffers:

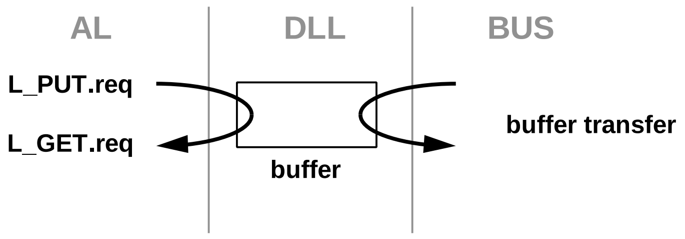
- a produced buffer corresponding to the identifier K.
- a consumed buffer corresponding to the identifier A.

The application layer uses a writing service (L_PUT.req) to place a new value (20) in the produced buffer.

The application layer then uses a reading service (L_GET.req) to remove the value of a consumed variable.

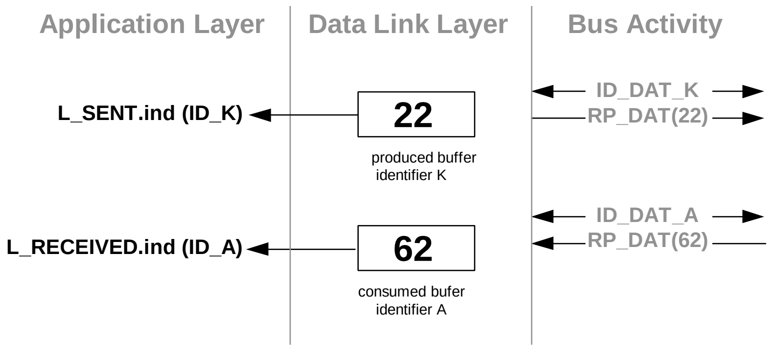
These reading and writing services are local to the station and generate no activity on the bus.

Produced or consumed buffers are also accessible through the bus, which can remove or deposit variable values. This mechanism is known as a buffer transfer.



Since buffers have two accesses the data link layer must resolve problems caused by conflict of access. When a variable is produced and consumed within a single communications entity the data link layer contains both a produced buffer and a consumed buffer.

To achieve a buffer transfer the bus arbitrator transmits the question frame ID_DAT, and states the identifier number.



If the station is declared a producer of the identifier, the data link layer responds with the value of the variable using a response frame RP_DAT. The data link layer then sends an indication of transmission of the value to the application layer (L_SENT.indication).

If the station is declared a consumer of the identifier, the data link layer accepts the value of the next response frame RP_DAT. It then sends an indication of reception of the identifier to the application layer (L_RECEIVED.indication).

Maximum buffer size is 128 bytes. Buffers contain only the values of identified variables; they contain no messages.

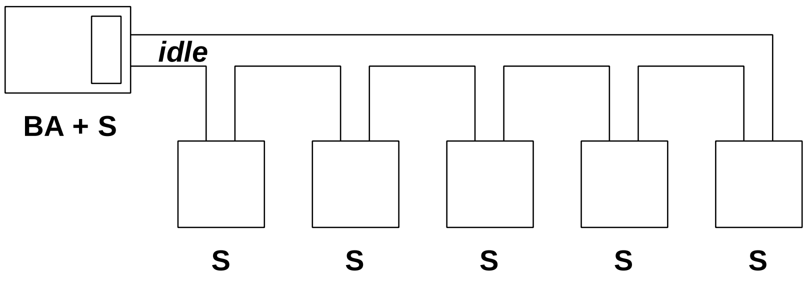
The remainder of this section describes the other communications resources of the data link layer and services provided to the application layer.

3.3 Medium allocation mechanism

A WorldFIP network is made up of stations with two types of functionalities:

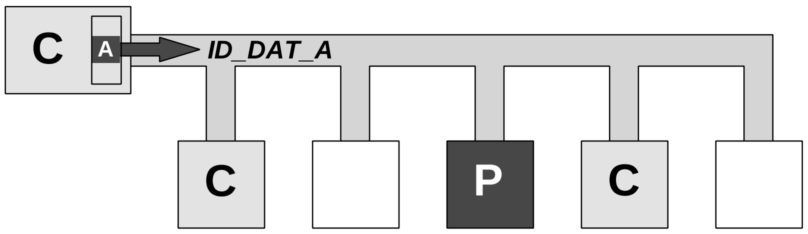
- bus arbitrating: management of access to the transmission medium.
- production/consumption functions.

Any WorldFIP station can simultaneously perform these two functions, but at any given instant only one station can perform the function of active bus arbitrating.

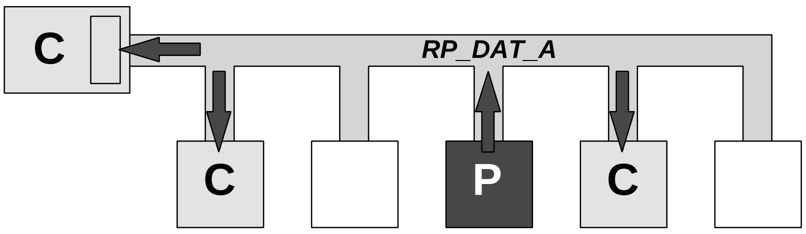


The bus arbitrator (BA) has the resources needed to scan variables at the instants defined when the system was configured. The bus arbitrator has a scanning table with a list of identifiers to circulate on the bus.

The bus arbitrator's job is relatively simple. It uses the question frame ID_DAT to broadcast on the bus the name of an identifier. This question is simultaneously recorded by all the data link layers of all stations connected to the bus. One and only one of these stations recognizes itself as being the producer of the identifier. One or more other stations recognize that they are consumers of the variable.



The producer of the variable then broadcasts the value of the identifier in a response frame (RP_DAT). This value is simultaneously captured by all consuming stations. The bus arbitrator then goes on to the next identifier in the scanning table and the same question-response cycle is repeated.



Transmission on the bus of a value produced or reception of a value consumed involves the buffer transfer mechanism. Buffer transfer takes place at the initiative of the bus arbitrator; it is totally independent of any user activity.

When new stations that consume one or more variables are connected to the bus no extra time is needed to supply those variables to the new stations. The mechanism remains the same.

Timers in data link layers' status machines constantly monitor activity on the bus. Thus if a response frame (RP_DAT) is lost or is slow in coming, consumers return to a status of waiting for a question frame and ignore all other types of frames.

3.4 Bus arbitrating tables

The bus arbitrator is the network's orchestra conductor. When the system is configured the bus arbitrator is given the list of variables to scan and the periodicities associated with each of these variables. If this configuration has been validated, and if it respects the time constraints for exchanges of the variables listed, the bus arbitrator infinitely repeats the mechanism described above.

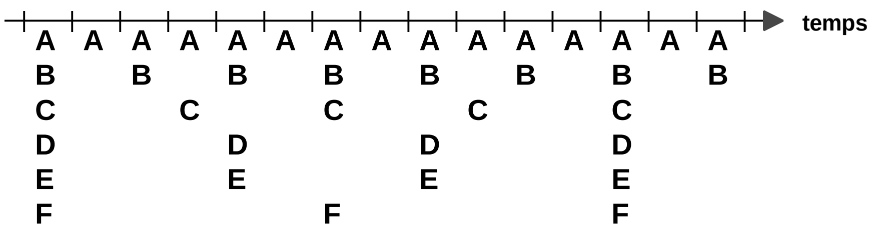
Variable scanning is deterministic. WorldFIP can guarantee that a variable with a given periodicity will be scanned at the proper instant.

The example below indicates how to configure the bus arbitrator to guarantee deterministic scanning.

Variable	Périodicity	Type	Time (µs)
A	**5**	**INT_8**	**170**
B	**10**	**INT_16**	**178**
C	**15**	**OSTR_32**	**418**
D	**20**	**SFPOINT**	**194**
E	**20**	**UNS_32**	**194**
F	**30**	**VSTR_16**	**290**

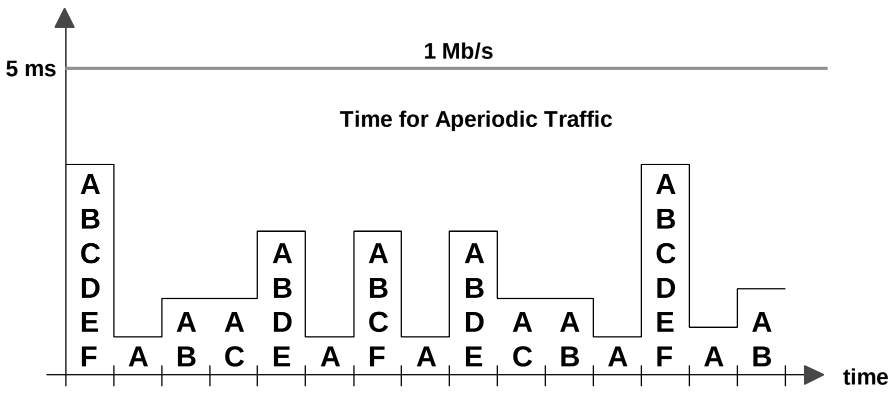
The bus arbitrator must scan six periodical variables. For each variable the arbitrator knows: its periodicity expressed in milliseconds, its applicative type (ex: 8-bit integer, chain of 32 characters, etc.). Using the transmission time (here 1 Mb/s) and the turnaround time, the bus arbitrator can calculate the time needed for an elementary transaction made up of the transmission time for a question frame, followed by the transmission time for the associated response frame. (The time column is expressed in mseconds. It corresponds to the length of transactions, given a turnaround time of 20 mseconds.)

The figure below shows a possible distribution of the identifiers A...F on a time axis, as a function of the periodicity of each of the variables. Each period of time constitutes an elementary cycle. Elementary cycles in this example are all of the same length: 5 milliseconds.



In the first elementary cycle the bus arbitrator scans all the variables. In the second it scans only the variable A.

Using the information above, network load can be calculated as follows:

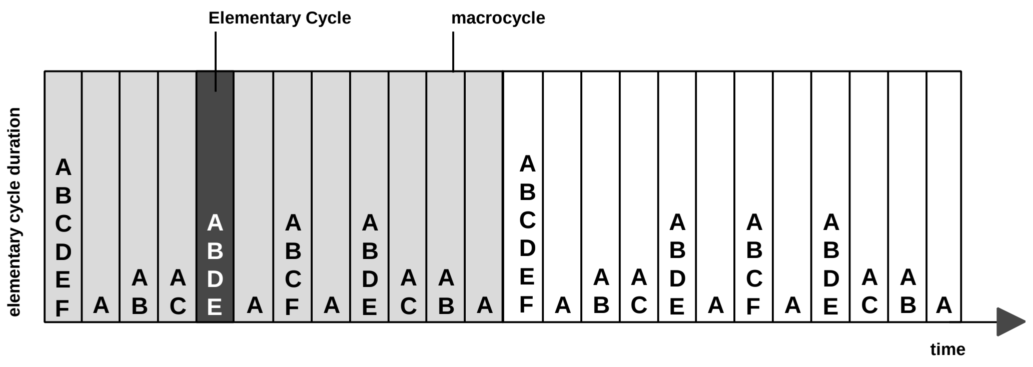


The vertical axis corresponds to the scanning time for each elementary cycle. This time is the sum of the times of all transactions making up the elementary cycle. The horizontal axis corresponds to the flow of elementary cycles in time.

