For a connected world

MarathonTP©

\\Protocol reference V1.1

INERTIA SYSTEMES

\\Copyrights

Inertia Systemes
73 rue des Colombes
Ange-Gardien, Québec
J0E 1E0
http://inertiasystemes.com
http://marathontp.info

\\License

Attribution-NoDerivatives 4.0 International

http://creativecommons.org/licenses/by-nd/4.0/deed.en

\\Revision process

The MarathonTP protocol specification is not static. It is expected to evolve over time. To standardize the development of the protocol it is the responsibility of Inertia Systemes to approve any changes. Any user of the protocol may propose a revision to Inertia Systemes by means of communications available in the section «Copyrights». Upon acceptance of the revision, Inertia Systemes is committed to providing the community an updated version of this document.

\\Warnings

This reference manual is for informational purposes only and is provided "as is". The information contained in this document are presented as a guide and not as a step by step process. We strongly recommend that you engage additional expertise to further assess the requirements for your specific environment.

INERTIA SYSTEMES RESERVES THE RIGHT TO MAKE CHANGES TO THE SPECIFICATIONS OF THE PROTOCOL AT ANY TIME WITHOUT NOTICE.

USERS MUST ASSUME FULL RESPONSIBILITY FOR THE APPLICATION OF THE RULES MENTIONED HEREIN. INERTIA SYSTEMES ASSUMES NO RESPONSIBILITY, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, RELATING TO THE APPLICATION OF THE RULES HEREIN.

Obtaining this document gives you no right to license for patents, trademarks, copyrights or other intellectual property.

\\Contents

\\List of figures

\\List of tables

1 \\Introduction

MarathonTP is a machine to machine (M2M) communication protocol. It is designed to be simple to implement and can be adapted to a multitude of platform capable of handling strings. It enables a reliable message exchange between devices with limited resources. These devices often use 8-32 bit cheap microcontrollers.

MarathonTP is distinguished by the fact that it says "human readable". In effect, the transmitted messages are actually strings of UTF-8 format. Thus, during its implementation, it is easy to confirm the structure of the transmitted messages with network analysis tools such as Wireshark.

According to the OSI model (Open Source Interconnection), MarathonTP is a layer 7 protocol, either the application layer. From the point of view of the OSI model, the application layer is responsible for the representation of data to the user, their coding and control of the dialogue between the systems. MarathonTP sets a standardized message structure as well as a methodology for reliable transfer. MarathonTP can adapt to a wide range of transport protocol. Examples include RS-232, UDP/IP, etc. This document does not cover the establishment of connections between the machines.

The name of the protocol has been chosen to describe the goals during its development. The marathon represents a test of endurance and "TP" stands for "Transport Protocol". MarathonTP is a reliable and sustainable information transport protocol.

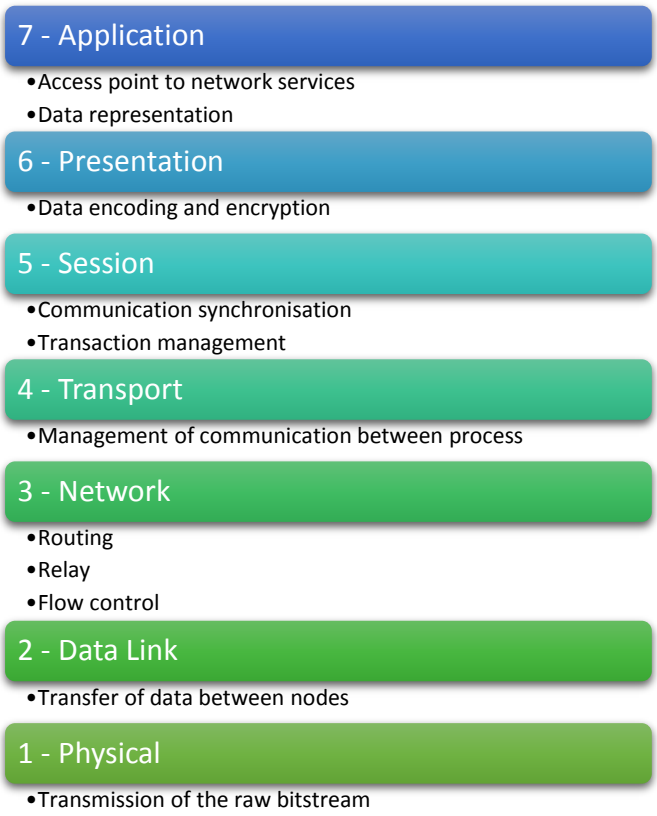

Figure 1 : OSI Model

2 \\Data types

The Protocol provides for the exchange of information in several types. To each of these types corresponds a unique identifier. This allows the interpreter of the message to convert the string received in the correct format. The following table provides a summary of the types that the protocol supports :

