RemoteLab: A Reliable Tele-Laboratory Environment[1]

Janek Schwarz, Andreas Polze, Kristopher Wehner[‡], and Lui Sha[‡]

February, 10, 2000

Departement of Computer Science
Humboldt University of Berlin
Rudower Chaussee 25
12489 Berlin, Germany
Tel: +49 +30 2093 3028
Fax: +49 +30 2093 3029

{apolze|schwarz}@informatik.hu-berlin.de

[‡]Departement of Computer Science
University of Illinois Urbana-Champaign
1304 W. Springfield Ave.
Urbana, IL 61801
Tel: 217-244-1887

{wehner|lrs}@cs.uiuc.edu

Abstract

Experiments on real physical systems play a major role in engineering disciplines. Traditionally, the use of real physical systems requires a students' physical presence in the laboratory. We have developed the RemoteLab *tele-laboratory environment, which allows students to remotely conduct physical experiments via the Web, creating a reliable integrated virtual and real laboratory education environment.*

The main challenge in such a situation comes from the fact that students interact with non-fail-safe laboratory equipment without on-site human supervision. Our solution uses the online-replacement mechanisms in the previously developed Simplex system. Simplex is based on the concept of analytic redundancy and provides a "supervisor" for experimental, student-supplied controllers in form of an ultra-reliable safety controller, which eventually takes over control of the experiment.

Within this paper, we describe a RemoteLab-experiment where a robot is controlled by a joystick via a CORBA event channel and the Simplex system. The actual experiment has been carried out in a distributed fashion between the authors' universities. Within this context, the paper reports on the extension of the fault-tolerant real-time Simplex control system with open, CORBA-based interfaces.

Key words*: RemoteLab, tele-education, real-time control systems, online component replacement, CORBA*

1 Introduction

Tele-teaching is a modern education model, which extends the traditional face-to-face apprenticeships by using electronic networks, to provide a powerful context for learning in education courses. During the last years, several sites deployed tele-teaching environments for various purposes, with different goals, using different technologies.

[1] This work is supported in part by the Office of Naval Research and in part by German Research Network (DFN).

The DIANA project at Humboldt University has established a synchronous tele-teaching environment used for transmitting lectures from the Computer Science department to Humboldt University's main campus in Berlin's city center over a distance of 25 km. This Giga-bit tele-teaching link is currently extended to include the Technical University in Munich. The DIANA project focuses on reliable audio and video transmission using stream-based communication (CORBA Audio/Video streams), it employs a CORBA-based platform for device control (tele-teaching equipment). In context of DIANA we have developed the *RemoteLab* tele-laboratory environment, which allows students to remotely conduct physical experiments via the Web, creating a reliable integrated virtual and real laboratory education environment.

In a traditional laboratory environment, several mechanisms, i.e. switches, which have to be pushed to make a device operable, ensure the safety of non-failsafe equipment. These mechanisms are not applicable in a remote laboratory environment. However, fault-tolerance regarding faulty behaving controllers is crucial for the safety of the equipment. In our system we use analytic redundancy, introduced first with the Simplex architecture [Sha96], to circumvent this problem.

Our RemoteLab scenario is not restricted to the education domain only. The technology used in this environment is equally well applicable to other scenarios, such as manufacturing, tele-repair or tele-monitoring.

The remainder of this paper is organized as follows: Section 2 describes the tele-laboratory. Section 3 gives an overview on the underlying principles. Section 4 extends our current scenario by introducing a CORBA-based framework for component replacement. Section 5 presents related work and Section 6 concludes the paper.

2 A Tele-Laboratory Environment

Experimentation with real physical systems is a key part in education. Our RemoteLab tele-laboratory project allows students to remotely conduct physical experiments via the Web easily and reliably, creating an integrated virtual and real laboratory education environment. The tele-laboratory is based on Simplex technology, which is described in some detail in Section 3. The novelty of our project lies in:

1. The use of real experiments in addition to simulation and animation without being physically at the laboratory. This is achieved through the real-time, fault-tolerant control architecture based on Simplex principles, which guarantees stable performance while allowing users to download and test their control modules.

2. Horizontal integration across engineering disciplines. Our tele-laboratory is developed with a common interface allowing students from different disciplines to gain practical experience with a variety of control and distributed computing problems and devices.

3. Integration with standard tools for controller development. Our laboratory software has interfaces to off-the-shelf tools used by control engineers today, such as Matlab+Simulink.

