



## Choosing a Protocol

|                       |                |
|-----------------------|----------------|
| <b>Published date</b> | September 2006 |
| <b>Version</b>        | 1.0            |
| <b>Author</b>         | Paul Perez     |

**Pymma group** 22 Chestnut Avenue, Rickmansworth Herts WD3 4HB United Kingdom

**Contact UK** +44 79 44 36 04 65

**Contact FR** +33 6 99 36 07 10

**Contact NL** + 31 621 273 973

**Email** [contact@pymma.com](mailto:contact@pymma.com)

**Web site** [www.pymma.com](http://www.pymma.com)

Table of content

|  |           |
|--|-----------|
| <b>Introduction.....</b>   | <b>3</b>  |
| <b>Protocol : Simplified Definition.....</b>                               | <b>3</b>  |
| <b>Characteristics of a protocol.....</b>                                  | <b>3</b>  |
| Three questions .....  | 3         |
| Strong bond and weak bond (coupling).....                                  | 4         |
| Synchronous or Asynchronous Communication.....                             | 5         |
| Conservation of the context.....   | 7         |
| <b>Technical implications of the characteristics of the protocols.....</b> | <b>8</b>  |
| Introduction.....  | 8         |
| Strong coupling and weak coupling.....                                     | 9         |
| Synchronous or asynchronous communication.....                             | 11        |
| Characteristic of the conservation of the context.....                     | 15        |
| <b>Conclusion.....</b>   | <b>18</b> |
| The step of the decision maker.....  | 18        |
| <b>References.....</b>   | <b>19</b> |
| About the author.....  | 19        |

## Introduction

This architecture document belongs to a series of which the goal is to familiarise the non-architects with a certain number of concepts of data-processing architectures. These often simple concepts must be well understood before approaching more complex reflections or decision-making. Once assimilated, it will be easier to understand the jargon of technicians, their way of thinking and to join in the decision-making and "to feel" the coherence of a technical solution.

We present with this architecture document, three characteristics of communication protocols. We will not discuss the protocols in technical detail, nor the structures of the messages which they transport, we will remain as much as possible on the level of the concepts. We will talk on how to communicate and not on the information which one exchanges.

## Protocol : Simplified Definition

It is never easy to define something in short and simple terms. Let us launch and say that that a protocol is **a convention chosen between two data-processing programs to communicate.**

**Definition of Convention:** *Voluntary agreement of two or several persons or entities (French Academy).*

The interesting point of this definition is that a protocol is an agreement between two or several entities. That is to say that before even communicating it is necessary to act in concert and agree on the way of communicating. Prior agreement between the communicating partners is essential.

## Characteristics of a protocol

### Three questions

When, as an architect, we must choose a communication protocol, it is practical to qualify the type of exchange which will exist between the communicating entities. For that, we are accustomed to putting forward the three following questions:

Does the communication create a strong or weak bond between the applications?

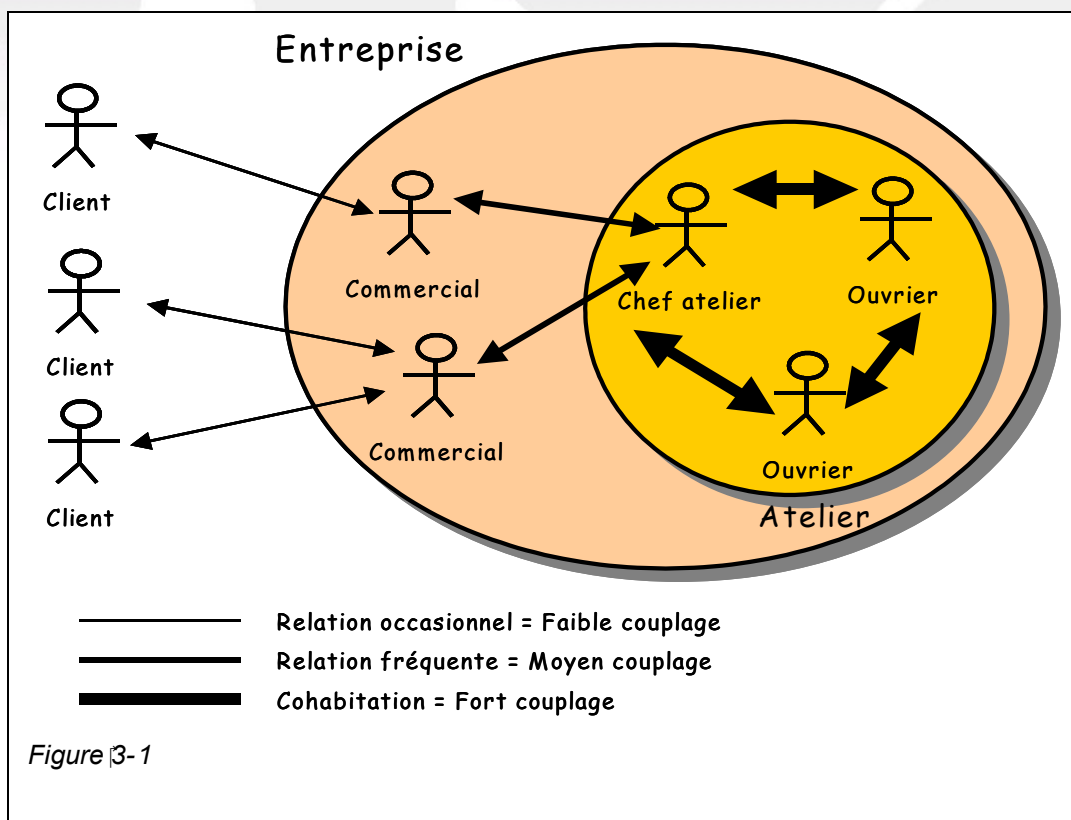
Is the communication between the applications synchronous or asynchronous?