Data types	Description	Identifier	Allowed values
BOOLEAN	Contains values that can be only true or false. The True and False keywords correspond to the two states of Boolean variables.	"Bo"	"True" "False"
INTEGER	Contains 32 bits (4 byte) signed integers.	"In"	All integer values between : "-2147483648" and "2147483647"
SHORT	Contains 16 bits (2 byte) signed integers.	"Sh"	All integer values between : "-32768" and "32767".
USHORT	Contains 16 bits (2 byte) unsigned integers.	"USh"	All integer values between : "0" and "65535"
LONG	Contains 64 bits (8 byte) signed integers.	"Lo"	All integer values between : "-922337236854775808" and "922337236854775807"
SINGLE	Contains IEEE 32 bits (4 byte) single precision floating point numbers. Single precision numbers store an approximation of a real number.	"Si"	For negative numbers : from "- 3,4028235E+38" to "- 1,401298E-45" For positive numbers : from "1,401298E-45" to "3,4028235E+38"
DOUBLE	Contains IEEE 64 bits (8 byte) single precision floating point numbers. Single precision numbers store an approximation of a real number.	"Do"	For negative numbers : from "-1.79769313486231570E+308" to "-4.94065645841246544E-324" For positive numbers : from "4.94065645841246544E-324" to "1.79769313486231570E+308".
BYTE	Contains 8 bits (1 byte) unsigned integers.	"By"	All integer values between : "0" and "255".
STRING	Represents a Unicode text character string.	"St"	All UTF-8 characters except reserved characters "{", "}" and ":".
NUL	Represents a zero value. This type is used in case of error.	"Nil"	The string " Nil ".

Table 1 : Protocol data types

2.1 Numbers representation

MarathonTP supports the numbers under the fixed-point notation or scientific. On the other hand, the Protocol sets no specific rule on the representation of one or other of the notations. The following examples are all valid representations:

Examples:

0.000135569887426

1.35569887426E-05

220000000000000000

2.2E17

2.2 Notes on text strings

MarathonTP uses three specific UTF - 8 characters in its implementation. The use of these characters in a text string will cause exceptions at the level of message interpreter. If these characters must be transferred from one application to the other a mechanism must be devised by the developer.

UTF-8 character	Function
{ (Hex 7B)	Message beginning marker.
} (Hex 7D)	Message ending marker.
: (Hex 3A)	Field separator.

Table 2 : Reserved characters

3 \\Message structure

The MarathonTP protocol defined a specific message model which allows compatible devices to exchange information between them.

A MarathonTP message, which is also called 'package', consists of four basic elements.

1. The beginning and ending of message markers
2. The field separators
3. The descriptor
4. The payload

Figure 2 : MarathoTP packet format

3.1 Elements definition

3.1.1 Beginning and ending markers

MarathonTP messages start with the UTF-8 character "**{**", Hex 7B and ends with the UTF-8 character "**}**" Hex 7D.

It's two characters reserved by the protocol that cannot be part of the message itself. (See paragraph 2.2 for exception handling)

3.1.2 Field separators

The field separators delimit the different elements of a package. They are present only in the descriptor and the payload. They are represented by the UTF-8 character "**:**", Hex 3A.

It's a characters reserved by the protocol that cannot be part of the message itself. (See paragraph 2.2 for exception handling)

3.1.3 The descriptor

The descriptor is used to identify the nature of the message. It contains 4 mandatory descriptive elements. The following table shows the enumeration of these elements:

Field	Description	Condition	Allowed values
VER	Protocol version.	Mandatory	"1.0" or "1.1"
RA	Message type "request" or "answer"	Mandatory	"R"or "A"
TNS	Transaction number.	Mandatory	"0-65535"
CMD	Message command.	Mandatory	"0-255"

Table 3 : Descriptor fields

VER : The protocol version allows the evolution of it. Thus, a particular system could answer one or other of the available versions according to used features. This field provides the interoperability of systems.