In an instructional situation it is likely that the user-supplied controllers will malfunction, possibly in dangerous ways. Therefore, it is necessary to take precautions to protect non-fail-safe laboratory equipment. Simplex provides this protection using the technique of analytic redundancy, and can detect and replace a malfunctioning controller with the predefined safety controller to keep the equipment safe and in working order.

With the Java-based lab applet, the user-visible portion of the tele-laboratory, the student is given feedback as to the state of their controller (running, failed), as well as the system states that led to the termination of their controller in the case of failure. In addition, the student is provided with a video stream of the physical device and graphs of its status. Facilities are also in place for the controller to transmit debugging information output during a run back to the student after execution completes (so as to not interfere with the real-time requirements of the student controller).

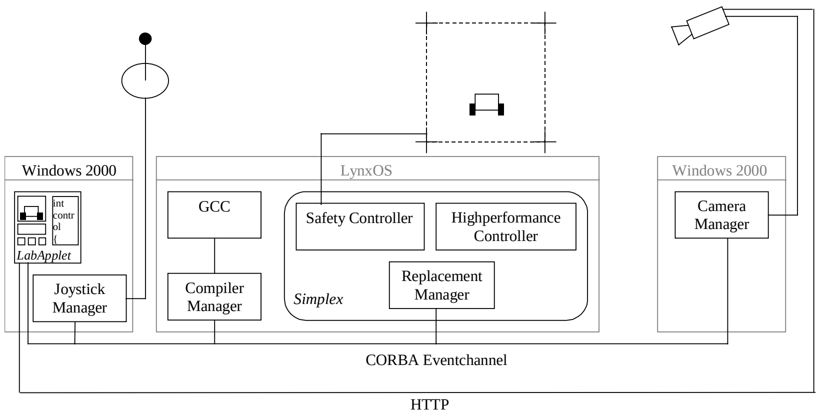

Figure 1: Tele-laboratory scenario

The scenario depicted in Figure 1 shows an experiment, where a Khepera robot [K-Team95] is driven by an interactive user's commands entered via a force-feedback joystick. The joystick commands are transmitted to the robot via a CORBA event channel, the user-supplied ("highperformance") controller and the CORBA-enhanced version of Simplex. The robot has a limited, imprecise position tracking system, which supplies data to the user-supplied controller, which in turn sends feedback to the joystick.

The independent safety controller is connected to a rectangle of light barriers, which enable it to judge whether the user-supplied controller is moving the robot out of the safety region (i.e. the table). In this case, the Simplex online-replacement mechanism transfers control of the robot solely to the safety controller and terminates the user-supplied controller. The Simplex replacement manager is accessible via a CORBA interface. Using the lab applet, the user may request the compilation of his controller code by communicating with the Simplex compilation manager, which has an interface to the C compiler and coordinates its tasks with the replacement manager.

After successful compilation the user can request insertion of the controller into the active Simplex system. Simplex then executes the controller and provides status information about the execution, which are presented to the user by the lab applet together with the video feedback.

3 Underlying Principles: The Simplex Architecture

The Simplex architecture [Sha96] has been developed for automatic control applications to support safe and reliable online upgrade of software components in spite of errors in new modules. This is important when introducing new control technologies into running systems.

To mitigate the risks in a control system upgrade, Simplex has been designed to tolerate timing faults such as overrun, programming system faults such as illegal addressing, and semantic faults due to modeling, algorithm design or implementation errors. Simplex handles timing faults and programming system faults by temporal and spatial encapsulations. Software semantic faults are handled by the use of analytic redundancy.

The basic building block of Simplex Architecture (see [Sha96]) is a replacement unit. A replacement unit is simply a process with a given communication template. Replacement units are organized into application units. An application unit typically has communication and process management modules, and at least one replacement unit that implements the application functions. If the system is safety critical, a separate safety unit is needed. The safety unit is responsible for reliable operation and operation monitoring.

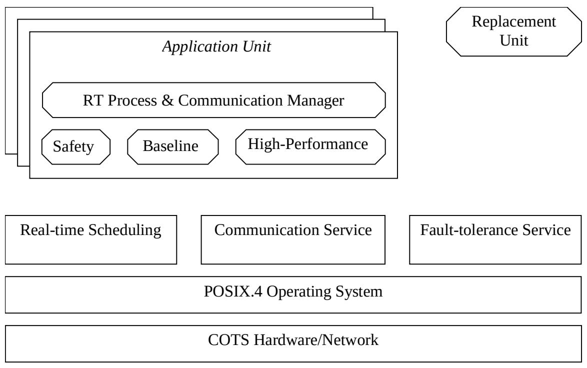

Figure 2: The Simplex architecture

Simplex uses forward error recovery for semantic faults in control applications. System states reached by the experimental controller must stay within the stability envelope of the safety controller. Intuitively, this is similar to a student pilot's flight test. The trainer will let the student pilot fly, even making some minor mistakes, as long as the plane remains in a state that is controllable by the trainer. The safety controller is designed to have a large stability envelope at the cost of performance whereas the experimental controller is designed to maximize the performance. They have different designs but analytically related design objectives.