Is there a conservation of a context during the communication between the applications?

According to the answers, we obtain a classification in such or such category. That facilitates the choice and the emergence of a protocol answering the specifications of our architecture. Even if this step is a little "easy trick", it questions correctly on the characteristics of a protocol. Below, we explain in detail the direction of these three questions.

**Strong bond and weak bond (coupling)**

To explain the concept of the strong and weak bond, let us take the example of a company in which we distinguish three various levels: the workshop, the company and customers.



In this diagram, we observe several types of relations.

Inside the workshop, the workmen are in permanent contact. There is a kind of cohabitation in the etymological direction (state of two people who live together). Because the employees often work on the same tasks, the communication must be very fast and the reaction times weak. The bond, the coupling between the entities of the workshop are strong because they depend one another.

Inside the company, the communications between the sales departments and the workshop are frequent but do not have the same intensity, nor the same urgency. If for an unspecified reason the workshop did not function a day, the sales department would still be operational and conversely.

The third type of relation is that between the company and its customers. Here the relation is generally occasional. The coupling is weak. One communicates by mail, telephone or e-mail. If a request is sent by a customer, the answer is often obtained after several hours to several days. There is not the same urgency.

As for the company, data-processing applications also have a variable coupling. It can be more or less strong according to their relation. If as in the workshop, one wishes a strong interaction and a great reactivity between applications, the bond will be known as "strong". If in the contrary case, one wishes to separate the applications, to avoid the interdependences, that they can exist independently from/to each other, then a coupling will be known as "weak". The terms of weak coupling or loose coupling are use alternatively by architects.

### Synchronous or Asynchronous Communication

Before continuing any further, let us recall what one calls a synchronous communication and an asynchronous communication.

#### Synchronous communication

A data-processing application emits a request and receives an answer.

A **synchronous** communication has the principal characteristic of correlating an answer and a request. In other terms, that means that the request and the answer are dependent in the same **unit of time** and **instruction**.

**Unit of time:** acceptable duration during which the transmitter of the request can await an answer.

**Unit of Instruction:** started from a treatment during which the context of the process is unchanged.

To be simple, the time should be known in a synchronous communication, once the request is emitted, the transmitter is blocked whilst waiting for an answer. Thus, the received answer will be associated with the emitted request and will be treated in the same unit of time and instruction. In this blocking, the transmitter freezes the context of the process to receive the response in the same instruction unit as the request.