Note that no elementary cycle goes beyond the 5 ms limit. A bus arbitrating table that respects the periodicities stated can be constructed.

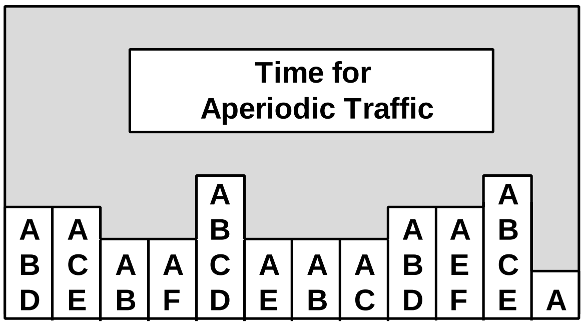
The time separating the horizontal 1 Mb/s bar from the end of scanning of each elementary cycle is free for aperiodic traffic.

The bus arbitrator will infinitely perform the same elementary cycles, or more precisely the same macrocycle:



The macrocycle corresponds to a juxtaposition of elementary cycles. The number of elementary cycles in a macrocycle is equal to the **L**east **C**ommon **M**ultiple of the periodicities divided by the **H**ighestt **C**ommon **D**enominator of the periodicities..

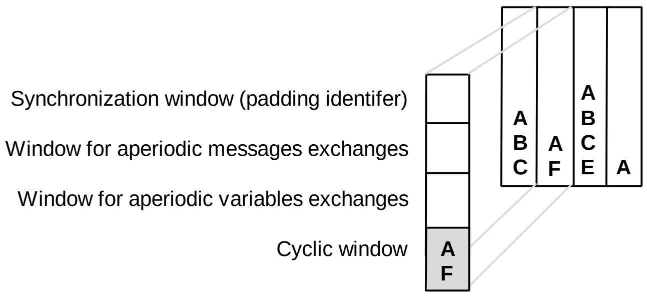
A more uniform distribution of identifiers within elementary cycles could appear as follows:



When the producer of the variable D scanned in the first elementary cycle has transmitted its value, the bus arbitrator can use the remaining time to fulfill requests for aperiodic transfers. If no such request has been made, the bus arbitrator transmits padding identifiers until the end of the elementary cycle, to indicate to other stations connected to the network that it is still functioning. A padding identifier is an identifier not produced by any station.

This type of macrocycle is composed of synchronous elementary cycles. The bus arbitrator can also manage asynchronous elementary cycles. In this case there is no emission of padding identifiers. Initial periodicities are not respected, as the identifiers may be scanned more frequently.

Each synchronous elementary cycle has four windows:


Synchronization window (padding identifer)
Window for aperiodic messages exchanges
Window for aperiodic variables exchanges
Cyclic window
A F
A B C
A F
A B C E
A

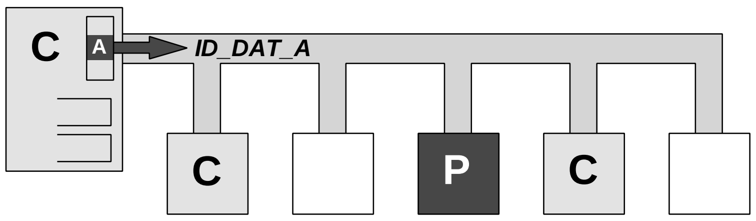
3.5 Requests for aperiodic transfer

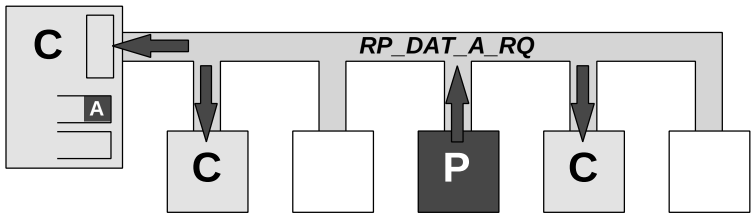
Not all variables in a distributed application are necessarily included in the bus arbitrator's cyclical scanning tables. Some variables may be exchanged only occasionally. WorldFIP provides a mechanism for aperiodic transfer.

This mechanism has three stages:

stage one:

The bus arbitrator broadcasts a question frame concerning the identifier A in a window set aside for periodic traffic. The producer of the variable A responds with the corresponding variable and sets an aperiodic request bit in the control field of its response frame (RQ). The bus arbitrator notes identifier A in a queue of requests for variable transfers.

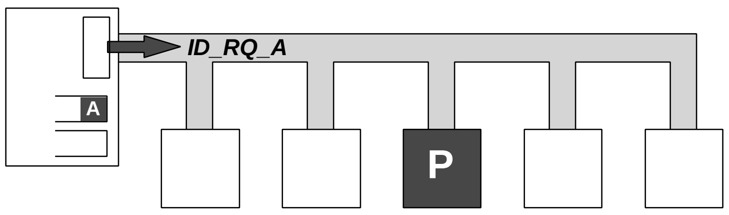


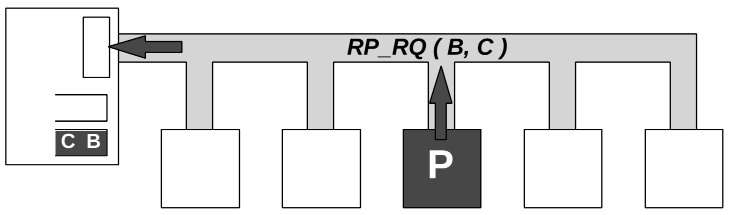


Two priority levels can be stated when the request for aperiodic transfer is made: Urgent or Normal. The bus arbitrator has two queues, one for each level of priority.

stage two:

In the time window set aside for aperiodic variables traffic the bus arbitrator uses an identification request frame (ID_RQ) to ask the producer of the identifier A to transmit its request. The producer of A responds with an RP_RQ frame (list of identifiers). This list of identifiers is taken into account by the bus arbitrator, which stores the list in another queue.



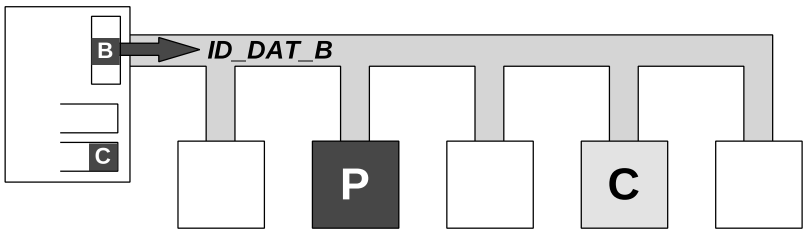


A response request frame can contain from 1 to 64 identifiers.

The bus arbitrator may transmit the identification request frame very soon after the request is noted, or much later. The service delay depends on periodic traffic load and requests in progress.

stage three:

This third and last stage takes place in an aperiodic window and consists of filling the requests for aperiodic transfers that are stored in the bus arbitrator's queue. The bus arbitrator fills one or more requests depending on the time available in the elementary cycle for this type of traffic. To do so, the arbitrator uses the same mechanism described above: the arbitrator broadcasts an identifier, which is then produced by its sole producer and consumed by all stations subscribing to the variable.



A station that requests an aperiodic transfer can be:

- the producer of the variable
- the consumer
- producer and consumer
- neither producer nor consumer (third-party variables)

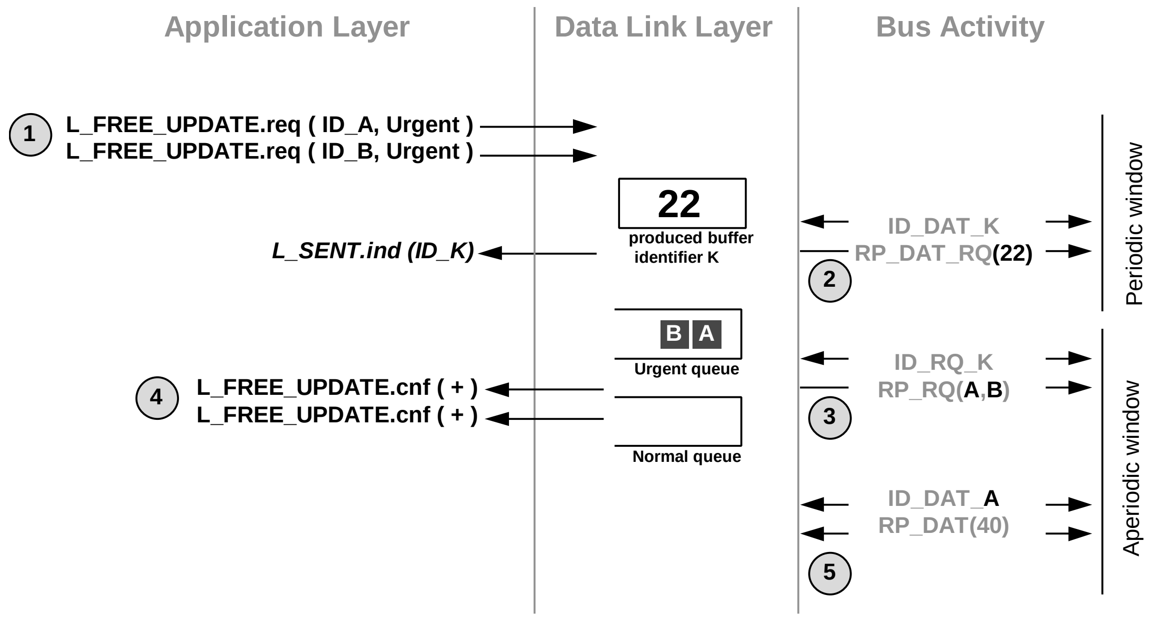
A station can only request aperiodic transfers using responses to variables that it produces and that are configured in cyclic traffic.

There are two types of aperiodic transfer requests:

- free requests (L_FREE_UPDATE.req (list_ID, priority))

The application layer calls upon this service by indicating to the data link layer the identifier or identifiers to be circulated in an aperiodic window. This data link layer service adds to a queue the stated identifier or identifiers. This queue contains all identifiers requested.

The request will be carried to the bus arbitrator by the first identifier the arbitrator calls for that is produced by the station.



Stages:

1 - The application layer makes a request for an urgent aperiodic transfer concerning identifiers A and B. The data link layer notes these two identifiers in its queue of urgent aperiodic transfer requests.

2 - When the bus arbitrator requests the identifier K, which is produced by the station, the data link layer responds with the value K and sets bit RQ.

3 - In an aperiodic window the bus arbitrator requests the contents of the urgent queue. The data link layer responds with the list (A, B).

4 - The data link layer then confirms the free requests for aperiodic transfer. This confirmation does not mean the requests have been filled.

5 - Finally, and still in an aperiodic window, the bus arbitrator fills the aperiodic requests concerning the identifiers A and B.

- specified requests (L_SPEC_UPDATE.req (ID_Spec, ID_requested))

  This mechanism functions in almost the same fashion as the free mechanism. The request is transmitted to the bus arbitrator when the arbitrator calls for the specified identifier (ID_Spec) given as a parameter.

