Towards a Library of Distributed Algorithms and Animations (Lydian)

Marina Papatriantafilou   Philippas Tsigas
Computing Science Department
Chalmers University of Technology and Gothenburg University
S-412 96 Gothenburg, Sweden
Email: (ptrianta,tsigas)@cs.chalmers.se

April 1998

Summary

As distributed algorithms may involve a large amount of data describing local state information and complex interactions between elements, it is often very difficult to achieve an understanding of their control flow (and performance behaviour) only from a pseudo-code description or from data streams (e.g. execution traces). This paper is on our project which constitutes a systematic effort to explore the use of visualisation and animation aids to illuminate the key ideas of distributed protocols and algorithms. In particular, it involves:

- development of a database of distributed protocols and concurrent data objects implementations;
- availability of an animation program for each entry of the database (i.e. each protocol or concurrent object implementation), which, given the (on- or off-line) trace of any execution, it animates it in a way that demonstrates the key ideas of the respective protocol or concurrent object implementation;
- development of a database of network descriptions.

The integrated library, Lydian, will be of big help for educational purposes, such as for teaching distributed computing, computer networks, communication protocols, operating systems; students will be able to gain a direct impression of the behaviour of the protocol and teachers will be able to illustrate concepts that can otherwise be explained concisely only in a technical paper.

For the protocols and data objects implementation part we use simulation platforms for network and multiprocessor systems. Since distributed protocols are designed to execute on any type of system that can be described as a set of interconnected processing units, the use of these platforms enables the implementation of a wide range of protocols. For the creation of network descriptions and their traffic behaviour a simple and nice graphical editor was used accompanied by a set of drawing algorithms.

For the animation part we use a powerful toolkit, which, like the simulation platforms, is appropriate for many different architectures.

1 Introduction

The nature of distributed computation —i.e. the existence of multiple flows of control in distributed protocols and algorithms, together with the non-determinism in timing and the non-existence of global state information in the system— often leads to the sort of complicated interactions, which make it difficult to understand and explain the essential issues of distributed algorithms using simple methods. Pseudo-code descriptions, simple or even multiple data streams (e.g. execution traces) are most often inadequate to convey the intuition, the ideas and methods for the solution of an algorithm. In addition, distributed algorithms vary in nature, since they are designed to execute on systems which vary in type, from heterogeneous message-passing systems, "distributed" over some geographical area, to shared memory multiprocessors, or any type of system that can be described as a set of interconnected processing units.

We have started a systematic effort to explore the use of visualisation and animation aids to illuminate the descriptions of distributed algorithms and concurrent data object constructions. The idea is for the animation of any interactive execution of each protocol in the archive to use on- or off-line traces of the execution and demonstrate the "key ideas" of the functionality of the solution and the protocol's behaviour under different timing and workload of the system.

The resulting library of animations for distributed protocols, Lydian, will provide useful material for *educational purposes* —such as for teaching distributed algorithms, computer networks, communication protocols, operating systems—, and for establishing connections with interested research, teaching and implementations parties.

1.1 State of the art

The exploitation of visual methods to present distributed system's executions has been realized in distributed performance debugging tools; they are useful for performance monitoring, but at the same time the data they represent is essentially independent of the application being studied —since performance metrics demonstrate more about the behaviour of the system, rather than the particular algorithm being executed.

Current research efforts in the area of distributed computing focus in the development of generic tools for visualisation of algorithms execution, e.g. PARADE [9], NESL[1]; the idea for the latter tool is a parallel programming language that comes with a visual interface; the idea in the PARADE system is to enable instrumentation of concurrent programs and to provide a set of optional views, each of which can visualise some specific aspect of the execution of the program (e.g. messages being transferred, local states, etc); the user can choose a subset of the views to monitor information of interest. This has been a fruitful effort; flow of control is described in a much better way than a single information stream. However, the interpretation of this information into something that resembles the way that a human can understand an algorithm is still left to the observer; i.e. it is still the case that a group of streams —containing a large amount of data— is provided and that a significant amount of cognitive effort is required to extract the useful information from it. If a user needs views that match the human understanding of the protocol, he/she has to write an animation program, using the underlying toolkit POLKA [8].

This is also where our point comes into picture: our idea is to use directly POLKA, which is powerful and highly portable, to create an animation program (entry) for each of a wide range of protocols and concurrent data object constructions that we implement and maintain in Lydian's archive.

To the best of our knowledge, there has been only one attempt towards such a systematic archiving of distributed protocol and algorithms animations, ZADA [6], based on the animation package Zeus, a Modula-3 based system for specialised platforms, —not as highly

Figure 1: Basic components and functionality in Lydian

portable. The ZADA project resulted in a small archive of protocols, for each of which the implementation is the same program as the animation (this implies essentially fixed timing, workload, etc). The project lasted for only one year; members of the responsible group who initiated this effort at the University of Dortmund are now at different universities.