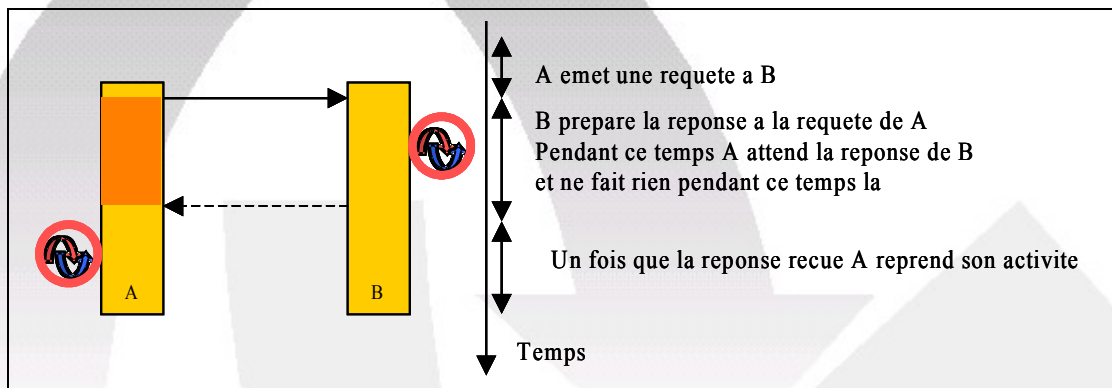
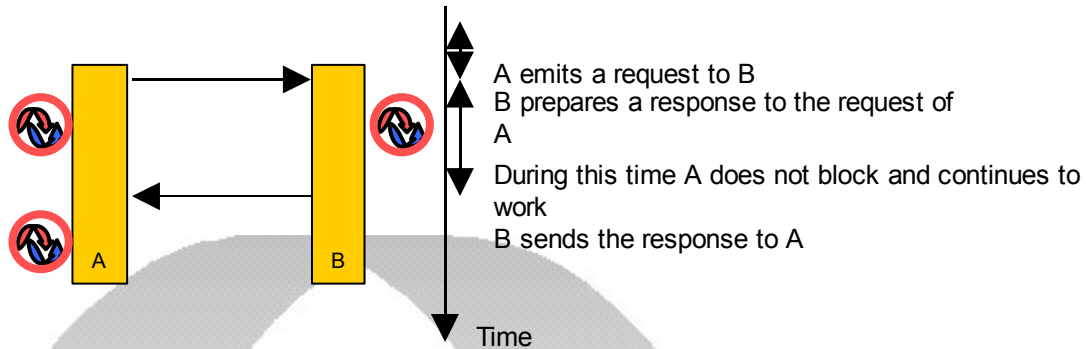


Figure 3-2

**Asynchronous Communication**

An asynchronous communication has the principal characteristic of decorrelating a request and its answer. That is to say that the transmitter sends a request and has no priori requirement at the time when it will receive its answer (Of course, it hopes that it will come as quickly as possible). It does not know if an answer will come, nor when it will come. In military jargon this is called "Fire and Forget".

Consequently, after the sending of the request, the transmitter is not constrained to await an answer. The transmitter is not blocked in waiting of the answer and the context of the process in progress evolves during this waiting period. When it arrives, it must try to associate the response to the request intrinsically, however, nothing does bind request and answer in the protocol directly.



It should be known (“to simplify things”) that certain protocols are at the same time synchronous and asynchronous. We will not discuss the detail of these hybrid protocols here.

**Conservation of the context**

**Definition of a Context :** It is a historical form from what occurred previously between two applications which communicate.

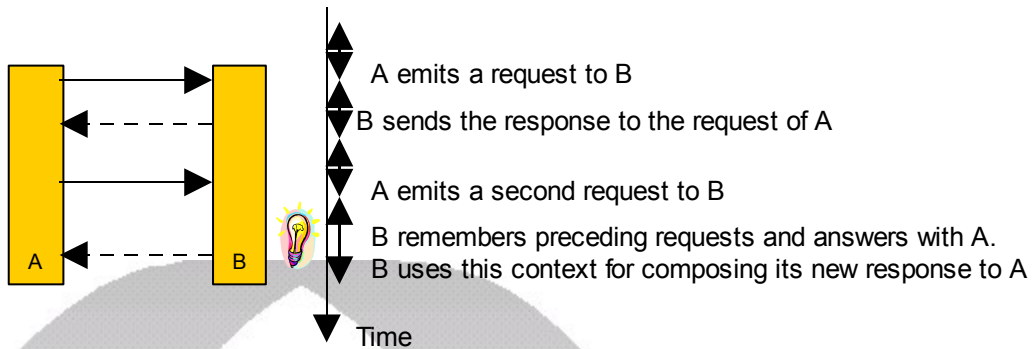
Before continuing any further, it should be known that certain communication protocols make it possible to preserve a “permanent” connection between the data-processing entities (ex: Java RMI). Other protocols create a connection at the time of sending of the request and destroy it after receipt of the answer (ex: HTTP).

When a connection is permanent, we can create at either side a context associated with connection. In this context, it is possible to manage a history of the exchanges between the entities, to create objects or variables dedicated to the treatments of the requests and forward answers by permanent connection.