  The requested identifier or identifiers are in this case not stored in a common queue. They are indicated in a request buffer associated with the identifier.

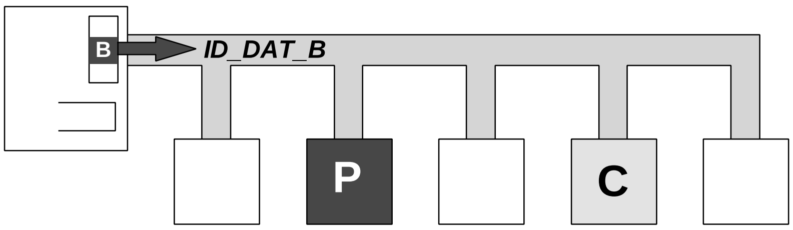
  This service is used only by the application layer for the mechanism for spatial consistency of data. This mechanism is internal to the application layer.

3.6 Unacknowledged message transfer request

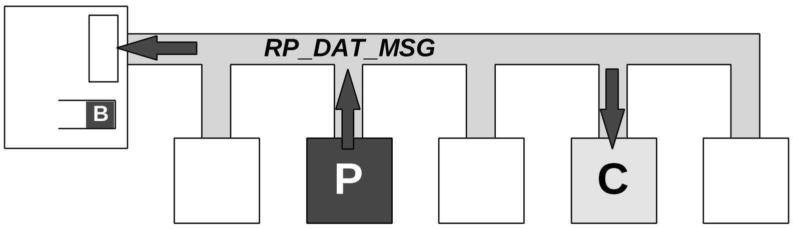
WorldFIP's data link layer provides for the transfer of unacknowledged messages in point-to-point or broadcast mode. The principle for an aperiodic message transfer request is similar to that of an aperiodic variable transfer request. There are three stages to the mechanism:

first stage:

The bus arbitrator calls for identifier A. The producer of A responds with the value of A and indicates in the frame control field (MSG bit) that it has a request for a message transfer.



Producteur de B

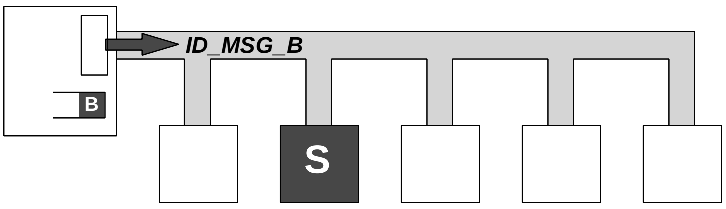


Requête message

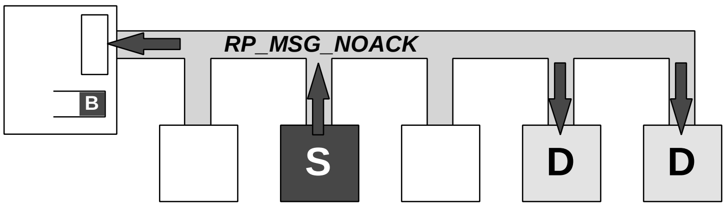
The bus arbitrator notes the identifier A that carried the request in a message request queue.

second stage:

In a window for the scanning of aperiodic message transfer requests, the bus arbitrator gives the producer of the identifier A the "right to speak". The producer of A then transmits its message. This message includes its address and the address of the destination station or stations in a frame of the RP_MSG_NOACK type.



Source message

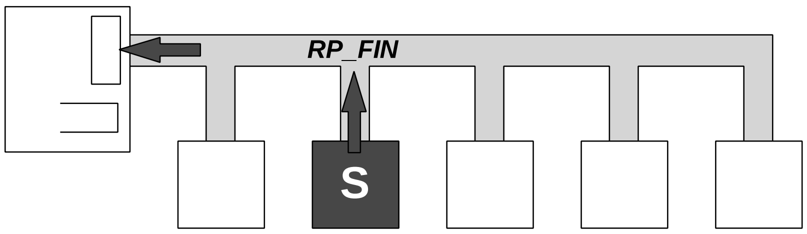


Destinataires message

The bus arbitrator then waits to receive a frame indicating that the message transfer transaction is finished.

third stage:

The third stage returns control of the bus to the bus arbitrator.



After sending its message on the bus the transmitter of the message transmits an RP_FIN frame.

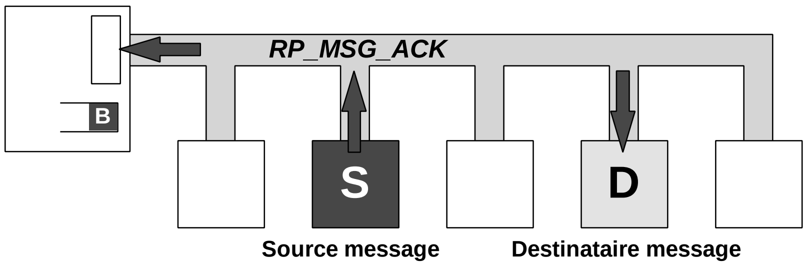
When the bus arbitrator gives the producer of a message the right to use the network, it does not know in advance if this request is acknowledged or unacknowledged. Thus the arbitrator must verify that there is enough time in the elementary cycle to fulfill the request.

The bus arbitrator sets a timer to avoid waiting indefinitely for a frame indicating that the message transaction is finished.

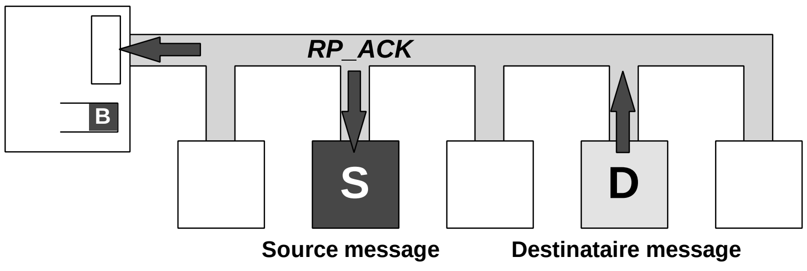
3.6.1 Acknowledged message transfer requests

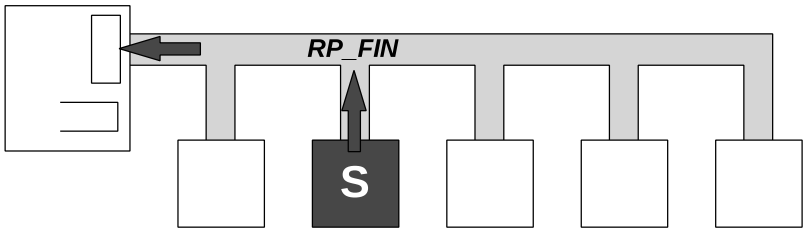
WorldFIP's data link layer also provides for acknowledged message transfer services that make point-to-point exchanges more reliable.

The operating principle is almost identical to unacknowledged service in the first two stages. The message transmitter responds with an RP_MSG_ACK frame instead of an RP_MSG_NOACK frame. The exchange is point-to-point.



The destination station responds with an acknowledgement frame and the transmitter then signals the end of its message transaction to the bus arbitrator (RP_FIN).





This service uses a modulo 2 message numbering mechanism that allows the destination station to detect the loss or duplication of a message.

The number of repetitions in case of absence of an acknowledgement is between 0 and 2. This parameter is a global operating parameter of the WorldFIP network.

3.7 Conformance classes

Not all data link layer services are included in all WorldFIP stations. The table below shows the 9 conformance classes that have been defined:

Services	Classes 1	2	3	4	5	6	7	8	9
buffer transfert	▨	▨	▨	▨	▨	▨	▨	▨	▨
buffer write	▨	▨	▨	▨	▨	▨	▨	▨	▨
buffer read	□	▨	▨	▨	▨	▨	▨	▨	▨
free update request	□	□	□	▨	▨	□	□	▨	▨
specified update request	□	□	▨	□	▨	□	□	□	▨
unacknowledged message	□	□	□	□	□	▨	▨	▨	▨
acknowledged message	□	□	□	□	□	□	▨	▨	▨

▨ supoorted service

□ no supported service

All WorldFIP stations include at least the buffer transfer and buffer write mechanisms.

3.8 Frames

buffer transfer

The buffer transfer mechanism uses two types of frames: ID_DAT and RP_DAT. Every WorldFIP frame transmitted is sandwiched between information from the physical layer. The frame is placed between a DTR field ("dÝbut de trame" or beginning of frame) and an FTR field ("fin de trame" or end of frame). These fields are explained in the section on the physical layer.

All WorldFIP frames begin with a control byte that is used by network subscribers to recognize the type of frame they are receiving. This control field is used to code variable transfer requests, acknowledgement frames, etc.

All WorldFIP frames end with two bytes (FCS: Frame Check Sequence) used by the frame receiver to verify the integrity of the frame received. FCS is the result of a polynomial operation performed on the preceding bytes.

All question frames, be they ID_RQ or ID_MSG frames, are of the ID_DAT type. They are distinguished only by a few bits in the control field.

ID-DAT	FSS	Controle	Identifier	FCS	FES
	2 bytes	1 byte	2 bytes	2 bytes	1 bytes

The format of a frame transmitted as a response to an ID_DAT frame is shown below:

RP-DAT	FSS	Controle	DATA	FCS	FES
	2 bytes	1 octet	n bytes (n <= 128)	2 bytes	1 bytes

The DATA field can contain up to 128 bytes from the application layer. The control field indicates if there are any aperiodic variable or message transfer requests.

response request transfer (RP_RQ)

When a station that has made a variable transfer request receives an ID_RQ frame it responds with an RP_RQ frame coded as follows:

RP-RQ	FSS	Controle	List of identifiers	FCS	FES
	2 bytes	1 octet	n * 16 bits (n <= 64)	2 bytes	1 bytes

a request frame can hold 64 16-bit identifiers.

response message transfer (RP_MSG_xxx)

When a station that has made a message transfer request receives an ID_MSG frame it responds with an RP_MSG_NOACK or an RP_MSG_ACK frame coded as follows:

RP-MSG-xx	FSS	Controle	dest. adr.	srce adr.	Message	FCS	FES
	2 bytes	1 byte	3 bytes	3 bytes	max 256 bytes	2 bytes	1 bytes

a bit in the control field indicates if the message transfer is acknowledged or unacknowledged. The destination and source fields show the addresses of the communicating entities.

response acknowledgement transfer (RP_ACK)

When a destination station receives a message with a request for acknowledgement it transmits an acknowledgement frame.

RP-ACK	FSS	Controle	FCS	FES
	2 bytes	1 byte	2 bytes	1 bytes

This frame is very short, as the acknowledgement information is contained in the control field.

end of message transaction response frame (RP_FIN)

When a message has been transmitted the sender, after waiting for an acknowledgement if necessary, transmits an end of message transaction frame:

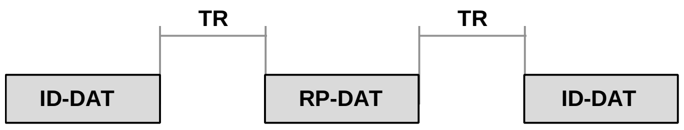
RP-FIN	FSS	Controle	FCS	FES
	2 bytes	1 byte	2 bytes	1 bytes

This frame is very short, as the transaction finished information is contained in the control field.