RA : The message type. For each command there are two types of message. The request ”R” and the answer ”A”.

TNS : The transaction notion allows the traceability of a message between two systems. All active message must have a unique transaction number. As the number is limited to 65536, it is possible to resume numbers already used. As long that the uniqueness rule is respected.

CMD : There are three types of MarathonTP message. These are detailed in point 4.

3.1.4 The payload

The payload is the main part of the message. It is at this level that is found all data exchange between the machines. The payload is in the form of a list of object. The format of these objects varies according to the nature of the exchanged message. There are three object types: single, double and triple containing respectively 1, 2 and 3 elements. The following figure illustrates an example of a list of triple objects:

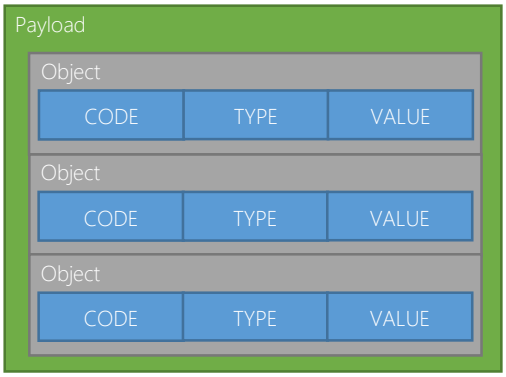

Figure 3 : List of triple objects

4 \\MarathonTP commands

MarathonTP follows a request/response transaction model. This model contains two types of command. Each of these commands set a specific task to be performed by the system which receives it.

As seen in section 3.1.3 the message command is the 4th parameter of the descriptor. This is a required element. The protocol is intended to define up to 255 different commands. At the present time, MarathonTP defined two separate commands. The structure of message associated with each of these commands follows a specific format. The compliance of a MarathonTP implementation depends largely on the respect of this formatting.

As the Protocol is a request/response, there are two query formats and two response formats. For a total of 4 different formats.

4.1 Variable reading

It is the command "1". Reading variable is used to get the value of one or more items in the exchange list of the target system. See point 8 for more detail about exchange list. The maximum number of element that can be read with the same query is limited to 10. Combining multiple items in a single query allows, in many scenario, a maximization of the bandwidth of the underlying network.

4.1.1 Request format

The following table lists the various elements of the request.

Elements	Description	Allowed values
VER	Protocol version.	"1.0" or "1.1"
RA	Message type "request" or "answer".	"R"
TNS	Transaction number.	"0-65535"
CMD	Message command.	"1"
...	Enumeration start.	
ELE	Exchange list element.	"0-65535"
...	Enumeration end.	

Table 4 : Elements of a read request

Message example:

1.0	R	25693	1	0	1
VER	RA	TNS	CMD	ELE	ELE

Packet: {1.0:R:25693:2:0:1}

4.1.2 Answer format

The answer enumeration must match the enumeration of the corresponding request. Each element is accompanied by a response code as defined in point 4.4.

For any response code different of 0, it is mandatory to use the element type "NUL" in table 1 with a value of 0.

Elements	Description	Allowed values
VER	Protocol version.	"1.0" or "1.1"
RA	Message type "request" or "answer".	"A"
TNS	Transaction number.	"0-65535"
CMD	Message command.	"1"
...	Enumeration start.	
CODE	Answer code.	See table 10
TYP	Element type	See table 1
VALUE	Element value	See point 2
...	Enumeration end.	

Table 5 : Elements of a read answer

Example 1: Resulting message from point 4.1.1 request. :

1.1	A	25693	1	0	Si	84.83	0	Do	8.936E+10
VER	RA	TNS	CMD	CODE	TYPE	VALUE	CODE	TYPE	VALUE

Packet: {1.1:A:25693:1:0:Si:84.83:0:Do:8.936E+10}

Example 2: Resulting message from point 4.1.1 request, but element “1” is not part of the receiver exchange list. :

1.1	A	25693	1	0	Si	84.83	1	Nil	0
VER	RA	TNS	CMD	CODE	TYPE	VALUE	CODE	TYPE	VALUE

Packet: {1.1:A:25693:1:0:Si:84.83:1:Nil:0}

4.2 Variable writing

It is the command "2". The writing of variable is used to update the value of one or more elements in the exchange list of the target system. See point 8 for more detail about exchange list. The maximum amount of element that can be updated with the same query is limited to 10. Combining multiple items in a single query allows, in many scenario, a maximization of the bandwidth of the underlying network.