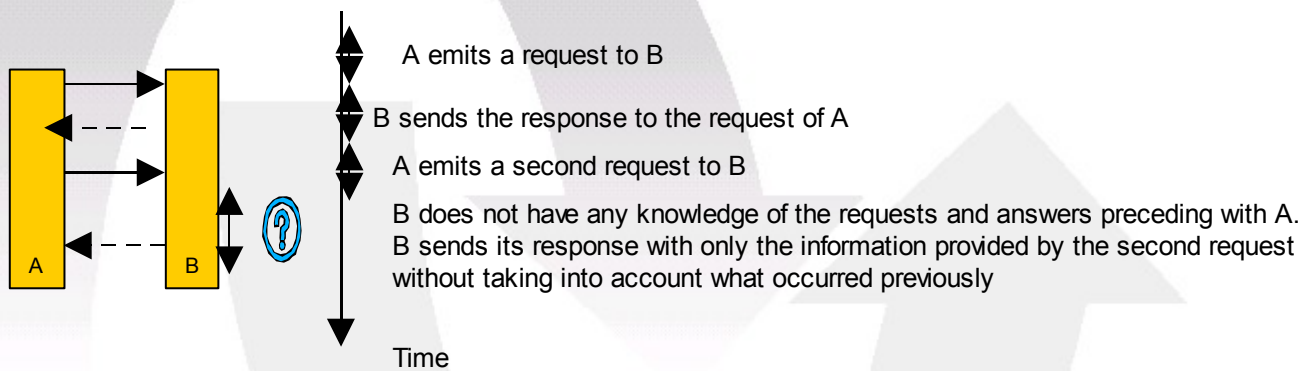
On the opposite side, when a connection is limited in time, it is not possible to create this type of context, instead, after the emission of the first answer, the connection disappears and the context which would be associated for it also disappears.

The first protocols where often called protocols with state or stateful protocols. The second, are named protocols without states or stateless protocols.

Example of protocol with conservation of the context



Example of protocol without conservation of the context



## Technical implications of the characteristics of the protocols

### Introduction

To identify that a protocol is stateless or asynchronous is the first stage. It is not, however, sufficient to decide on a choice a protocol. It still misses a stage. We must initially, answer a question: What are the technical consequences of the characteristics of a protocol? To be more concrete, we put the question: "technically, what is the contribution that a synchronous protocol has compared to an asynchronous protocol, that matters to me that a protocol is stateful or stateless..."

To answer these questions, There are still a certain number of rules to understand and know of. This is the objective of this chapter.

### Strong coupling and weak coupling

The problem arising from the strong and weak coupling is the choice of the effectiveness during the communication and the interdependence between applications. The rule to retain is that there cannot be a great effectiveness between two data-processing applications without interdependence. And the same, there is no independence without loss of effectiveness.

Let us look again at the example of the company of the preceding chapter: In the workshop, the exchanges of information are frequent, many tasks are made jointly and as in all the professional environments, one speaks an occupational jargon so that one may be understood. As in a sporting team, a certain proximity between employees of the workshop brings a great effectiveness in work.

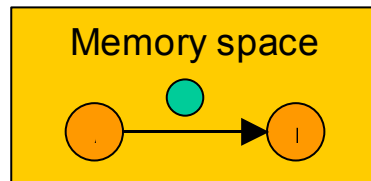
Let us suppose now that for personal reasons one of the employees of the workshop needs to change field, to say, the finance section. Then their professional jargon and automatisms used in their preceding position are no longer required. For an employee beginning a period of adaptation is something which we all meet during a professional change.

Fortunately, the human being is flexible and is characterized by its adaptabilities. It is not the same for data-processing programs. If an application is written specifically for an environment E1, it will be able to adapt to another environment E2 only at the price of an important rewriting. That is to say that with the design, if one decides on a level of coupling between applications, it will be very difficult to change afterwards.

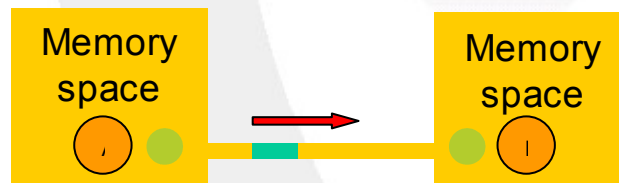
### Level of coupling in practice

Let us try to be more practical and to give some examples of coupling. Let us take as example, the communication in a Java program. Java applications use various techniques to communicate.

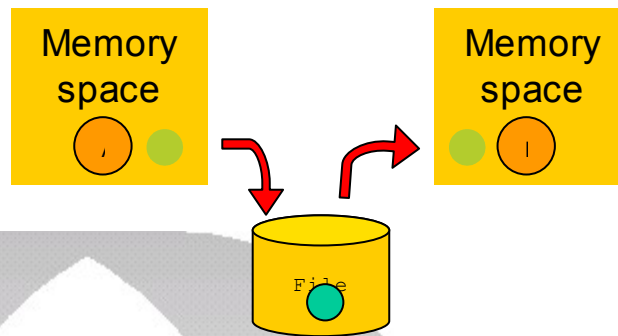
**Communication in the same memory capacity:** Java applications communicate with one and other directly, they are in the same space, the same environment. We find in the situation of the workshop. Application A knows the application B perfectly, they have common objects which they exchange like parameters of requests or answers. These very fast and very effective exchanges remain confined in a memory capacity. (In the same workshop)