3.9 Buffer transfer performance levels

WorldFIP's efficiency can be calculated using the frame coding explained above. All WorldFIP transactions are made up of the exchange of two frames: an ID_DAT frame followed by an RP_DAT frame.

The RP_DAT frame should appear within a given time. This time is called the turnaround time. Turnaround time is the time elapsed between the end of reception of one frame and the beginning of transmission of the following frame.



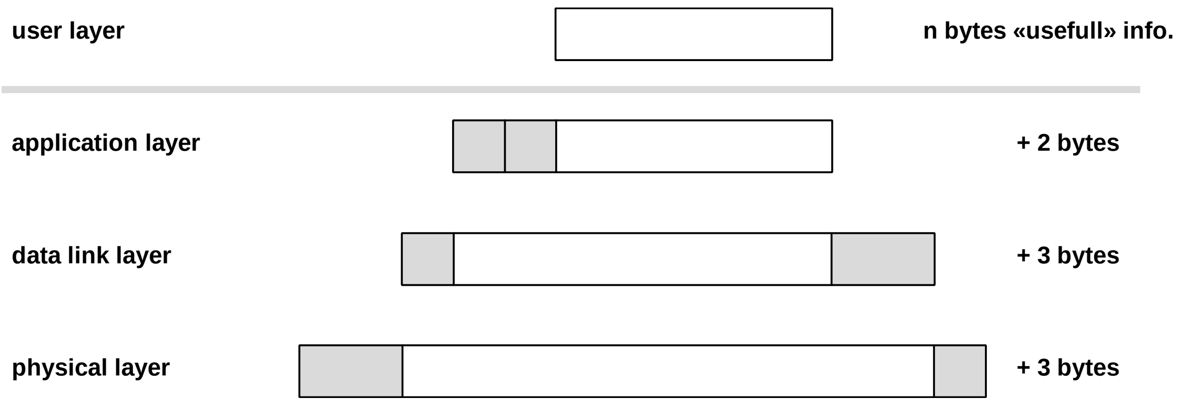
Turnaround time (TR) is defined in the network management standard. 10 TMAC <= TR <= 70 TMAC.

TMAC corresponds to the time needed to transmit a physical layer symbol ("1", "0", "V+", "V-", "EB+" or "EB-".

· at 31.25 kb/s TR is between 320 µs and 22.4 µs
· at 1 Mb/s TR is between 10 µs and 70 µs
· at 2.5 Mb/s TR is between 4 µs and 28 µs

As we shall see, this parameter can be very important in determining performance levels.

Each layer adds service information to what it receives from the layer above. The three layers add a total of 64 bits, no matter what the length of the useful information.



Total transaction time includes times needed for:

· transmission of a question frame
· turnaround time
· transmission of the response frame
· turnaround time

- question frame transmission time: 61TMAC (constant)
- response frame transmission time: 61 TMAC + **n***8 TMAC (**n** = number of user bytes)
- turnaround time = 2 * **TR** TMAC (TR >= 10 and TR <= 70)

Efficiency is equal to the transmission time of the useful information divided by total transaction time.

Efficiency = **n***8 TMAC/(64 TMAC + **TR** TMAC + 64 TMAC + n*8 TMAC + **TR** TMAC).

Efficiency = **n***8 TMAC/(128 TMACÿ+ **2TR** TMAC + **n***8 TMAC)

- **calculations based on TR = 10**

length useful information	efficiency	useful throughput at 1 Mb/s	useful throughput at 2.5 Mb/s
n = 1	5.12 %	51.28 kb/s	128.21 kb/s
n = 2	9.75 %	97.56 kb/s	243.90 kb/s
n = 4	17.77 %	177.78 kb/s	444.44 kb/s
n = 8	30.18 %	301.89 kb/s	754.72 kb/s
n = 16	46.37 %	463.77 kb/s	1159.42 kb/s
n = 32	63.36 %	633.66 kb/s	1584.16 kb/s
n = 64	77.57 %	775.76 kb/s	1939.39 kb/s
n = 128	87.37 %	873.72 kb/s	2184.30 kb/s

- **calculations based on TR = 70**

length useful information	efficiency	useful throughput at 1 Mb/s	useful throughput at 2.5 Mb/s
n = 1	2.89 %	28.99 kb/s	72.46 kb/s
n = 2	5.63 %	56.34 kb/s	140.85 kb/s
n = 4	10.66 %	106.67 kb/s	266.67 kb/s
n = 8	19.27 %	192.77 kb/s	481.93 kb/s
n = 16	32.32 %	323.23 kb/s	808.08 kb/s
n = 32	48.85 %	485.55 kb/s	1221.37 kb/s
n = 64	65.64 %	656.41 kb/s	1641.03 kb/s
n = 128	79.25 %	792.57 kb/s	1981.42 kb/s

When the useful information is very short, turnaround time is an important factor. If n = 1 efficiency is almost two times lower when TR = 70 ms.

WorldFIP does not escape the rule for calculating efficiency. The shorter the useful information, the lower the efficiency. Thus it is preferable to increase useful throughput by combining information in a data structure for transmission.

When comparing WorldFIP's efficiency with that of other types of networks it must be remembered that information produced is broadcast on the bus to all consumers. When information is destined to a large number of consumers, a single transaction is all that is needed to refresh the buffers of these consumers. In other types of networks there would be as many transactions as there are consumers.

In a **1 Mb/s** network with a turnaround time of **10**ms it is possible to scan the following numbers of variables in a **5 ms** elementary cycle:

- 32	1-byte variables	- 18	16-byte variables
- 30	2-byte variables	- 12	32-byte variables
- 27	4-byte variables	- 7	64-byte variables
- 23	8-byte variables	- 4	128-byte variables

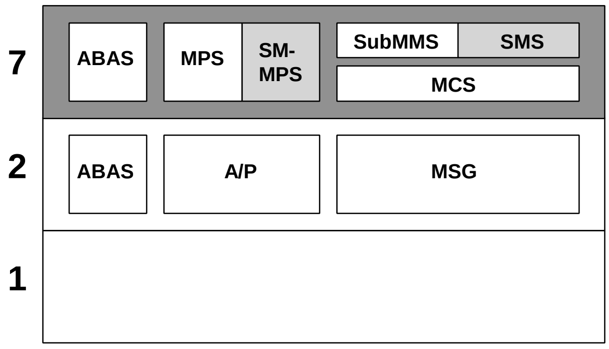
In a **2.5 Mb/s** network with a turnaround time of **10**ms it is possible to scan the following numbers of variables in a **20 ms** elementary cycle:

- 320	1-byte variables	- 181	16-byte variables
- 304	2-byte variables	- 123	32-byte variables
- 277	4-byte variables	- 75	64-byte variables
- 235	8-byte variables	- 42	128-byte variables

4. APPLICATION LAYER

WorldFIP application layer services are divided into three distinct groups:

- ABAS (bus arbitrator application services)
- MPS (manufacturing periodical/aperiodical services)
- subMMS (subset of messaging services)



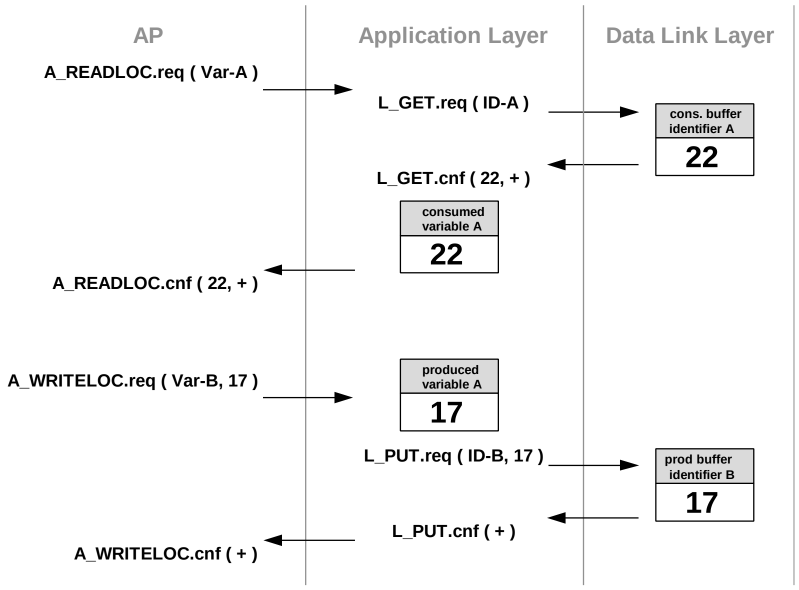
This section addresses only MPS.

The MPS application layer provides the user with:

- · local read/write services
- · remote read/write services
- · variable transmission/reception indications
- · information on the freshness of information consumed
- · information on the spatial and temporal consistency of data

4.1 Local read/write

WorldFIP's application layer provides users with local variable read/write services. These services use data link layer services L_PUT.req and L_GET.req to place values in buffers or remove values from buffers. These services generate no traffic on the bus.

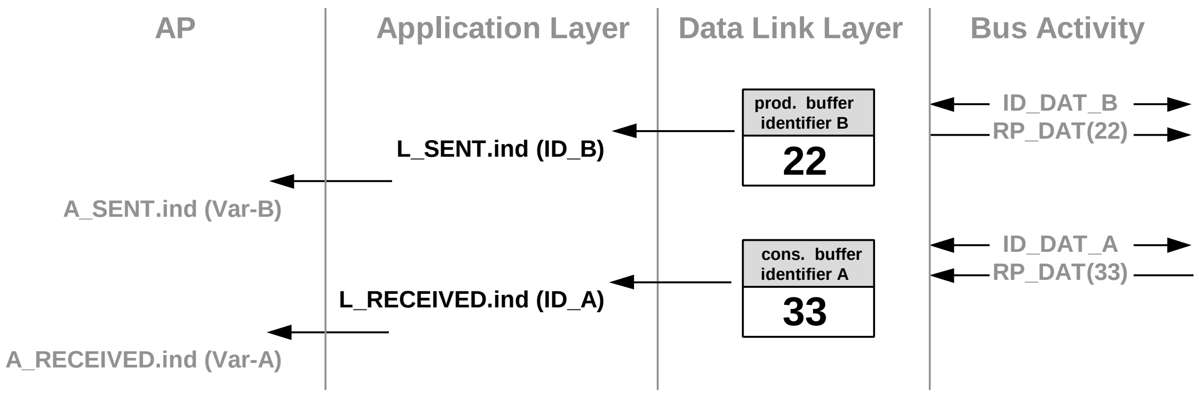


The mechanism shown above is valid only for non-resynchronized variables.

4.2 Reception/transmission indications

If the user so chooses he can be informed of the transmission or reception of an identified variable. He can then use this information, for example, to synchronize himself with a piece of network information.

When the application layer receives a transmission or reception indication for a produced or consumed variable the application layer forwards this indication to the user.