A very successful example of the world of sequential computing is LEDA [5], which started as a library of data structures and algorithms and has seen significant benefits from its added visualisation and drawing features; LEDA is being widely used as educational tool in various universities, as well as a development tool in industry.

The effort for implementing our archive of protocols and animation programs has started several months ago. The first steps have been very successful and the results have already been used for teaching a type of concurrent data objects in class. The following sections describes in more detail the objectives and the methodology for the whole project and the expected benefits in the educational domain.

2 Research and work under development for Lydian

The work for this project, which has been summarised in the beginning of this document, involves (cf. also figure 1):

• The development of a database of distributed algorithms and protocols implementations; for this we are using efficient simulation platforms, which allow non-intrusive monitoring and make it easy to repeat executions so that different phenomena can be studied at a variety of detail levels and system characteristics. We provide options for (i) implementing protocols for different types of systems, (ii) describing features of various systems to be simulated (such as varying topologies, timing, options to simulate failures, shared and distributed memory, various types of interconnection media, option for cache memory chips modelling, etc.), (iii) specifying protocols to be executed on selected simulated distributed systems, (iv) producing execution traces with optional information request (option for various debug classes of information per trace).

• The provision, for each entry of the database (i.e. each protocol or concurrent object implementation), of an animation program, which, given the (on- or off-line) trace of any execution, it animates it in a way that demonstrates the key ideas of the protocol. For this part we are using POLKA [8], which provides a rich library of visualisation and animation features, and appropriate for many application platforms, as opposed to systems like the Pavane [2], IVE [3]. The user needs only to program in C++ using calls to the POLKA library, and does not need to program anything directly in windows (unlike other toolkits e.g. the ParaGraph [4] or the Voyer [7]); this constitutes a big advantage, as the latter may be often be a tedious task.

We also leave an option for use of additional tools/components, if, e.g. a need arises for the simulation of special-purpose distributed systems (e.g. anti-blocking system, navigation system of a vehicle, space stations, ...)

3 Expected educational benefits from Lydian

We believe that Lydian will be very helpful for educational purposes, such as for teaching distributed computing, computer networks, communication protocols, operating systems; it will offer a tool to teachers to illustrate concepts that can otherwise be explained concisely only statically in the static blackboard with the help of a book or a technical paper, and it will give to the students a direct impression of the behaviour of the protocols. In all the above mentioned courses the teacher has to describe systems where threads of control compete for resources, try to synchronise and dynamically change execution behaviour. Each execution involves many processes, a large amount of data of processes' local state to describe the system state and an even larger amount of date to describe complex interactions between the processes. Moreover, different executions of the same algorithm, even if they start from the same initial system configuration, may not result in the same output, due to asynchrony.

Lydian is build to have the following two basic characteristics:

- Lydian offers an easy, visual way to the students to produce their own network description parameters as well as the traffic parameters and thus their own execution that they would like to use in order to see the behaviour of the protocol or algorithm; while at the same time lydian gives the ability to the students to select one possible execution from a database of executions that are part of Lydian. After selecting or creating such a kind of an execution then they can use it in order to see the behaviour of an algorithm in such this specific execution.

- For each protocol in the database, the animation will be based on the basic ideas behind the design of the algorithm. Ideas that are used in the classroom to describe the corresponding algorithm its correctness and its analysis. These ideas will be demonstrated under any different execution selected by the teacher or the students.

References

[1] Guy E. Blelloch, Jonathan C. Hardwick, Jay Sipelstein, and Marco Zagha. NESL User's Manual (3.1). *CMU-CS-95-169*, September 1995.

[2] K.C. Cox, G. Roman Visualizing concurrent computations. *Proceedings of the IEEE Workshop on Visual Languages*, pp. 4-9, 1991.

[3] M. Friedell, M. LaPolla, S. Kochhar, S. Sistare, J. Juda Visualizing the behavior of massively parallel programs. *Supercomputing '91*, 472-480, 1991.

[4] M.T. Heath, J.A. Etheridge Visualizing the performance of parallel programs. *IEEE Software*, 8(5):29-39, 1991.

[5] K. Mehlhorn, St. Näher The LEDA Platform of Combinatorial and Geometric Computing. *Cambridge University Press*, 1998.

[6] A. Mester, P. Herrmann, D. Jager, V. Mattick, M. Sensken, R. Kukasch, A. Ritter, S. Bunemann, P. Unflath, M. Bernhard, F. Austel, T. Alders, A. Rohrbach ZADA: Zeus-based animations of distributed algorithms and communication protocols. *T.R. Universität Dortmund*, 1995.

[7] D. Socha, M.L. Bailey, D. Notkin Voyer: Graphical views of parallel programs. *SIGPLAN Notices*, 24(1):206-215, 1989.

[8] John Stasko POLKA Animation Designer's Package. *Technical Report*, Georgia Institute of Technology, 1995.

[9] John Stasko The PARADE Environment for Visualizing Parallel Program Executions: A Progress Report. *Technical Report GIT-GVU-95-03* Georgia Institute of Technology, Atlanta, GA, 1995.