The experimental controller is encapsulated in a replacement unit and can be replaced during runtime. The replacement of a component is not a real-time operation. However, once the component takes over the control, the computation and communication involving the component are hard real-time operations. Simplex uses POSIX.21 real-time message queues for communication among controllers, however, it provides CORBA interfaces to the replacement procedure.

4 A CORBA-based Framework for Component Replacement

Within context of Simplex, analytic redundancy and online-replacement have been proven as viable concepts for the construction of reliable dynamic real-time software systems. However, programming for the Simplex system requires adherence to a number of coding conventions. In order to support dynamic replacement, Simplex components (replacement units) have to support certain interaction patterns, which are expressed in actual component code, rather than in an interface definition language.

Introduction of CORBA IDL would significantly simplify programming of Simplex replacement units. However, since CORBA interactions have varying, unpredictable communication latency, some measures have to be taken to ensure Simplex' real-time behavior. Since the component replacement operation in Simplex is not a real-time operation, it can be easily replaced by a CORBA-based service to achieve a higher degree of flexibility and improved portability.

The open question is, how to interface the real-time operations with CORBA. There exist a number of object request broker implementations with real-time extensions, such as the TAO ORB. In certain system configurations, where a hard real-time control loop is statically distributed over several nodes, the RT-CORBA implementations with their specific communication protocols show great advantages over proprietary solutions. However, in systems, where the hard real-time loop is not distributed, such as in the tele-lab (see Section 2), it is sufficient if middleware provides a distributed multimedia user interface to the system and supports component management with soft real-time characteristics.

In case of our tele-lab scenario, the non-distributed nature of the real-time control loop does not require the use of RT-CORBA. The challenge in our case is the integration of standard

CORBA with a proprietary real-time system. We introduce the notion of a “Composite System” to solve the problem.

Within the Composite System, components’ interfaces are described using CORBA’s interface definition language IDL. The construction of new replacement units is simplified. Additionally, many standard CORBA services (COS – common object services), such as naming service, event service, or domain specific services, can now easily be used from within the replacement units. On the other hand, we retain the real-time publication and subscription service of Simplex to ensure the timing predictability for safety controller and decision unit, Simplex’s most vital components. Thus, we use Simplex’s inherent concept of analytical redundancy to tolerate CORBA’s unpredictable timing behavior.

To interface the real-time operations with CORBA we use a previously developed concept called “Composite Objects” [Polze98]. Composite objects allow the programmer to make an explicit tradeoff between an application's predictable resource utilization and its communication latency.

Composite objects consist of a real-time part and a non-real-time part. Design time and run-time guarantees can be given for execution of real-time methods. In contrast, methods in the non-real-time part are executed following a best-effort approach. Data replication and a weakly consistent memory management protocol (implemented via interprocess communication – such as POSIX message queues) are used to decouple real-time and non-real-time processing.

A composite object establishes a timing firewall [Polze97] between the real-time and non-real-time (CORBA) computing part, so that the non-real-time part cannot violate the real-time scheduling rules that are needed by the real-time part [Sha94]. Implementation details on Composite Objects and timing firewalls using a scheduling server can be found in [Polze97] and [Polze98].

In our approach towards a CORBA-based version of Simplex, we implement decision unit and safety controller inside the real-time part of composite objects. We use the real-time publisher/subscriber protocol for communication between safety and decision unit. This way we can ensure predictable timing behavior for interactions between these two essential components.

As in the original Simplex system, the safety unit implements the basic version of the desired service. Additional, more sophisticated and complex versions of a service now can be implemented as CORBA components. These components are described by their interfaces given in CORBA IDL. Multiple versions of the CORBA service implementations correspond to Simplex’s replacement units and inside the decision unit Simplex’s concept of analytic redundancy is used to judge a components output.

Since CORBA communication is location-transparent, we can now distribute the Simplex system in a heterogeneous environment. Simplex’s predictable timing behavior is preserved as long as safety unit and decision unit are co-located on a real-time operating system (as in the original Simplex system).