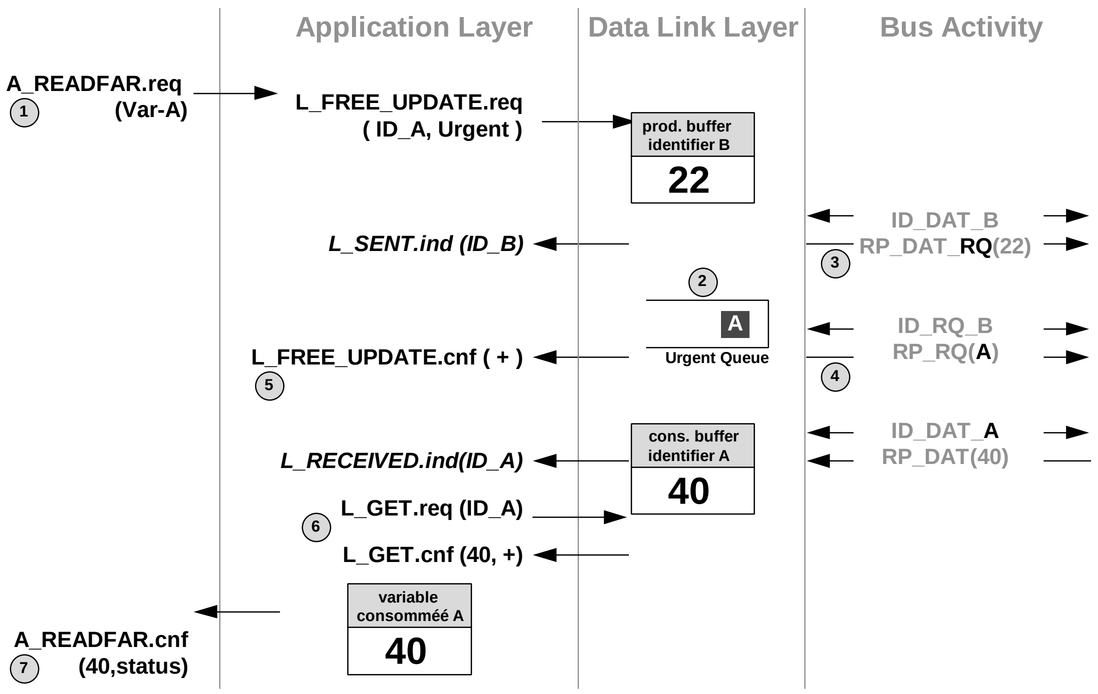


When the station is configured it is necessary to indicate for each produced or consumed variable whether or not said variable generates one of these indications.

4.3 Remote read/write

WorldFIP's application layer provides remote read/write services for identified variables.

Mechanism for the remote reading of variable A:



Remote reading takes place in several stages:

1 - The user makes a request for remote reading of the variable B: A_READFAR.req(var_B). The application layer then requests a free updating using L_FREE_UPDATE.req(ID_B).

2 - The identifier ID_B is added to a queue of aperiodic transfer requests.

3 - Using the first identifier produced by the station that is requested by the bus arbitrator, the data link layer responds by setting the RQ bit in the control field.

4 - Then in an aperiodic scanning window the bus arbitrator requests the contents of the aperiodic variable transfer request queue.

5 - Transmission of this queue triggers a confirmation of the request to update.

6 - Later, and once again in an aperiodic window, the bus arbitrator will scan the variable Var_B. An indication of the reception of this variable will be sent to the application layer. The application layer will then use a local read service to access the latest value received.

7 - Confirmation, when positive, contains the value of the variable. A timer in the application layer is used to detect excessive waiting times.

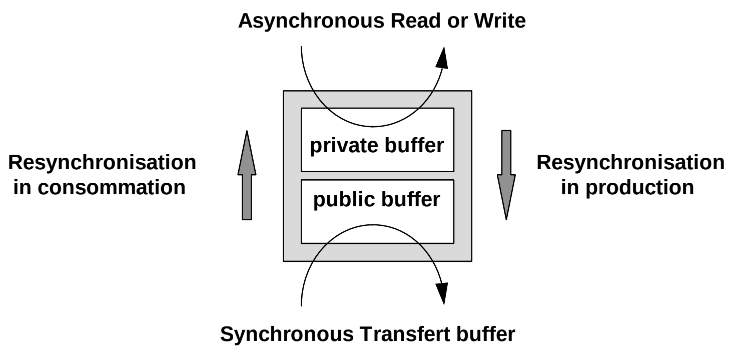
The remote writing mechanism WRITEFAR.req(var) functions in almost the same manner. The application layer begins by updating the buffer containing the variable, then makes a request for its transfer. When the application layer receives an indication of transmission of the variable it confirms the service.

4.4 Resynchronization

The application layer makes it possible for asynchronous application processes to participate in distributed synchronous applications by providing resynchronization services in production and consumption.

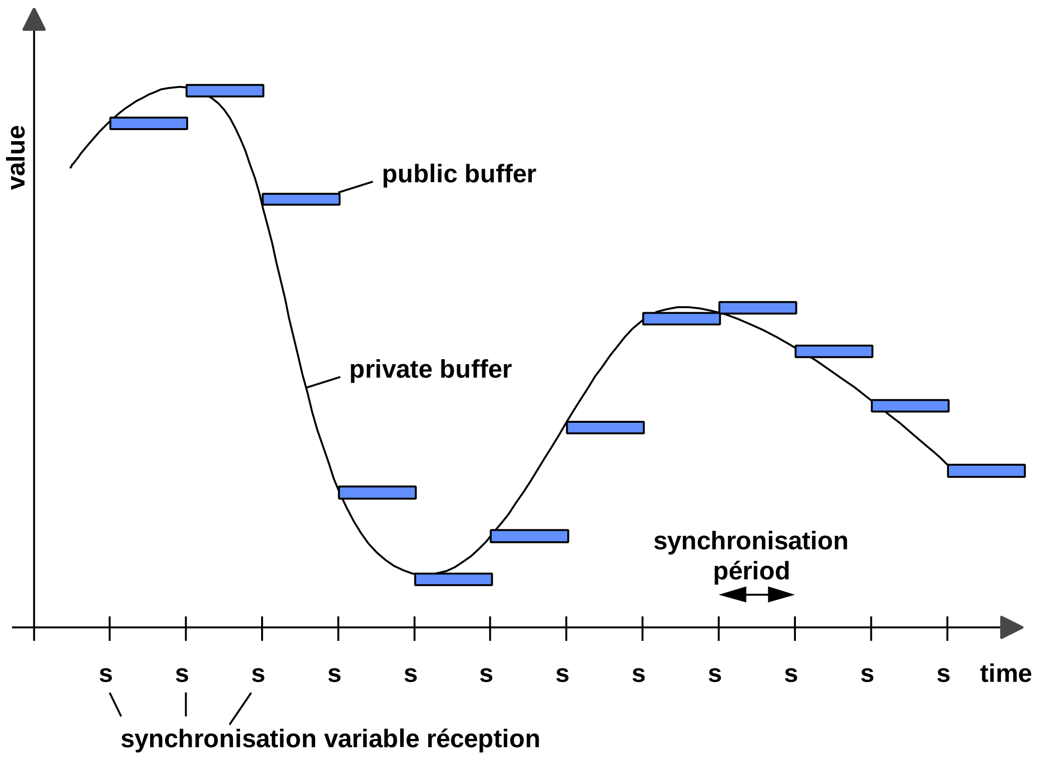
The resynchronization mechanism uses double memorization on the application layer level: a private buffer accessible only through the producing or consuming application process, and a public buffer accessible through the network, which removes or deposits values.

The consumption resynchronization mechanism consists of recopying the contents of the public buffer in the private buffer upon reception of a synchronization variable.


Asynchronous Read or Write
Resynchronisation in consommation
private buffer
public buffer
Resynchronisation in production
Synchronous Transfert buffer

The production resynchronization mechanism consists of recopying the contents of the private buffer in the public buffer upon reception of a synchronization variable.

The figure above shows the contents of private and public buffers as a function of time for a produced resynchronized variable.



Using a local write service the user updates the variable's private buffer (curve on the figure).

Each time a resynchronization variable is received the application layer recopies the private buffer in the public buffer (shaded rectangles). The value in this buffer will be held until another synchronization variable is received.

The synchronization period corresponds to the periodicity of the synchronization variable in the bus arbitrator's scanning tables.

Resynchronization is an optional mechanism. When the station is configured it must be indicated for each produced or consumed variable if the variable is resynchronized. If the variable is indeed resynchronized the associated resynchronization variable must also be indicated.

When a variable is resynchronized in production, calling upon the service A_WRITELOC.req places the new value in the private buffer without calling on the data link layer buffer updating service.

When a variable is resynchronized in consumption, calling on the service A_READLOC.req removes the new value from the private buffer without calling on the data link layer buffer read service.

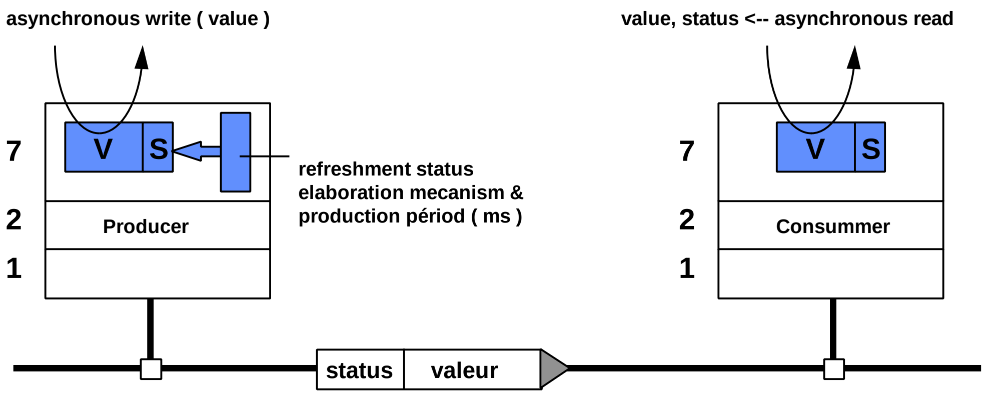
4.5 Promptness and refreshment

When a user reads a variable in his local communications entity he can at the same time receive qualitative information concerning the freshness of the variable. This information is boolean, and can be elaborated for any produced or consumed variable.

Promptness and refreshment can be asynchronous, synchronous or punctual.