**In different memory capacities:** One of the fundamental principles that is necessary to know in the Java world .Net..., is that an object exists only in the memory capacity in which it was created and that as soon as one tries to leave it, like a red fish out of its aquarium, it dies. However, It is often necessary to transfer an object from a memory capacity towards another in order to exchange information between applications which are not localised in the same memory capacity (ex: two different sites, two distinct machines). For that, Java proposes a technique which makes it possible for two memory capacities to communicate. It is a kind of virtual pipe by which information circulates. The disadvantage of the system is that it is necessary to flatten the Java objects to insert them in the pipe and to unfold the ribbon of data to reconstitute a normal object on the other side. In the situation where two workshops of the same factory wish to communicate. It is not very complicated, but it is not as simple as the previous example.



**Another type of coupling:** Now here is another type of coupling a little different from the preceding ones. An application A wishes to obtain a response to its request. Here, A is not concerned who will answer him, but wants to obtain an answer. A formulates its request in the form of a message and records it in a file. Later in time, an application B reads the file written by A, recovers the message, the interpreter, gives an answer and finally writes the response in the file. Later in time, A recovers the file with its answer. At this point in the figure where A does not know B and conversely B does not know A. the answer could have been given by another application C or D. That does not pose a problem. In this figure we have no dependence between A and B but we are far from a maximum effectiveness in the exchange.



### What is necessary to retain of coupling

I could still have enforced the feature a little and given some additional examples to show the obviousness: there is a direct relation between the effectiveness in an exchange and knowledge and the proximity between partners and by consequence the “dependence” between the partners. We will retain from this chapter that there is incompatibility between independence of the partners and the effectiveness of their exchanges.

### Synchronous or asynchronous communication

#### Introduction

Before further discussing the characteristics of synchronous and asynchronous communications, we would like to raise the following paradox:

#### The paradox of synchronous and asynchronous in data processing

At the beginning of IT history, computers were so limited in power that the user was “to reduce” his way of thinking and to act within the capacities of the machine. Fortunately, this period is finished and the rule now is to adapt the data-processing applications to that of “everyday life”. It is by analyzing our daily behavior that one produces good ergonomics and intelligent trades processes.

Here is a question which rejoins our subject. **Is life made of synchronous or asynchronous communications?**

Obviously, 99,99% of our exchanges are asynchronous. You can seek to remember the moments when you remained blocked, suspended not doing anything, awaiting the answer of your partner. I struggled to find a time for which I was suspended on the actions of my partner. Well, the day of moving home, when we transported a heavy piece of furniture and it was necessary to make it pass by the door, with my partners amateur removal skills it was necessary to be synchronized. I was at this time completely dependent on the actions on my partner.

Except in these delicate moments, we do not immobilize ourselves whilst waiting for an answer. If the processing applications were to adapt to our way of living, then at least 90% of the protocols used should be asynchronous but it is not the case. A very great majority of the data-processing applications are synchronous. The way of thinking of the originators and the developers is synchronous. Only some hairy hurluberlus propose to develop asynchronous solutions. As for the decision makers ready to engage for this type of solution, they are rarer than a day of rain in the Atacama desert. We will not try to understand this fact because that would involve us stepping out of the framework of this study. But know that the existing asynchronous applications, even if they are difficult to implement are among the most powerful and most reliable. (Which does nothing but accentuate the paradox).

### **Characteristics of the synchronous and asynchronous communications**

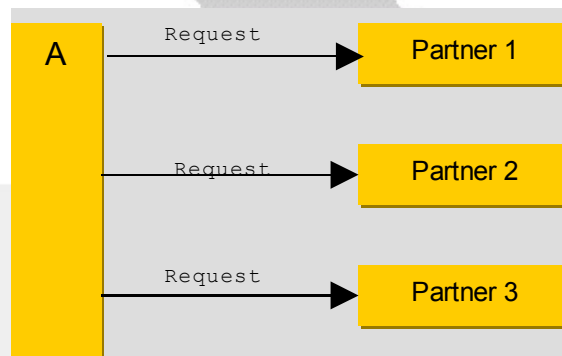
The behavior of the protocol influences the behavior of the data-processing application greatly. The fact of not blocking the requesting application requires additional instructions.

