# Understanding Distributed Program Behavior Using a Multicast Communication Scheme

Mihai Mocanu[1] and Emilian Guţuleac[2]

[1] University of Craiova, Software Engineering Dept., Bvd.Decebal 107, Craiova, RO-200440, Romania
`mocanu@software.ucv.ro`

[2] Technical University of Moldova, Computer Science Dept., 168 Bd. Ştefan cel Mare, Chişinău, MD-2004, Republic of Moldova
`egutuleac@mail.utm.md`

**Summary.** Events in a distributed global computation framework, unlike those in a sequential local computation, form a partially ordered set with respect to the causality relation revealed by timestamps. This paper describes a new logical timestamping mechanism based on multicasting, called *Collective Logical Time*, and compares it with other known schemes that have been developed in the domain mainly to help in detecting undesired (global) properties of distributed computations (such as deadlock). Unfortunately, due to excessive complexity and some unrealistic restrictions (such as a fixed number of processes), these schemes have produced limited results. Some of the benefits in using our scheme are revealed, together with the possibilities for direct applications in the development of low-level communication protocols.

## 1 Introduction

Tracing event execution in a distributed computing environment (DCE) might seem a simple idea if we want to analyze a programs' behavior, its efficiency, or with respect to unusual event occurrences, but the formation of correct global time measurements is a difficult task. It is hard to understand an execution using a set of traces, due to the non-deterministic duration (Non-DD) of processes, as a consequence of their distributed nature and multiple interactions. Time is not absolute and the events in DCEs, unlike those in a sequential computation (SC), form a partially ordered set with respect to the causality relation, revealed by timestamps. Real time clocks are not relevant here, more useful to track causal dependencies between global events is the "logical" time, based either on scalar [1] or vector clocks [2] [3]. Difficulties here are in mapping partial order of distributed events into total ordering, or in the need to setup a constant, known-in-advance number of processes in the DCE [4][5]. Moreover, a distributed program may be easily perturbed by a metric code inserted, or too sensible to the application architecture and deployment decisions.

We present here a new logical timestamping mechanism - named *Collective Logical Time* (CLT), and a way to build it on top of a multicast scheme. We show also some of its advantages over other known schemes and how can it be directly applied in the construction of concurrency domains of execution, defined and denoted here as *Collective Work Domains* (CWD). These can in turn be used to understand the behavior and the implications of alternative implementations for the overall performance.