Asynchronous refreshment

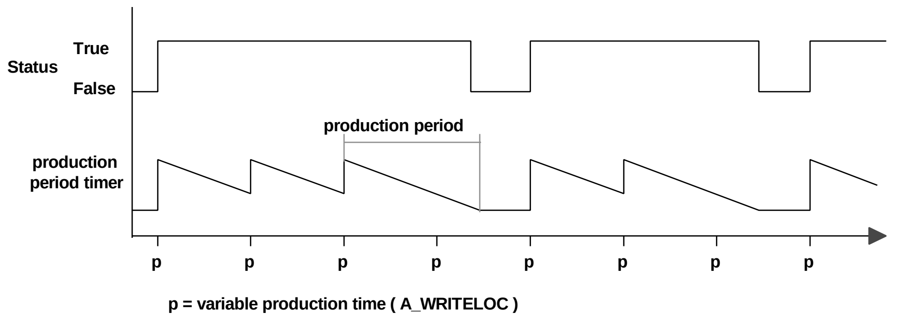
Refreshment statuses are worked out by the application layers of producers of identified variables. For each variable produced it should be indicated whether or not a refreshment status is provided. If there is a refreshment status, the period of production associated with the variable must also be indicated.


asynchronous write (value)
value, status <-- asynchronous read
7
2
1
V
S
Producer
refreshment status
elaboration mecanism &
production période (ms)
7
2
1
V
S
Consummer
status
valeur

The producer of the station on the left produces new values at its own rhythm in its communications entity. With each new write operation the producer's application layer uses the production delay to elaborate a refreshment status. A "true" status indicates the producer's application process is functioning properly.
All consumers of this variable consume an object made up of a value and a refreshment status.

Each consumer uses a read service to access the status and learn if the producer of the variable has respected the production delay attached to the variable.

The diagram below shows the mechanism for elaborating an asynchronous refreshment status using time elapsed and the instants when new values are written.

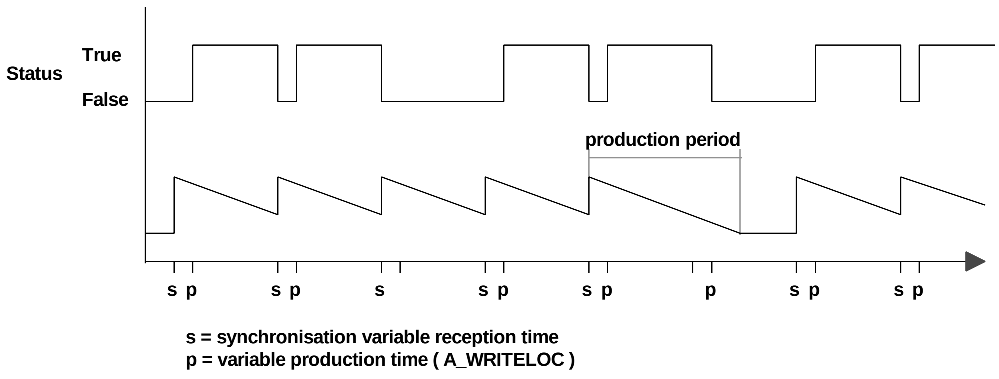


Whenever a new value is written the application layer sets the timer associated with the variable with the value of the production period. Status is "true" as long as this time has not expired.

Synchronous refreshment

If a synchronous refreshment status is to be elaborated for a variable, the production period and the synchronization variable for elaborating the status must be indicated.

The diagram below shows the mechanism for elaborating a synchronous refreshment status using time elapsed and the instants when new values are written and associated synchronization variables received.

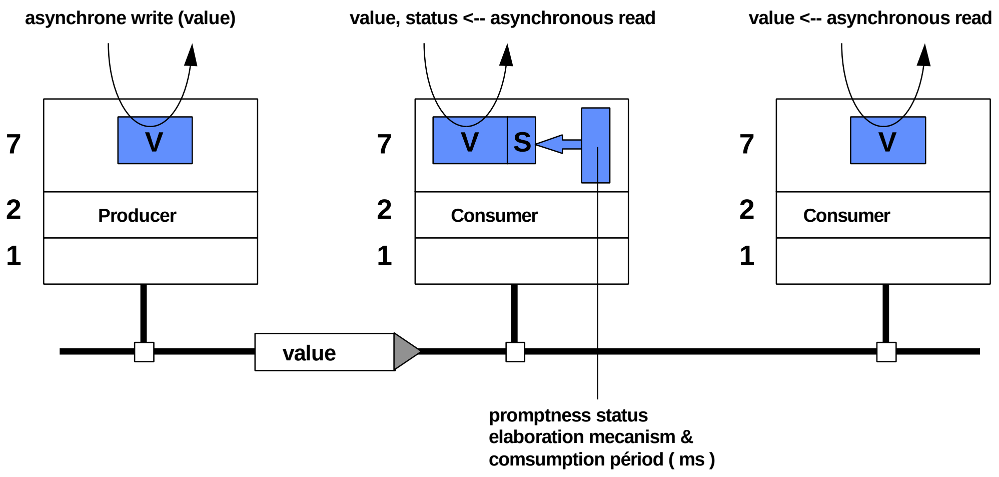

Status
True
False
production period
s p
s p
s
s p
s p
p
s p
s p
s = synchronisation variable reception time
p = variable production time (A_WRITELOC)

Each time a synchronization variable is received the timer is reset with the value of the production period and status becomes "false".

If the timer has not expired when the user writes a new value the status becomes "true" and remains "true" until the timer expires or a new synchronization variable is received.

Asynchronous promptness

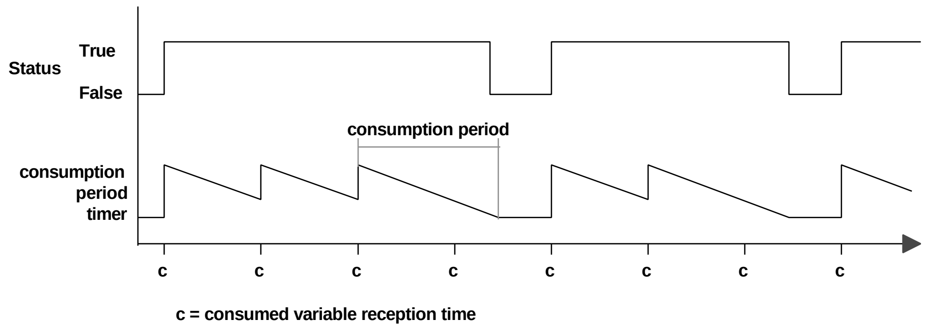
Asynchronous promptness statuses are elaborated by the application layers of consumers of identified variables. For each variable consumed in each communications entity one must indicate whether or not a promptness status is elaborated. If a promptness status is required, the consumption period associated with the variable must also be indicated.



When an asynchronous promptness status is true it signifies that the buffer transfer mechanism is working properly, i.e., the bus arbitrator has respected the variable's scanning period and the data link layers of the producer and consumer are operating properly.

In the example below one station elaborates an asynchronous promptness status for a variable and the other does not. When the first station reads the value it obtains both the value of the variable and the promptness status. Since the value is made up of the <value, refreshment status> pair, the station knows whether or not the producer has respected the period of production and whether or not the buffer transfer mechanism is functioning correctly.

The diagram below shows the mechanism for elaborating an asynchronous promptness status using time elapsed and the instants when new values are received.



When a communications entity receives a new value it sets the asynchronous promptness timer associated with the variable consumed with the value of the consumption period. The promptness status becomes "true" and remains so until the timer expires.

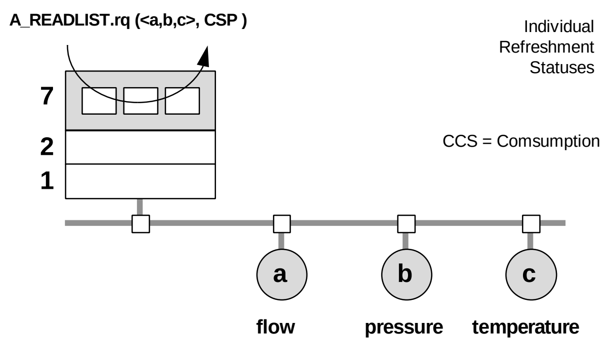
4.6 Spatial and temporal consistencies

WorldFIP's application layer provides a list reading service. A list is made up of a set of consumed variables. This service reads globally in the application layer all the values making up the list. It then provides the user with a "super-status" informing him on the freshness of the information he consumes. It can also provide a spatial consistency status indicating that all copies of this list are identical in all the communications entities that consume the list.

Temporal production consistency

Temporal consistencies are boolean statuses that can be optionally elaborated by the consumers of a list of identified variables. Temporal consistencies can be in reference to production or consumption. They can be asynchronous or punctual.

In the example below three sensors (flow, temperature, pressure) each produce a value with which a refreshment status is associated. The station consumes the list made up of these three variables.



		t1	t2	t3	t4
Individual Refreshment Statuses	a	T	T	T	T
	b	T	F	T	T
	c	F	T	T	T
	CSP	F	F	T	T

CCS = Comsumption Consistency Status = logical AND

The application layer of the consumer of the list consumes three <value, status> pairs and applies a logical AND to the individual refreshment statuses. If all these statuses are true, the temporal production consistency status is true; otherwise it is false.

The user, with a single read operation, will know if all the producers have respected the production periods associated with the variables they produce.

Temporal consumption consistency is produced by applying a logical AND to all the individual promptness statuses.

Spatial consistency

Spatial consistency is a boolean status optionally elaborated by the application layers that consume a list of variables. The principle is relatively simple: if all the copies of the list are identical the status is true, otherwise it is false.

Implementing this principle is much more complex and uses constructions of special elementary cycles, exchanges of individual spatial consistency variables, and mechanisms for restarts in case of faulty reception.

4.7 Variable types and Protocol Data Units (PDUs)

Identified variables produced or consumed by a user can be simple:

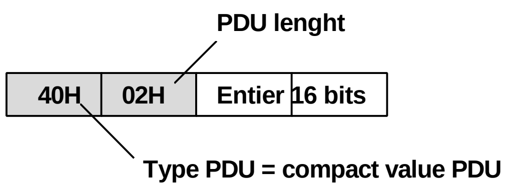
- Boolean
- Integers (two's complement notation)
- Bit string (32 bytes maximum)
- Byte string (126 bytes maximum)
- Visible string (126 bytes maximum, composed only of characters ® A...Z, a...z, 0...9, _, $ ¯)
- General time in 14 characters, coding: AAAA MM JJ HH MM SS
- 4-byte simple precision floating point (ANSI/IEEE/754 standard)
- 8-byte double precision floating point (ANSI/IEEE/754 standard)

Or structured:

- structure of simple types
- structure of structures
- table
- table of tables
- table of structures
- lists
- ...

These variables are transported on the network inside PDUs (Protocol Data Units). These different types of PDUs are coded using the ASN.1 transfer syntax (NF Z 70 005/2).

Variables are usually transported in Compact Value PDUs. The transmission of a 16-bit integer would be transported in a PDU coded as follows:



The application layer adds two bytes to the coding of the 16-bit integer. The first byte indicates the type of PDU, and the second the length of the information it contains.