For example:

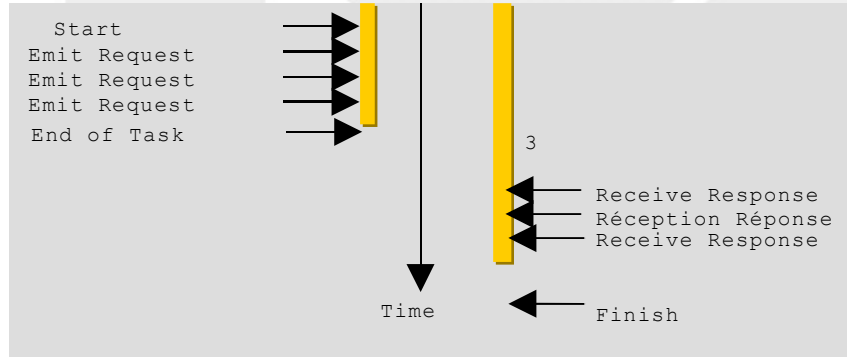
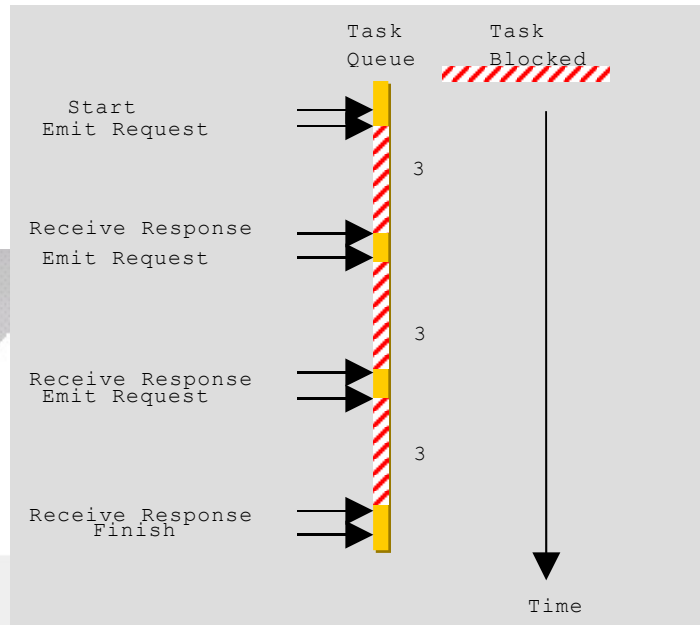
- To carry out a process an application A must emit several request Q1 Q2 Q3. Since A is not blocked in waiting of the answers R1 R2 R3, it is possible to send the 3 requests before the first answer has arrived.
- When an application A receives an answer, it must be able to find which request it is associated with.
- Once the answer is associated with the request, it is necessary to find the context of the request to treat the response in the associated context.

Often the emission and the reception of the request are treated by two tasks carried out in parallel. The management of parallel tasks is delicate and can require some "Know how". The use of an asynchronous protocol makes the communications more effective and powerful between applications, especially in the event of strong activity.

Example: Let us suppose that to carry out a process, it is necessary to emit a request with three different partners and that on average, the answer of a partner is approximately 3 seconds.



Also, let us suppose that the times of emission of request and reception of answer are negligible in respect of times of communication, the minimum time to carry out all the processes will be 9 seconds for a synchronous mode and 3 seconds for an asynchronous mode.



An asynchronous protocol makes it possible in certain cases to reduce the dependence between applications. If a powerful application is dependent on another less powerful application, as for the solidity of a chain, the performance of the complete system depends on the least powerful application. The use of an asynchronous protocol does not oblige the most powerful application to remain blocked in waiting for a response of the least powerful application. This principle is also applicable if the telecommunications network is degraded and does not allow an important flow between the applications.

