Cooperative Computing in Dynamic Environments MIT9904-12

Progress Report: January 1, 2001—June 30, 2001

Nancy Lynch and Idit Keidar

Project Overview

The Theory of Distributed Systems research group at MIT, led by Prof. Nancy Lynch, is working with the Cooperative Computing group at NTT on developing models and analysis methods for distributed systems, with a focus on cooperative group activities in networks. Such group activities range from human social activities in cyber communities to powerful distributed applications involving data sharing and cooperative work. These activities are often supported by agent communication services, which provide distributed intelligence, or by group communication services (GCSs), which manage group membership and guarantee coherent communication. The environments in which such activities take place are highly dynamic: participants come and go (and change location), network topology changes, and components fail and recover. Coping with such difficult environments leads to complex implementations, which are difficult to build, understand, and analyze.

This project addresses these problems using formal modeling and verification techniques, in particular, a combination of Input/Output automaton methods used at MIT and process algebraic and knowledge-based methods used at NTT. This involves extensions to the existing techniques, for example, extending I/O automata to allow dynamic process creation and destruction. As the basic framework is developed, it is being applied to a collection of typical examples from cooperative computing applications, including computer-supported cooperative work, e-commerce, and distributed databases. Other issues being studied include analysis of performance and fault-tolerance properties, and connecting the formal models with actual runnable code.

The project also develops tools to support program analysis and verification using the IOA language, which is a formal language based on the mathematical model of I/O automata.

Progress Through June 2001

IOA Language and Toolset

We have seen a tremendous amount of progress on the IOA project in the past six months.

Dr. Steve Garland has written a new parser and semantic checker for IOA; this tool has been used as the basis for much work on simulation and verification. The new design has enabled Garland and Joshua Tauber to add functionality to the front end tool that will automatically expand composite automata into primitive form.

Andrej Bogdanov has built a tool [B01] that automatically translates IOA programs into Larch Shared Language (LSL) specifications in a style which is suitable for formal reasoning. The framework supports two common strategies for establishing the correctness of distributed algorithms: invariants and simulation relations. The tool has been used to verify three distributed data management algorithms: a strong caching algorithm, a majority voting algorithm and Lamport's replicated state machine algorithm.

Chris Luhrs has been using the Larch Prover (LP), to establish properties of algorithms written in IOA. Using simulation relations and invariant proofs, he has established correctness of a distributed spanning tree algorithm. Through doing this proof, he has tested Bogdanov's IOA to LSL translation tool. In addition, Luhrs is using these experiments to drive future design improvements for LP and create tutorials for future naive LP users.

Laura Dean has led a team improving and re-implementing the IOA Simulator. The chief improvements include adding an interface to Daikon — an invariant-discovery tool — adding new data type implementations, sharing data type implementations with other IOA tools, improving the Simulator's documentation, and improving support for nondeterminism resolution. These improvements increase the IOA Simulator's usability as a tool in writing and verifying algorithms for distributed systems.

In conjunction with Dean's work, Toh Ne Win and Gustavo Santos have been connecting the IOA toolkit to the Daikon dynamic invariant detection tool. Output from the IOA simulator is fed into Daikon, which suggests likely program invariants. Daikon's invariants can then be proven sound by feeding them back into the IOA toolkit and the Larch Prover. The invariants discovered and proved with the help of Daikon can be stepping stones in proving correct other properties of programs or act as useful annotations to help designers of concrete implementations.

Shien Jin Ong is conducting experiments using the simulator to check the correctness of implementations of a strong caching algorithm and Lamport's replicated state machine algorithm. He is leveraging the ability the simulators "paired-simulation" capability to check for errors in proposed simulation relations between implementations and specifications. Ong has used these checks to speed work on and increase confidence in hand proofs of the simulation relations.

Stan Funiak has written a preliminary translator from IOA to Lamport's TLA+ language. TLA+ is the input language of the TLC model-checker. The TLC model checker automatically explores the state space of finite automata to search for counter examples of candidate properties.

Dynamic I/O automata

We have developed a new model which adds structure to IOA to support dynamic systems of the sort that arise in agent-based computing. In [AL01], We present a mathematical state-machine model, the Dynamic I/O Automaton (DIOA) model, for defining and analyzing dynamic systems of interacting components. The systems we consider are dynamic in two senses: (1) components can be created and destroyed as computation proceeds, and (2) the events in which the components may participate may change. The new model admits a notion of external system behavior, based on sets of traces. It also features a parallel composition operator for dynamic systems, which respects external behavior, and a notion of simulation from one dynamic system to another, which can be used to prove that one system implements the other.

The DIOA model was defined to support the analysis of mobile agent systems, in a joint project with researchers at Nippon Telephone and Telegraph. It can also be used for other forms of dynamic systems, such as systems described by means of object-oriented programs, and systems containing services with changing access permissions.

A paper on the DIOA model by Attie and Lynch [AL01] has been accepted for publication in Concur, where it will be presented in August 2001; a short version of this paper will be presented at the 20th ACM Symposium on Principles of Distributed Computing (PODC), also in August 2001.

Building blocks

In the past six months we have continued to work on modeling group communication systems for Wide Area Networks (WANs), with a new emphasis on performance evaluation.