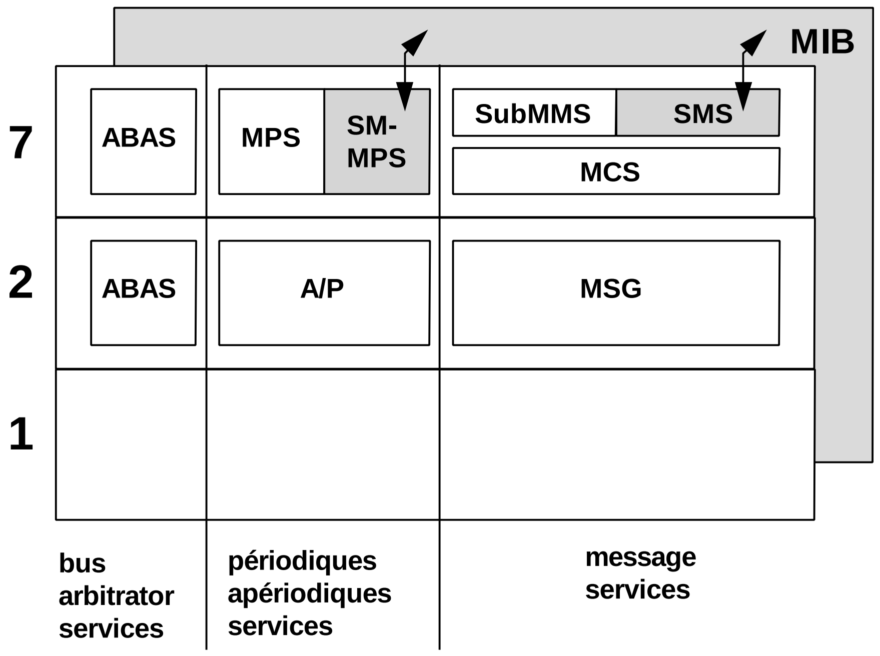
4.8 Conformance classes

The application layer defines a set of conformance classes:

- · vertical conformance
  is used to specify the application layer services supported.
- · horizontal conformance
  is used to specify the types of refreshment, promptness statuses, consistencies and resynchronizations supported by the application layer.
- · type conformance
  is used to indicate the sub-set of types supported by the station.

5. NETWORK MANAGEMENT

The network management standard describes the set of services used to manage communication, as well as the set of communications resources needed.



There are two families of network management services:

- SM_MPS: set of network management services based on MPS.
- SMS: set of network management services based on message services (MCS-MSG)

SM_MPS services are not sufficient for complete management of a remote communications entity.

Communications resources are represented in the MIB (Management Information Base). This MIB is a tree-like data structure. It describes layer by layer each of the attributes and the extent of each service.

There are three essential network management functions:

- · Management of the operating mode:

  - Start/Stop commands
  - Validation/Invalidation commands
  - Reset commands
  - Read/write functions

- · Management of the configuration:

  - Creation of objects
  - Destruction of objects
  - Starting/stopping communications entities

- · Management of failures and performance levels:

  - reading counters

- resetting counters to zero

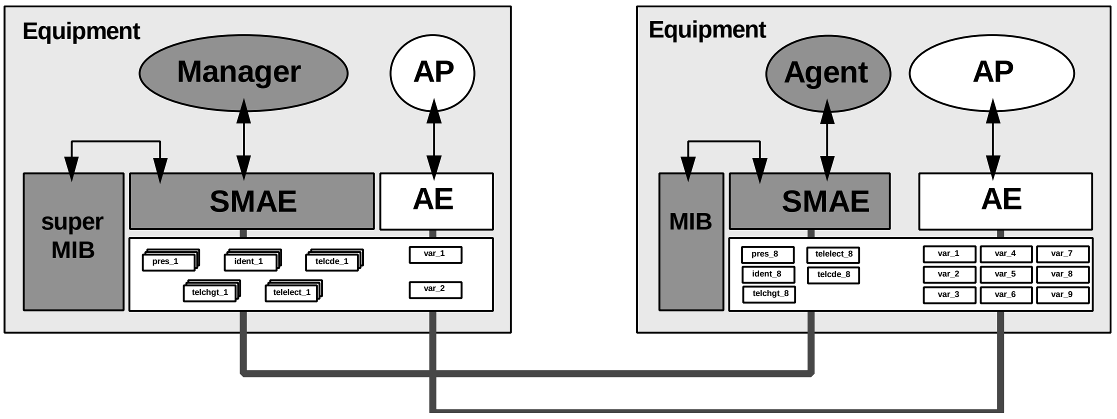
A WorldFIP station has two distinct types of application processes:

- the user AP that carries out functions of the distributed application
- the SMAP (System Management Application Process) that handles network management functions

There are two types of SMAP:

- The agent SMAP, that responds to remote calls
- The manager SMAP, that manages the network

The figure below shows two WorldFIP stations. Each contains an **AP** and a **SMAP**.



The first station is the Manager. The Manager conducts dialogue using network management variables (pres_10, ident_10, ...,) and read/write services. The Manager has a super-MIB describing the characteristics of all the stations on the network. This information enables him to configure and start up remote stations, etc. This station also has an AP, which participates in the distributed application using the variables var_1 and var_2 that it produces and consumes.

The second station is an agent, which will receive commands to start up or to remotely download a configuration through the network management variables managed by its system management application entity (SMAE). These commands, downloading, etc. will act directly on its MIB (Management Information Base). This station also has an AP, which participates in the distributed application through the variables var_1, ..., var_9 managed by its AE (Application Entity).

5.1 Local and remote services

Network management services can by local or remote. Although no physical address is used in the transfer of identified variables, all WorldFIP stations must have a physical address and an application tag. The address of a station is coded in a byte, and numbers of stations are from 0 to 255. The application tag (TagName) is a string of 32 characters (example: "blue tank sensor").

This information can be furnished locally within the station (switch, pocket, PROM, ...) or remotely through the network. A manager cannot assign a physical address to a station if the station does not have an application tag.

To assign an application tag to a device the manager and the agent must be linked in a point-to-point logical connection. Thus to assign an application tag to a device, at a given instant there must be only that device which does not have a tag. The simplest means of achieving the point-to-point logical connection is to be physically point-to-point.

A universally allocated identifier (9000 hexa) is produced by the manager. This identifier contains the string of characters making up the application tag. This identifier is consumed by the agent.

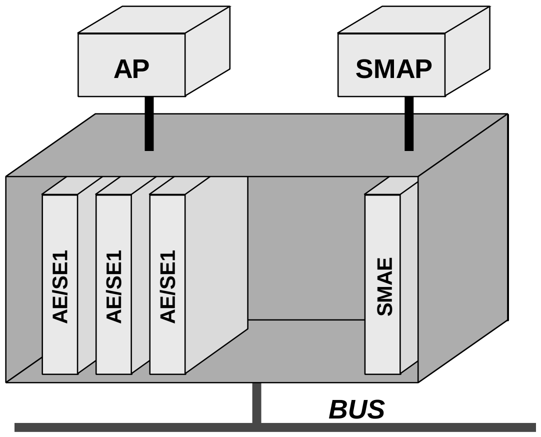
Once the agent knows its application tag (entered locally or downloaded from a distance) the manager can assign his physical address. To do so, he produces another universally allocated identifier (9001 hexa) containing the < tag, address > pair. The manager manages a table showing the correspondence between tags and station numbers.

When the agent receives the variable he compares the tagName received and the tagName stored. If these are identical the station takes the physical address into account. The station then knows its identity and its number. Using this number the station can create a set of network management variables in its System Management Application Entity (SMAE). These variables will enable the station to receive other remote downloads or commands.

5.2 Multi-AEs

As we have seen before, a station has an AP that carries on dialogue using the variables contained in an AE. A station also has a SMAP which communicates using variables contained in a SMAE.

For reasons of security or progressive start-ups, or for problems with limited SM_MPSs, a WorldFIP station can manage up to eight application entities (AE/SEi).



The communications entity contains four boxes:

- **AE/SEi** boxes: These boxes are data structures from the MIB. They each contain a certain number of user identified variables. These boxes contain the complete description of each variable: type, periodicity, refreshment status, identifier number... These boxes also contain information on the number of objects present in the box, the operating status of the box (In_Operation, Stop,...). One AE/SEi box can be in operation while another is stopped. An identifier cannot be part of two or more boxes at the same time.

- **SMAE** box: This box contains all WorldFIP variables produced and consumed for network management. These variables are used for example to load a configuration, to start up an application entity, or to validate/invalidate a set of variables.

5.3 List of SM_MPS services

SM_MPS services are used for:

- attributing a physical address and tagName
- remote downloading of identifiers
- remote reading (rereading of configuration)
- remote control operations (Start, Stop, Validation or Invalidation of one or more AE/SEs)
- remote monitoring (reading of AE/SE operating states)
- reports (all error counters and performance level evaluation devices)
- management of a presence or identification variable
- management of a list of stations present

5.4 Identifier allocation space

In order to carry out the SM_MPS services defined above, when a station learns its physical address it creates a number of network management variables whose identifier numbers are a function of this address.

Since WorldFIP identifiers are coded in 16 bits, 65536 identified variables can be addressed. The table below shows the identifier allocation space that a station must respect.

ZONE 1	0000 0FFF	Physical allocation 16 variables/station	0 0 x x 0 F x x
ZONE 2	1000 1FFF	Physical allocation 16 variables/station (SM_MPS)	10xx identification 11xx report 13xx remote monitoring 14xx presence 18xx remote control 1 .... 1Fxx remote control 8
ZONE 3	2000 2FFF	Physical allocation 16 variables/station (SM_MPS)	20xx download AE/SE1 21xx upload AE/SE1 .... 2Exx download AE/SE8 2Fxx upload AE/SE8
ZONE 4	3000 7FFF	Free Allocation application layer variables 20 K identifiers	
ZONE 5	8000 8FFF	Physical allocation description variables for zone 1 variables 16 variables/station	0 0 x x 0 F x x
ZONE 6	9000 97FF	Associated variables with universal management identifiers	9000 tag name 9001 physical address 9002 liste of présents 9003 synchro_BA 9004 absolute time 9005 compl. time 10µs 9006 compl. time 100µs 9007 reset counters
	9800 9FFF	Associated variables with universal user identifiers	9010 ID_RET (Acerli) 9011 ID_COM (Acerli) 9012 ID_STAT (Acerli)
ZONE 7	A000 AFFF	Description variables for additinal objects Free allocation	
ZONE 8	B000 FFFF	Description variables zone 4 variables Free allocation	

This allocation space is divided into eight separate zones. Each zone contains identifier numbers that play a special role.

Zone 1

This zone contains the identifiers of physically allocated variables. Each station has 16 physically allocated variables that can be produced or consumed. When a station learns its physical address it creates these identifiers using the address. The physical address is represented by XX and the variable number by the shaded area. If the station number is 45 hexa, the station will create the identifiers 0045, 0145, 0245, 0345, ..., 0D45, 0E45, 0F45. This memory contains 4096 identifiers.

Zone 5

This zone is the description zone for zone 1. For each identifier this zone contains an associated description variable. The identifier number of the description variable is obtained by setting bit 15 of the zone 1 variable identifier to "1".

Zones 2 and 3

These zones contain the network management identifiers. When the station learns its physical address it creates the identifiers associated with the SM_MPS services supported. For example, if station number 45 hexa supports all SM_MPS services, manages 3 AE/SEs and can be piloted by 3 managers, it creates the following variables in its SMAE entity:

10**45**	Identification variable produced
11**45**	Report variable produced
13**45**	Remote monitoring variable produced
14**45**	Presence variable produced
18**45**	Man1 remote monitoring variable consumed
19**45**	Man2 remote monitoring variable consumed
1A**45**	Man3 remote monitoring variable consumed
20**45**	AE/SE1 remote monitor variable consumed
21**45**	AE/SE1 remote read variable produced
22**45**	AE/SE2 remote download variable consumed
23**45**	AE/SE2 remote read variable produced
24**45**	AE/SE3 remote download variable consumed
25**45**	AE/SE3 remote read variable produced

The SM_MPS agent dialogues with a manager by consuming and producing these variables. These two zones contain 8192 identifiers.