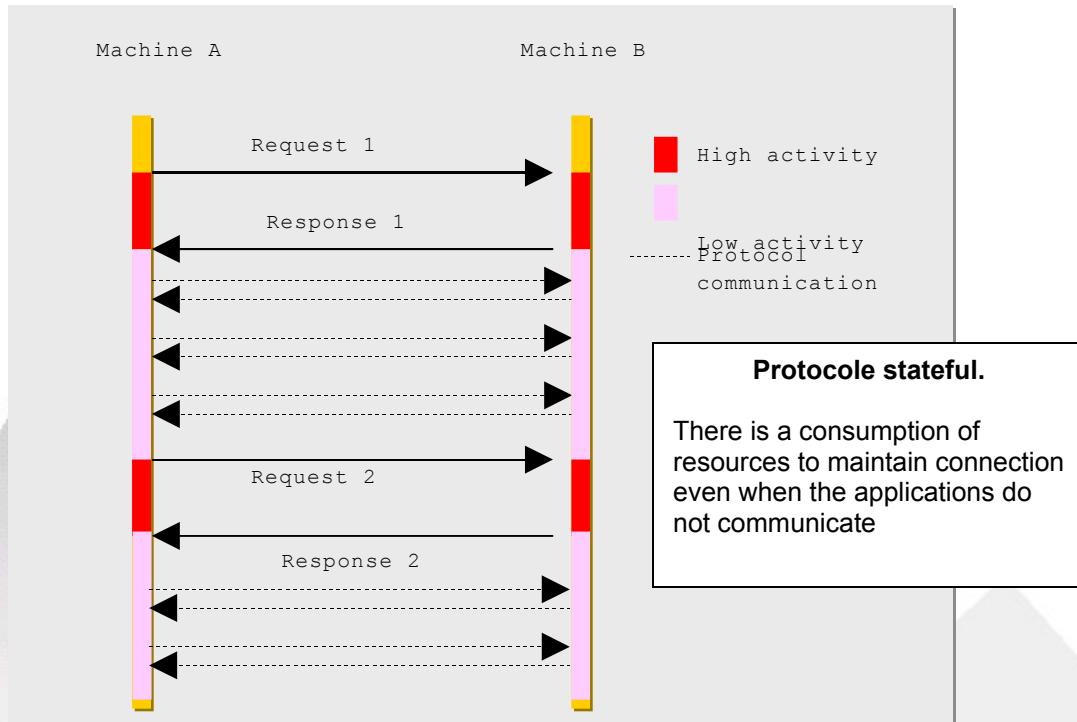
### **What is necessary to retain of the synchronous and asynchronous protocols**

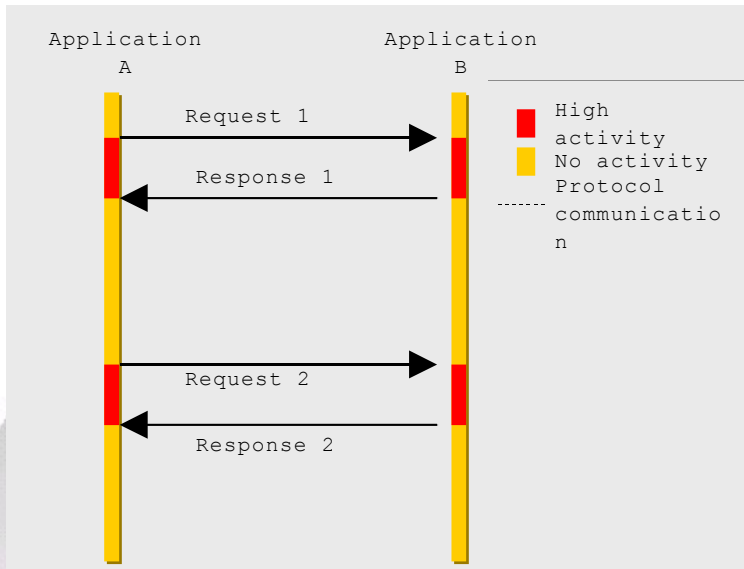
The applications using the synchronous protocols are easier to conceive and realize. The blocking of the application and thus of the context of use allows a linear management of the processes where requests and answers can be queued. The management of the transactions is facilitated by it (especially in the event of transactions between several partners). But this protocol type is not very effective at the time of a strong rise in load. It is in seeking this effectiveness that all the modern operating systems use in-house asynchronous communications.

### **Characteristic of the conservation of the context**

The first rule that should be known is that a stateful protocol is much more a consumer in resource than a stateless protocol. It is a easy rule to understand. Maintenance of a connection requires many resources. Even if the user thinks that nothing is taking place, the machines resources and part of the band-width of connection are used for maintaining the connection. To simplify the comprehension of the problem, it is as if both entities communicating were permanently saying : one "Are you there" and the other would answer "yes I am".

In a stateless protocol, this permanent dialogue does not exist. The resources are used only during the lifespan of connection. If one returns to the detail of the network layer, the reality is much more complex and the use of the resources in the two modes is much more subtle. But the macroscopic vision of the situation remains the same: a stateful protocol is a much bigger consumer in resources than a stateless protocol.





**Protocole stateless**  
 There is no consumption of resources due to the protocol when the applications do not communicate

**Why protocols with conservation of the context?**

A question comes immediately to mind. Why are there protocols preserving the context since they are a bigger consumer in resources?