4.2.1 Request format

The following table lists the various elements of the request.

Elements	Description	Allowed values
VER	Protocol version.	"1.0" or "1.1"
RA	Message type "request" or "answer".	"R"
TNS	Transaction number.	"0-65535"
CMD	Message command.	"2"
...	Enumeration start.	
ELE	Exchange list element.	"0-65535"
VALUE	Value to be entered in the element.	See point 2
...	Enumeration end.	

Table 6 : Elements of a write request

Message example :

1.1	R	25693	2	0	25.6	1	8.15698563
VER	RA	TNS	CMD	ELE	VALUE	ELE	VALUE

Packet : {1.1:R:25693:2:0:25.6:1:8.15698563}

4.2.2 Answer format

A variable writing returns no value. Simply an answer code. The order in which are sent the answer codes is the same as that of the corresponding request message.

Elements	Description	Allowed values
VER	Protocol version.	"1.0" or "1.1"
RA	Message type "request" or "answer".	"A"
TNS	Transaction number.	"0-65535"
CMD	Message command.	"2"
...	Enumeration start.	
CODE	Answer code.	See table 10
...	Enumeration end.	

Table 7 : Elements of a write answer

Example of resulting message from point 4.2.1, but element "1" is not part of the receiver exchange list.

1.1	A	25693	2	0	1
VER	RA	TNS	CMD	CODE	CODE

Packet : {1.1:A:25693:2:0:1}

4.3 The Discovery

It is the command "3". The Discovery is used to verify the existence of a compatible MarathonTP system to a defined IP address. The Discovery is a special request that does not attach to the standard send/receive process as defined in point 5. Thus, the transaction number is defined as compatibility with other commands. The Discovery can be especially effective in network broadcast mode. Thus, a single request should generate as much response as there are MarathonTP systems on the network. The Discovery uses the index 2 and 3 of the exchange list that returns the IS ID of the system as well as the security mode of the latter. See section 7 for more information on security codes. The correlation of the response with the Inertia Systemes open database is left to the system using it.

Note :

The discovery is always used without any security level. Therefore, in broadcast mode, the protocol sets to 5 seconds the minimum timeout before sending a new request, limiting the network load.

4.3.1 Request format

The following table lists the various elements of the Discovery request. The element "2" should always precede the element "3".

Elements	Description	Allowed values
VER	Protocol version.	"1.1"
RA	Message type "request" or "answer".	"R"
TNS	Transaction number.	"0-65535"
CMD	Message command.	"3"
...	Enumeration start.	
ELE	Exchange list element.	"2" or "3"
...	Enumeration end.	

Table 8 : Elements of a Discovery request

Message example:

1.1	R	25693	3	2	3
VER	RA	TNS	CMD	ELE	ELE

Packet: {1.1:R:25693:3:2:3}

4.3.2 Answer format

The Discovery answer enumeration must match the enumeration of the corresponding request. Either the element "2" follow-up of the item "3" from the exchange list. Each element is accompanied by a response code as defined in point 4.4.

For any response code different of 0, it is mandatory to use the element type "NUL" in table 1 with a value of 0.

Elements	Description	Allowed values
VER	Protocol version.	"1.1"
RA	Message type "request" or "answer".	"A"
TNS	Transaction number.	"0-65535"
CMD	Message command.	"3"
...	Enumeration start.	
CODE	Answer code.	See table 10
TYP	Element type	See table 1
VALUE	Element value	See point 2
...	Enumeration end.	

Table 9 : Elements of a Discovery answer

Example of resulting message from point 4.3.1:

1.1	A	25693	3	0	St	76be3439-414b-4646-808d-af457aa6ddd6	0	By	0
VER	RA	TNS	CMD	CODE	TYP	VALUE	CODE	TYP	VALUE

Packet: {1.1:A:25693:3:0:St:76be3439-414b-4646-808d-af457aa6ddd6:0:By:0}

4.4 Answer codes

All MarathonTP answers are accompanied by code defining the state of the transaction. There are circumstances where MarathonTP requests will cause errors in systems that will run them. The protocol provides four types of error defined by a numeric code, shown in the following table. :

Error code	Description
0	Operation successfully completed.
1	Element not found
2	Incompatible data type. This error occurs on variable writing.
3	Index out of range of exchange list limits.

