Communication Patterns and Input Patterns in Distributed Computing
(Invited Talk)

Michel Raynal
Institut Universitaire de France &
IRISA, Université de Rennes, France &
Department of Computing, Polytechnic University, Hong Kong
raynal@irisa.fr

This paper was written during the week January 5-9, 2014.
“Je suis Charlie...” (January 7, 2014)

Abstract. A communication pattern is a pattern on messages exchanged in a distributed computation. An input pattern is a vector made up of the input parameters of the processes involved in a distributed computation. This paper investigates three such patterns. The first two, which are related to the causality relation associated with a distributed execution, are on causal message delivery and the capture of consistent global states, respectively. The last one, which concerns the consensus problem, is on vectors defined by the input values proposed by processes (this is also called the “condition-based” approach).

An aim of the paper is to promote the concept of pattern in distributed computing, both as a way to provide higher abstraction levels (as it is the case in communication patterns), or a tool to investigate computability or optimality issues (as it is the case with input patterns).

Keywords: Agreement problem, Byzantine failure, Causality, Causal message order, Checkpointing, Consensus, Crash failure, Error-correcting code, Input vector, Message pattern, Zigzag path.

1 Introduction

On Patterns Encountered in Computing. In this paper a pattern is seen as a specific arrangement of objects (messages, control flows, processes, input data, etc.) whose aim is to provide either regular structures, or an appropriate abstraction level, or an appropriate setting, which facilitate the design of algorithms solving distributed computing problems.

Maybe one of the most famous patterns encountered in computing science is the pattern used by William George Horner (1786-1837) to compute a polynomial, namely,

\[
\cdots ((a_n \times x + a_{n-1}) \times x + a_{n-2}) \times x + a_{n-3}) \times x + \cdots + a_1) \times x + a_0.
\]

\footnote{In the SIROCCO context, a pattern can be seen as a specific type of structural information.}
The basic pattern $A \times x + a_i$ is iteratively used to obtain a very simple algorithm, which uses $n$ multiplications and $n$ additions (let us notice that this pattern-based method was known and used by Zhu Shijie, 1270-1330, under the name *fan fa* [40]).

More generally, all control structures of sequential computing (such as loops, and predicate-based statements) can be seen as familiar computation patterns. In the domain of parallel computing, where one has to solve problems whose solutions can be based on a regular structure, the pattern-based approach called *systolic programming* has proved to be both easy to use and efficient [10].

When considering the distributed setting, the situation is different. Only a few basic patterns have been abstracted and are now recognized as fundamental. One of them, introduced to help structure distributed computations, is called *round-based* computation. This pattern generalizes the notion of iteration to (both synchronous and asynchronous) distributed computing.

### Content of the Paper

This paper is a short scientific essay on patterns in distributed computing. As it is an essay, its aim is neither to be exhaustive, nor to give research directions. More precisely, the paper considers two kinds of patterns, one related to the causality created by messages exchanged by computing entities (processes), while another is related to the input data from which the processes have to agree.

**Patterns Related to Message Exchange.** In addition to the data they carry, messages create a causality relation (from causes to effects) among the events produced by the processes defining a distributed computation. This relation, expressed for the first time in 1978 by Lamport [21], is a master key to solve causality-related problems [35].

The paper presents two causality-based problems related to message exchange patterns. The first one, called *causal message delivery*, is addressed in Section 2. As indicated by its name, its aim is to reduce the asynchrony (noise) in message delivery, namely, for any process $p$, the delivery of the messages sent to $p$ has to respect their causal sending order. Hence, the aim is here to provide processes with a higher abstraction level where message delivery is always in agreement with the causality relation on their sending.

The second problem, which is related to the computation of consistent global states (also global checkpoints), is addressed in Section 3. As a global state is made up of a local state per process, its consistency requires that no two of its local states causally depend on one another. An important issue is then to ensure that any local state defined as local checkpoint belongs to a consistent global state. The difficulty here is related to the existence of hidden dependencies (captured with the notion of a *zigzag path* [29]). The paper will show how it is possible to cope with these hidden dependencies by demanding processes to take additional local checkpoints so that all checkpoints patterns are such that any local checkpoint belongs to a consistent global state/checkpoint.

**Patterns Related to Input Data.** The second type of pattern investigated is related to the consensus problem in asynchronous systems where processes may crash, or even commit Byzantine failures. As consensus is impossible to solve in asynchronous systems where even only one process may crash [12], it remains impossible to solve in