Zone 4

Zone 4 is a free allocation zone. APs can choose identifiers in this zone to represent application variables that are a part of the distributed application. This zone contains 20480 identifiers. Description variables for zone 4 variables are found in zone 8.

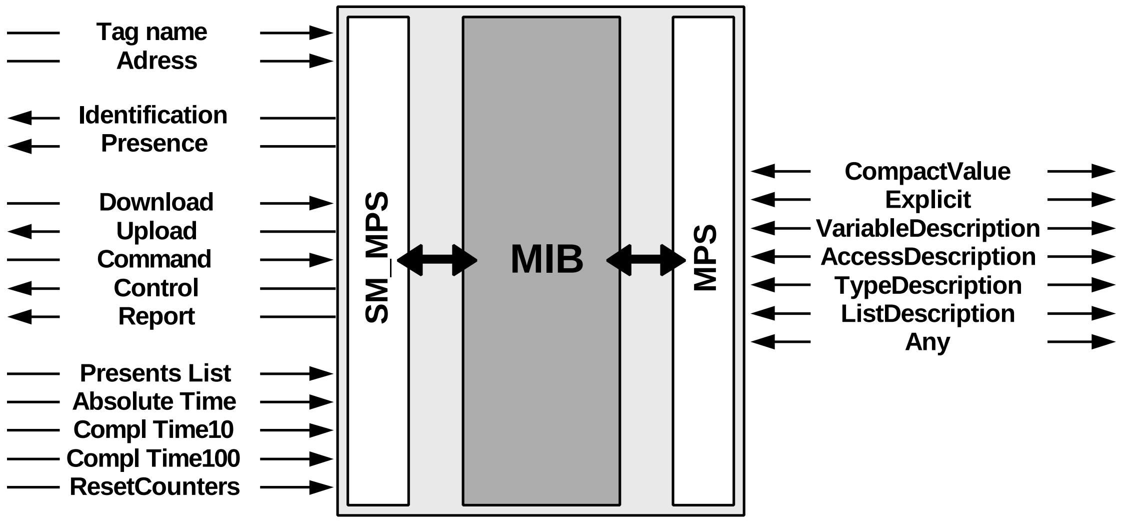
Zone 6

This zone contains universally allocated identifiers. These identifier numbers set aside by the standard are used to code variables of general interest such as the identifier for allocating a tagName, the identifier for allocating a physical address, the identifier for the list of stations present, etc.

5.5 SM_MPS services and PDUs

SM_MPS services can be called upon locally or remotely. When they are solicited by a remote manager, this remote manager generates PDUs containing the command to execute, and any necessary data.

The figure below shows the various types of PDUs generated by SM_MPS and MPS.



PDUs are coded using the same transfer syntax as that used by MPS.

5.5.1 Tagging and addressing PDUs

These PDUs are used to give a station an application tag and a physical address. The station will take this information into account only if it does not already have such information. When the station receives its number it creates physically allocated variables and then starts up its SMAE entity. These PDUs are produced by the manager and consumed by the agents.

5.5.2 Remote download PDUs

If a station supports remote downloading the manager constructs and sends a remote download PDU for each AE/SE. This PDU contains a set of identifiers.

A station that is completely pre-configured knows in advance the number of AE/SEs managed, the number of variables in each of these AE/SEs, the number of each variable in the AE/SEs and their mode (produced or consumed).

The manager constructs the remote download PDU in accordance with this pre-configuration. Identifiers are attributed in the order stated by the agent. The remote download PDU should contain exactly the same number of identifiers as the AE/SE.

These PDUs are produced by the manager and consumed by the agents.

5.5.3 Remote read PDU

Once the agent has received a remote download PDU, it creates identifiers in accordance with the values received. To create identifiers the station makes a connection between its layer 7 variables and these identifiers. To confirm the service the agent recopies the remote download PDU received in the associated remote read PDU.

When the manager receives a remote read PDU it checks to see if the remote download PDU transmitted and the remote read PDU received are identical.

A remote download PDU can contain some 60 identifiers.
These PDUs are produced by the agents and consumed by the manager.

5.5.4 Presence PDU

All stations transmit a presence variable. This 3-byte variable contains summarized information on the station's global operating state. Presence variables are systematically consumed by the active bus arbitrator, which compiles a list of stations present. The agent updates this PDU according to its configuration status.
These PDUs are produced by the agents.

5.5.5 Identification PDU

The identification variable contains a detailed description of the communications entity. This variable contains the following information:

- manufacturer name
- model
- tagName value
- SM_MPS functions supported
- SMS functions supported
- conformance classes layer-by-layer

The information contained in this variable is pre-set by the manufacturer.

5.5.6 Remote control PDUs

A manager can send the following commands to an agent:

- Start AE/SE
- Stop AE/SE
- Reset AE/SE
- Validate AE/SE
- Invalidate AE/SE
- Reset counters

When the manager sends the command it also states the scope of the command in an eight-bit word. Each bit corresponds to an AE/SE.

Remote control PDUs are produced by managers and consumed by agents.

5.5.7 Remote monitoring PDUs

Agents transmit remote monitoring PDUs in response to remote control PDUs. These PDUs are coded using eight bytes. Each byte represents the status of an AE/SE. The following AE/SE statuses are possible:

- AE/SE not supported
- AE/SE non-existent
- AE/SE ready without identifiers
- AE/SE ready with identifiers
- AE/SE in service

5.5.8 Report PDUs

This variable contains the values of error and performance level counters. These counters are incremented by the physical, data link and application layers. The network management standard defines 41 counters.

5.6 SMS

SM_MPS services are relatively simple to implement. They are particularly well adapted to stations that have a static and totally pre-configured configuration. These services cannot be used to create or destroy a new AE/SE or a new variable, nor can they be used to change production or consumption periods, etc.

SMS services provide many more possibilities. SMS services use the data link layer message services via MCS interface services. SMS services are used to:

- create objects (application entities, bus arbitrator entities, variables, elementary cycle...)
- destroy objects
- download configurations (application layer, data link layer, bus arbitrating tables...)
- upload configurations
- read attribute values in the MIB
- write MIB attribute values
- remotely control the station

SMS services can be carried out in an associated or non-associated mode.

6. SECURITY

From the outset of the WorldFIP project large-scale users have stressed the importance of addressing security problems in the network's standards. This section does not pretend to list all the security mechanisms implemented in the WorldFIP protocol or in WorldFIP components. Rather, it summarizes certain functionalities present in the various layers of the protocol.

6.1 Medium redundancy

WorldFIP supports medium redundancy. Network management provides a set of services to manage this redundancy. When the medium is redundant there is one line tool per track managed.

When the station produces a frame, this frame is simultaneously transmitted on both tracks.

Each receiver receives the frame on both tracks. A local mechanism activates the track with the first Carrier Detect. Network management can force stations to listen or transmit on one track for reasons of maintenance or when too many errors have been detected on one of the tracks.

The physical layer uses indications to inform network management of switches in the medium listened to. Network management manages a set of error and performance counters for each track.

6.2 Errors in the physical layer

The physical layer detects signal errors and informs network management, which manages hyper-current and hypo-current counters.

The physical layer also includes a jabber detection mechanism. This mechanism comes into play when one station tries to monopolize the network. With each bit transmitted the physical layer increments a counter. When this counter exceeds an authorized value the physical layer interrupts the data link layer. The jabber error is passed on to network management.

6.3 Layer 2 status machines

Data link layer status machines include security mechanisms that prevent the association of a response frame and the previously received question frame if they do not correspond.

When a data link layer detects an ID_DAT frame it sets an internal timer. If this timer expires the station ignores the following frame unless it is an ID_DAT frame.

Thus if the network is properly configured a response frame that does not correspond to the identifier called for cannot be associated with that identifier.

6.4 Frame Check Sequence

A 16-bit word is associated with each frame transmitted. This word is the result of a polynomial calculation on the useful information exchanged. This FCS is calculated when the frame is transmitted and when it is received. If the code received equals the code calculated there is a very high probability that the frame is correct.

The generating polynomial mathematically guarantees that on a 1 Mb/s network functioning 24 hours a day there will not be more than one incorrect frame detected as correct in 20 years.

6.5 Bus arbitrator redundancy

Network management specifies bus arbitrator redundancy. On a WorldFIP network there may be one or more bus arbitrators. Only one bus arbitrator is active at a given instant; the others are dormant but monitor the activity of the active bus arbitrator.

When the active bus arbitrator breaks down a local mechanism present in all potential bus arbitrators is used to elect a new arbitrator. This election takes place without consultation.

Each WorldFIP station has a physical station number between 0 and 255. The mechanism for electing a new bus arbitrator is a function of this number and of time. Whenever a potential bus arbitrator detects silence on the network, it sets the timer T3. When this timer expires the station elects itself bus arbitrator if no other activity has been detected on the bus since the timer was set.

The formula for calculating T3 is: 4 x (n+ 1)*TO

where: n = station number
T0 = basic time-filler (110µs by default)

For n = 1 and T0 = 110µs, T3 = 660µs (the arbitrator of station number 0 has broken down)

The new bus arbitrator starts up 660µs after having detected silence on the network.

It is clear that bus arbitrators for the network must be given a low station number.

Once the bus arbitrator has been elected it begins scanning. To prevent a possible catastrophe the new arbitrator must have been correctly configured with the same elementary cycles and macrocycles as the former arbitrator.

A new bus arbitrator can take control of the network in several different ways. At the beginning of each elementary cycle the active bus arbitrator updates and then scans a bus arbitrator synchronization variable. This variable is made up of the data pair: < elementary cycle number, macrocycle number >.

Dormant bus arbitrators consume this variable and follow changes in the numbers of macrocycles and elementary cycles. When a bus arbitrator elects itself it can:

- restart the current macrocycle at its beginning
- restart the current elementary cycle in the current macrocycle
- set a timer with the remaining time of the interrupted elementary cycle, and when the timer expires begin the following elementary cycle (to guarantee the temporal synchronization of variable scanning).

Network management also defines:

- initial election when the network is started up
- a priority mechanism for the bus arbitrators. When a BA with higher priority arrives the active bus arbitrator is interrupted.

VALIDITY OF VARIABLES

A value consumed by a station can be valid or invalid. Network management provides mechanisms for the validation/invalidation of data. When a variable is invalid on the data link level the station does not respond when the bus arbitrator scans the variable. It is thus possible to use redundant devices that perform the same task, and if the device in service breaks down switch over to the back-up equipment (which is configured in the same manner) simply by validating the latter's variables through a remote network management command.

Validation of variables also makes it possible to verify the coherence of the configuration of a set of stations before putting them on the network.

When a variable is valid a consumer can access its value. However this value may never have been updated by its owner. In this case the variable is said to be insignificant. The variable becomes significant when the producer gives it a value. A consumer is informed when it reads an insignificant variable.