Table 10 : MarathonTP error codes

5 \\Message flowchart

MarathonTP messages are request/answer. This methodology implies that sending a message to a receiver must end with an answer received by the sender. All within a time limit defined by the congestion control. (See item 6) If these conditions are not met then the message is considered lost and must be sent again if necessary.

Note : The Discovery is a particular type of message which does not require answer. A Discovery request and answer must be treated as unique events associated with a dummy transaction.

5.1 Message emission

The following flowchart illustrates the sequence of operation for the MarathonTP request message transmission.

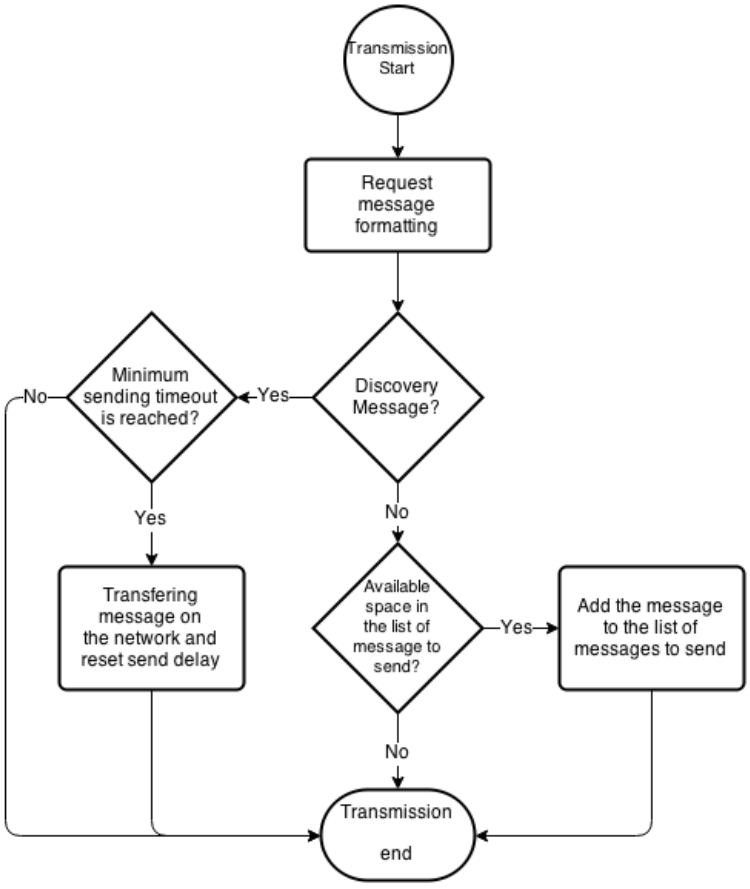

Figure 4 : Emission flowchart

5.2 Message receiving

The following flowchart illustrates the sequence of operation for the MarathonTP message receiving.