5 Related Work

Real-time and fault-tolerant CORBA

The idea of providing fault tolerance as an additional feature to CORBA implementations has been the focus for several research activities within the last years. With the request for proposal for a *Fault tolerant CORBA Using Entity Redundancy* [OMG98a] issued in April 1998, OMG is seeking to incorporate existing approaches for software fault tolerance into future versions of CORBA. However, most related work, i.e. [Morgan99], [Felber98], [Chang97], [Maffeis94], implements fault-tolerant behavior based on redundant execution and fault-masking without taking real-time constraints into account, a property absolutely needed for ensuring the safety of the devices in a tele-laboratory. While the OMG has founded a Real-time CORBA special interest group (SIG), which has been soliciting technology for a *Real-time Object Request Broker* (ORB) comprising: fixed priority scheduling, control over ORB resources for end-to-end predictability, and flexible communications [OMG99], the integration of fault-tolerance techniques with real-time computing is currently not under consideration in the OMG standardization process.

TAO is an innovative work on RT-CORBA where fixed priority real-time scheduling is tightly integrated into the system [Schmidt98] [Schmidt99]. Main goal of this work is to provide end-to-end quality-of-service qualities for CORBA-based applications. A list of requirements for Object Request Broker implementations is presented, among them are resource reservation protocols, optimized real-time communication protocols and a real-time object adapter.

Work at the University of Rhode Island and the MITRE Corporations deals with syntactical extensions to CORBA IDL to express timing constraints [Squadrito98] [Thuraisingham96]. “Timed distributed method invocations” are identified as one necessary feature in a real-time distributed computing environment. The “Affected Set Priority Ceiling Protocol” as a combination of semantic locking and priority ceiling has been proposed for concurrency control in real-time object-oriented systems.

Tele-Education

Tele-teaching environments are being used by several organizations. They differ in they way teacher and student access the material and how interaction between teacher and students happens. In an asynchronous environment, teachers and students prepare and access the material, such as audio-visual recordings of lectures or multimedia learning programs, at different times. Interactions between lecturer and students happen asynchronous via electronic mail or synchronous via telephone or video conferences. Known examples for asynchronous tele-teaching environments among others are DIALECT [Apostolopoulos96] and West [McLoughlin96]. Telepoly [Walter97] is a synchronous environment. Telepoly connects several classrooms to a virtual classroom constructing an interactive and synchronous distance learning environment.

While various sites deployed distance learning environments, only a few support remote training and laboratories. SimulNet [Llamas98] is a distributed remote access computer based training system, which provides a virtual laboratory enabling students to train theoretical knowledge in practice. This is done by delivering software through the Internet which can be run on any computer. These distributed applications are simulators of tools which can be found in a conventional laboratory. The main difference to our RemoteLab environment is the use of simulators. While our architecture allows students to experiment with real devices, SimulNet provides only simulators.

6 Conclusions

We have presented the RemoteLab tele-laboratory application, which enables students from remote locations to design, compile, and run their controllers using (possibly expensive) laboratory equipment in a predictable, reliable fashion. The main challenge in such a situation comes from the fact that students interact with non-fail-safe laboratory equipment without on-site human supervision.

Our solution uses the online-replacement mechanisms in the previously developed Simplex system. Simplex is based on the concept of analytic redundancy and provides a “supervisor” for the student-supplied controllers in form of an ultra-reliable safety controller, which eventually takes over control of the experiment.

We have extended Simplex’ replacement manager with CORBA interfaces, allowing installation and execution of new controllers to be triggered remotely. Furthermore, we have developed a Java-based lab applet which has been successfully used to conduct remote experiments. Within this paper, we have described an experiment where a robot is controlled from a joystick via a CORBA event channel and the Simplex system. We have run this experiment distributed between University of Illinois, Urbana-Champaign and Humboldt-University of Berlin, Germany. In this case, a remote user has accessed the robot in our Berlin lab.

In order to be usable in our system, controllers currently need to adhere to a number of design rules and a Simplex-specific interaction protocol. Future work will focus on modifications to the Simplex architecture to support execution of purely CORBA-based controllers in our environment.

References

[Apostolopoulos96] N. Apostolopoulos, A. Geukes, S. Zimmermann; "DIALECT – Network-based digital lectures"; Computer Networks and ISDN Systems, 28, pp 1873 – 1886, 1996.

[Chang97] Y.-S.Chang, D.Liang, W.Lo, G.-W.Sheu, S.-M.Yuan; "A Fault-Tolerant Object Service on CORBA"; Proceedings of International Conference on Distributed Computing Systems (ICDCS'97), Baltimore, May 1997.