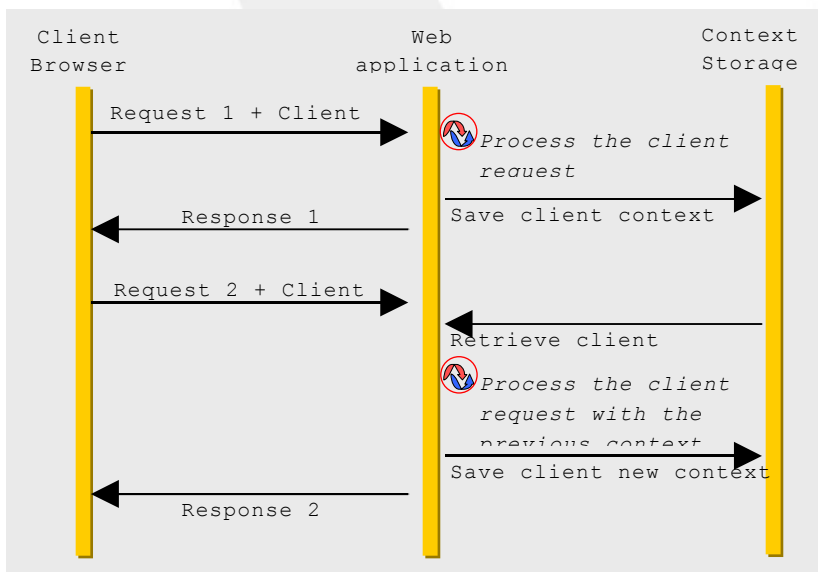
The answer is in the question: It is to keep a context between the two data-processing applications. The answer is simple. The conservation of a context gets a greater effectiveness when two applications communicate. To give a comparison with the real life, a stateful protocol would resemble the relation which exists during the maintenance of a private face to face meeting and a stateless protocol with the relation at the time of a larger meeting in a company where each one speaks very little, listening to the person that speaks and then 2 minutes later forgetting what he said. The relation is obviously not the same one but corresponds to two quite different needs. For certain applications, the conservation of the context can be essential (Synchronization of data, common management of transaction...)

It is the need for a context associated with a connection which will impose the choice of a stateful protocol.

**To make state with a stateless protocol**

On the Internet, HTTP is the typical example of the protocol without state. Once the response of the Web server is received, the connection is destroyed and nothing makes it possible to create a history or to connect to another request in time. The online Web Sales applications are examples of applications using a protocol without state (HTTP) but needing to preserve a context (the basket of purchases of the customer for example). The basket of purchase of a Net surfer is defined by a certain number of requests (a click for the fruits, for vegetables, for preserves...) which are all related to the account of the customer. There is a contradiction here between the possibility of leaving with a user with cumulation of their purchases and of behavior of the HTTP protocol.

To avoid this problem, Web developers created the contexts on the Web server independent of the protocol. When a user connects himself for the first time, a context is created on the server. It is related to the identification of the user. When a user emits a request it will it turn have a request to insert its identification (How the user is identified is not the object of this paper). It will be possible with this Web application to find the context of each customer at each request.



### **What it is necessary to retain of the protocols with or without conservation of the context**

The stateful protocol allows the creation and the conservation of a history of the exchanges between two data-processing applications thanks to the creation of a permanent connection between the two entities. This type of connection allows less consumption of resources. In contrast, the stateless protocols authorize a very significant number of short connections without possibility of history. The typical example of stateless connection is the HTTP protocol used by the Web

### **Conclusion**

#### **The step of the decision maker**

We have, in this paper, tried to simplify to the maximum our explanation on the characteristics of protocols, it can even, at points, be caricatured. However we believe that these simple rules can apply for a very large majority of architectures.

However, It should be known that in the majority of the cases, the protocol is often imposed by the technological choices on the level of the company or is a function of the partner with whom one wishes to communicate. It is rare that a manager is confronted with this type of decision, but when this moment arrives, it needs a simple and fast method to make its decision.

After having to evaluate which are the needs for the applications from a functional and technical point of view, a decision maker must be able to apply the few principles enacted in this paper, to put forward the judicious questions

Here some example questions:

Are the applications very related to one and other (strong or weak Bond)

Are the applications on two different continents (weak bond, asynchronous communication)

An application in the chain of processes is particularly slow (asynchronous communication)

Use of legacy imposes a type of communication (obligatory use of this protocol)

From there, one can apply the simple rules defined in this paper, thanks to which one will be able to understand and “challenge” the choices of his architects.

## **End of Document**

### **References**

#### **About the author**

Paul Perez is co-founder and Chief Software Architect of Pymma. Paul is an author, software architect consultant and speaker, brings more than 17 years experience helping corporations utilize object technology for mission-critical information systems. Prior to co-found Pymma, Paul was an independent software consultant, working for large financial services companies in France, UK, Benelux, Israel and Germany. Paul's extensive experiences in high-performance architecture design helps the company's clients solve real-world business problems through technology. As chief software architect, Paul assists current and future client engagements and develops best practices based on his domain expertise. Paul holds a post master degree from Paris-Orsay University and is a Sun Certified Enterprise Architect and holds several other certifications