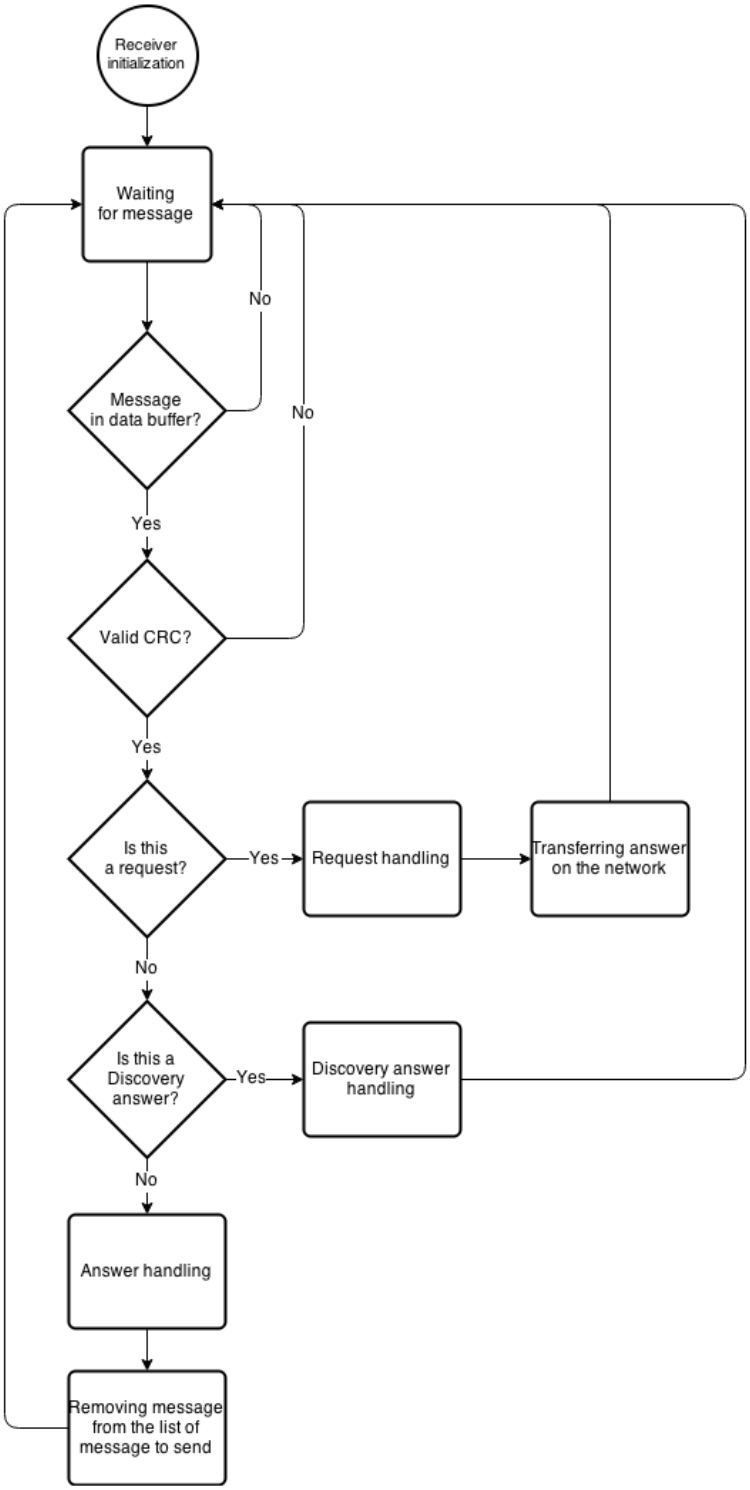

Figure 5 : Receiving flowchart

5.3 Message recycling

The following flowchart illustrates the sequence of operation for the MarathonTP message recycling.