[Felber98] P. Felber; "The CORBA Object Groups Service – A service approach to object groups in CORBA"; Phd Thesis, École Polytechnique Federal de Lausanne, 1998.

[K-Team95] K-Team SA; "Khepera User Manual, Version 4.06"; K-Team SA, Chalet du Vuasset, CP 111, 1028 Preverenges, Lausanne, Switzerland, 1995.

[Llamas98] Martín Llamas, Luis Anido, Manuel J. Fernández; "Student Participation and First Results from SimulNet, a Distant Access Training Laboratory"; En Gordon Davies (ed.) Teleteaching 98 Distance Learning, Training and Education. Proceedings of the XV. IFIP World Computer Congress. Österreichische Computer Gesellschaft, 1998.

[Maffeis94] S.Maffeis; "A Flexible System Design to Support Object Groups and Object-Oriented Distributed Programming"; in Proceedings of ECOOP'93, Lecture Notes in Computer Science 791, 1994.

[McLoughlin96] H. McLoughlin; "WEST: An internet based education delivery and support environment"; Computer Networks and ISDN Systems, 28, pp. 1887 – 1890, 1996.

[Morgan99] G. Morgan, S.K. Shrivastava, P.D. Ezhilchelvan, M.C. Little; "Design and Implementation of a CORBA fault-tolerant object group service"; in Proceedings of the 2nd IFIP WG6.1 International Working Conference on Distributed Applications and Interoperable Systems (DAIS), Helsinki, June 1999.

[OMG98a] OMG; "Fault tolerant CORBA Using Entity Redundancy RfP"; OMG Document orbos/98-04-01, at http://www.omg.org.

[OMG99] OMG; “Realtime CORBA Architecture”; OMG (Object Management Group) document orbos/99-06-02.

[Polze97] Andreas Polze, Gerhard Fohler, Matthias Werner; “Predictable Network Computing”; in Proceedings of 17th International Conference on Distributed Computing Systems (ICDCS'97), Baltimore, USA, May 1997, IEEE Computer Society Press, pp: 423-431(9), ISBN 0-8186-7813-5.

[Polze98] Andreas Polze, Lui Sha; “Composite Objects: Real-Time Programming with CORBA”; in Proceedings of 24th Euromicro Conference, Network Computing Workshop, Vol.II, pp.: 997-1004, ISBN 0-8186-8646-4, Vaesteras, Sweden, August 25-27, 1998.

[Schmidt98] D.C.Schmidt, D.Levine, and S.Mungee; “The Design and Performance of Real-Time Object Request Brokers"; Computer Communications, Volume 21, No. 4, April, 1998.

[Schmidt99] F. Kuhns, D. Schmidt, and D. Levine, “The Design and Performance of a Real-Time I/O Subsystem”, in Proceedings of the 5th IEEE Real-Time Technology and Applications Symposium (RTAS), Vancouver, Canada, June 2-4 1999.

[Sha94] L.Sha, R.Rajkumar, S.S.Sathaye; "Generalized Rate-Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems"; Proceedings of the IEEE, Vol. 82, No. 1, January 1994.

[Sha96] L.Sha, R.Rajkumar, M.Gagliardi; "Evolving Dependable Real-Time Systems"; in Proceedings of 1996 IEEE Aerospace Applications Conference, IEEE Inc., February 1996, ISBN 0-7803-3196-6.

[Squadrito98] M.Squadrito, L.Esibov, L.C.DiPippo, V.F.Wolfe, G.Cooper, B.Thuraisingham, P.Krupp, M.Milligan, R.Johnston; "Concurrency Control in Real-Time Object-Oriented Systems: The Affected Set Priority Ceiling Protocols"; Proceedings of First IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC), pp. 96-105, April 1998, Kyoto, Japan, IEEE Comp. Soc. Press, ISBN 0-8186-8430-5.

[Thuraisingham96] B.Thuraisingham, P.Krupp, and V.Wolfe; "Position Paper: On Real-Time Extensions to Object Request Brokers"; in Proceedings of Second Workshop on Object-Oriented Real-Time Dependable Systems (WORDS), February 1996, Laguna Beach, CA, USA, IEEE Comp. Soc. Press, ISBN 0-8186-7570-5.

[Walter97] T. Walter, B. Stiller, B. Plattner, H. Hänni; "Distributed Learning Scenari-os supported by ATM – Experiences in Telepoly"; in Proceedings of Fachtagung Me-diengestützte wissenschaftliche Weiterbildung, pp. 69 –83, Braunschweig.