We are currently analyzing the performance of the group communication service (GCS) we presented in [KK00]. One of the key performance characteristics of GCSs is how fast they deliver new views to their clients after network events occur. In our analysis, we consider the situation when a group component stabilizes at some point. We have derived an upper bound on the time by which each client of the component receives the final view after the final network event occurs. The algorithm of [KK00] operates atop an external group membership service. Performance analysis of our GCS is compositional: We have analyzed the VS algorithms alone, in terms of its inputs and timing assumptions. We then expressed reasonable performance guarantees for the group membership service, in terms of the time when the final network event occurred. Finally, we combined the two parts to yield conditional performance properties for the system as a whole. We are working to prepare a paper presenting these results.

In the area of reliable multicast, our work has focused on modeling and analyzing the scalable reliable multicast (SRM) protocol developed by Floyd et. al. SRM is an application level reliable multicast protocol that uses IP multicast as the underlying communication primitive. The core of our work has been to formally model both the reliable multicast service and the SRM protocol and to analyze the performance of the protocol. The specification of the reliable multicast dictates which packets each individual reliable multicast
member should receive. Despite all the research in this area, the reliable multicast property is seldom explicitly specified, especially in the case of faults and group membership changes.

Having specified the service, we are now engaging in its performance analysis. We are investigating possible improvements in performance that can be attained by optimizing the SRM protocol to take advantage of packet loss locality within multicast communication sessions. A preliminary abstract promoting this idea has been published in [LKL01].

Our work presenting a new inheritance-based modeling and verification technique has recently been accepted for publication in ACM Transactions on Software Engineering and Methodology (TOSEM) [KKLS01].

In April, we have presented two papers at the IEEE 21st International Conference on Distributed Computing Systems (ICDCS): (1) A framework for building highly available services [FK01]; and (2) An availability study of several different dynamic voting algorithms [IK01].

Plans for the Next Six Months

The IOA language and underlying model has already proved to be useful as a foundation for agent system modeling and verification, in collaborative projects between our group and NTT. We would like to continue developing the IOA toolset so that it can provide better support for this and related efforts. In particular, we would like to add composition features to the front end; improve the quality and usability of the IOA simulator; experiment further with instrumenting the simulator for use with Prof. Ernst's Daikon invariant discovery tool; improve the quality and usability of the connection with the theorem-prover LP; and perhaps connect IOA to other interactive theorem-proving tools and other automatic validation tools.

The mathematical foundations behind IOA require more work. The DIOA extension to the I/O automaton model, which provides support for dynamic process creation and destruction and for dynamic changes of process capabilities, is not quite as simple and elegant as we would like; we would like to improve it. Extensions of the underlying model and the IOA language itself for timing, hybrid (continuous/discrete), and probabilistic behavior are also interesting and important to pursue. Such extensions would expand the applicability of the model, language to domains such as robotics and embedded systems.

In our work on distributed systems building blocks, we plan to focus
on performance and fault-tolerance analysis for group-communication-based system designs. For example, we plan to continue our analysis of the algorithm we presented in [KK00]; we intend to compare the performance characteristics of our GCS service with that of several existing services.

We intend to continue our work on modeling and performance evaluation of reliable multicast. We intend to complete the performance analysis of our caching-based optimization of SRM. We plan to investigate the issue of restricting the receiver buffer size and potentially specify a weaker reliability service that explicitly considers the receiver buffer size. In addition, we intend to study the performance of additional reliable multicast protocols.

Bibliography

[AL01] P. Attie and N. Lynch. Dynamic I/O Automata: a Formal Model for Dynamic Systems. To appear in Concur, August 2001. Also to appear as brief announcement in the 20th ACM Symposium on Principles of Distributed Computing (PODC), August 2001.

[B01] Andrej Bogdanov. Formal verification of simulations between I/O automata. Master of Engineering thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, September 2001.

[FK01] Alan Fekete and Idit Keidar. A Framework for Highly Available Services Based on Group Communication. IEEE 21st International Conference on Distributed Computing Systems Workshops (ICDCS-21W 2001); the International Workshop on Applied Reliable Group Communication (WARGC), April 2001, pp. 57-62.

[IK01] Kyle W. Ingols and Idit Keidar. Availability Study of Dynamic Voting Algorithms. IEEE 21st International Conference on Distributed Computing Systems (ICDCS), April 2001, pp. 247-254.

[KK00] Idit Keidar and Roger Khazan. A Client-Server Approach to Virtually Synchronous Group Multicast: Specifications and Algorithms. IEEE 20th International Conference on Distributed Computing Systems (ICDCS),

April 2000, pp. 344-355. Full version: MIT Lab. for Computer Science Tech. Report MIT-LCS-TR-794, submitted for journal publication.

[KKLS01] Idit Keidar, Roger Khazan, Nancy Lynch, and Alex Shvartsman. An Inheritance-Based Technique for Building Simulation Proofs Incrementally. To appear in ACM Transactions on Software Engineering and Methodology (TOSEM). Previous version in ICSE 2000, pp. 478-487.

[LKL01] Carolos Livadas, Idit Keidar, and Nancy Lynch. Designing a Caching-Based Reliable Multicast Protocol. Fast abstract in the International Conference on Dependable Systems and Networks (DSN) July 1-4, 2001.