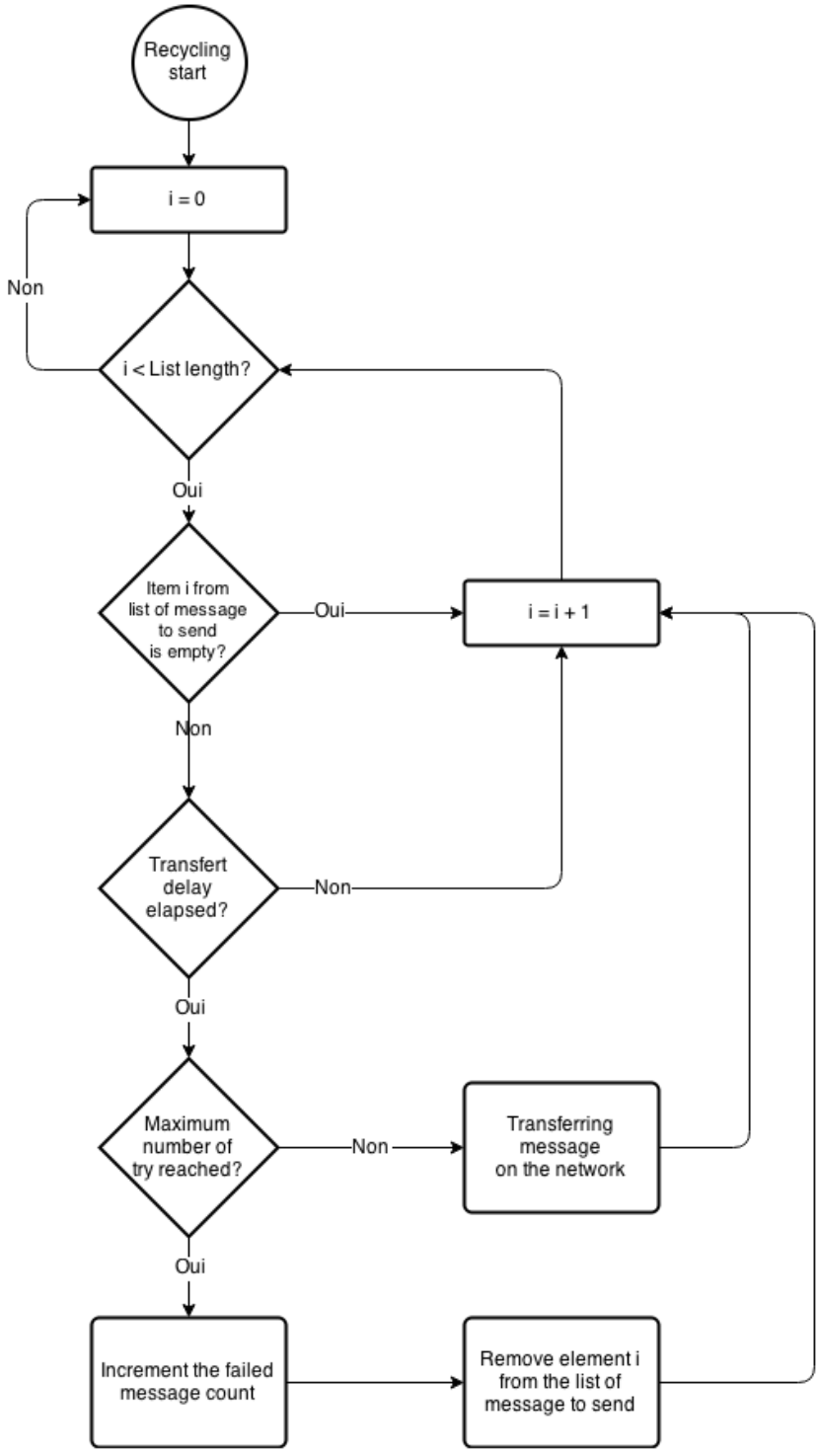

Figure 6 : Recycling flowchart

6 \\Congestion control

The burden of congestion control is mainly based on MarathonTP servers. For their part, clients always transmit responses according to a logic of the best effort. In other words, as soon as a request is processed the answer is immediately sent to the server. In situations where the network load is high, it is imperative to introduce messages emissions control. The basic mechanisms used for this purpose by MarathonTP are the exponential back-off and the sending controller. Each aimed to limit messages sending to avoid network overload.

6.1 Sending controller

In MarathonTP, the sending controller maximizes the capabilities of the transfer and processing of MarathonTP devices. Indeed, in an exchange between a client and a server, some of the waiting period between the emission of the message and the receipt of the response is induced by the message processing. A properly configured sending controller could send messages to a server with a greater rate than the average response time

The protocol provides a list of message to send. It is a stack containing all messages waiting for transfer to a specific recipient. There are as many message stack that active recipient within the application. This type of attribution allows to evaluate the congestion control scenarios tailored to each recipient. This type of analysis prevents the deterioration of sending capabilities to devices whose response times are good. If the list is full, no new messages can be transmitted over the network. The size of the list is a factor calculated by the sending controller.

Note :

At present, the protocol specification limits the list of message to send to a single element by recipient. A definition of the sending controller is expected in a future specification

6.2 Exponential back-off

This mechanism increase the "TimeOut" time of a message answer if this time limit has already been reached. The increase is calculated by a multiplication factor of 2 each time the previous time is reached. Two parameters affect the operation of this congestion control. This is the "Max Retransmit Interval" and "Max Retry Attempt". (See paragraph 8 for the definition of these parameters). In all cases, a message may be transmitted again on the network if any of the above parameters is reached. In such a case, the message is removed from the message list of and the "Failed Message Count" variable is incremented.

The protocol define the initial "TimeOut" to 3 seconds. Each of "TimeOut", "Max Retransmit Interval" and "Max Retry Attempt" parameters are modifiable by implementing such functionality. In accordance with RFC 5405, minimum "TimeOut" is set at 1 second.

7 \\Security

Because MarathonTP was designed to provide a means of communication for the internet of things (IoT), it is required to provide a secure information exchange method. By cons, it is essential to consider that most of the IoT devices have low processing capabilities. Thus, the protocol provides three levels of security. The implementation must provide a way to change the security mode of the device using it. The Discovery answer (point 4.3.2) must include information specifying the security mode of the requested device. This way, clients can adjust their communication by using the correct mode. The following table summarizes the security modes that the protocol supports:

Security mode	Allowed values
NONE	« 0 »
XTEA	« 1 »
ADVANCED	« 2 »

Table 11 : Security modes

7.1 No Security

A device in this mode will exchange messages over the network with any other third parties using non-encrypted raw text message. It is the responsibility of the user to understand the implications of using this type of transmission.

7.2 XTEA (eXtended TEA)

XTEA is a secure encryption algorithm that uses a 128-bit key and requires little CPU power. It is an ideal choice in the context of the Internet of things (IoT). It is important to note that this encryption is not as secure as more complex implementations such as RSA for example. Refer to the following website for XTEA implementation examples: http://marathontp.info

7.3 Advanced security

No advanced encryption technology has yet been set for the protocol. This feature will be presented in a future version.

8 \\Exchange list

The data access in MatahonTP are made through index. These indexes are "UShort" data type. There are therefore 65536 distinct data that can be transferred between two MarathonTP systems including 65436 usable by third parties.

A certain amount of these data are reserved by the protocol itself. The following table shows the index ranges distribution.

Index	Element	Type	Description
0	Ping	Boolean	MarathonTP ping answer.
1	Device Serial	String	Device serial number.
2	Device IS Identifier	String	Inertia Systemes unique identifier for the device.
3 to 9			MarathonTP reserved.
10	Sended Count	Integer	Sended message count.
11	Received Count	Integer	Received message count.
12	Failed Count	Integer	Failed message count.
13	Retried Count	Integer	Retried message count.
14	Successful Per Second	UShort	Successfull message per second.
15 to 99			MarathonTP reserved.
100 to 65535			Usable by third parties.

Table 12 : Index range distribution.

It's the role of the designer of the device to provide a document specifying the function of the 65436 indexes available for its needs. It's this document which constitutes the exchange list. The exchange list should be accessible to any user of the device.

8.1 Precision on reserved indexes

The following points describe more accurately the nature and function of the protocol reserved indexes.

8.1.1 Index 0 : Ping

"Ping" simply returns to the client a response of boolean data type whose value is true. A true response by the client indicates the existence of an active server for the queried IP address.

8.1.2 Index 1 : Device Serial

The unique serial number of the device, presented in textual form. The serial number is specific to the manufacturer of the device.

8.1.3 Index 2 : Device IS Identifier

Unique device identifier provided by Inertia Systemes during the approval of MarathonTP compliance. This textual identifier allows services and softwares to consult the Inertia Systemes device database to obtain the characteristics of the product.

8.1.4 Index 3 : Security Mode

Number between 0 and 2 inclusive that represents the security mode of the queried device.

8.1.5 Index 10 : Sended Count

Incremental number representing the amount of message sent by the device. This parameter is useful for purposes of analysis and statistics. The device must manage the overflow of an "Integer" with return to 0.

8.1.6 Index 11 : Received Count

Incremental number representing the amount of message received by the device. This parameter is useful for purposes of analysis and statistics. The device must manage the overflow of an "Integer" with return to 0.

8.1.7 Index 12 : Failed Count

Incremental number representing the amount of message which have never been interpreted by the receiver. This parameter is useful for purposes of analysis and statistics. The device must manage the overflow of an “Integer” with return to 0.

8.1.8 Index 13 : Retried Count

Incremental number representing the amount of message resent by the device. This parameter is useful for purposes of analysis and statistics. The device must manage the overflow of an “Integer” with return to 0.

8.1.9 Index 14 : Successful Per Second

Number representing the amount of message correctly exchanged between two MarathonTP devices every second. This number is updated every seconds. This parameter is useful for purposes of analysis and statistics.

8.1.10 Index 15 : Max Retransmit Interval

Number in milliseconds representing the absolute maximum time limit for the issuing a message.

8.1.11 Index 16 : Max Retry Attempt

Number of allowed retry.

8.1.12 Index 17 : TimeOut

Number in milliseconds representing the delay allowed between sending a message and receiving an answer.

9 \\Protocol conformance

To ensure the compatibility of the MarathonTP protocol between various technologies and systems, some fundamentals must be observed.

9.1 Communication port

MarathonTP has a reserved UDP/IP communication port. It is the port 8384.

9.2 Data types

At a minimum, all data types in table 1 must be available.

9.3 Packet size

Although MarathonTP limits to 10 the number of element per packet, it remains the responsibility of the developer to respect the constraints of the used transport protocol. Packets must not be splitted.