CS 7301: Advances in Distributed Computing
Spring 2004
Section 001

List of Projects:

(1) Atreya, Mittal and Garg's algorithm for detecting a locally stable predicate assumes that processes and channels are non-faulty. Also, the algorithm, in the worst case, has unbounded message complexity, which implies that an adversary can force the algorithm to exchange an unbounded number of control messages. Develop an algorithm for detecting a locally stable predicate when one or more processes may fail by crashing. Your algorithm should have *bounded* message complexity and should be as efficient as possible. Prove the correctness of your algorithm and also analyze its performance.

Relevant Publications:

- R. Atreya, N. Mittal and V. K. Garg. Detecting Locally Stable Predicates without Modifying Application Messages. In *Proceedings of the International Conference on Principles of Distributed Systems (OPODIS)*, December 2003.
- K. Marzullo and L. S. Sabel. Efficient Detection of a Class of Stable Properties. Distributed Computing (DC), Volume 8, Number 2, pages 81–91, 1994.
- S. Venkatesan. Reliable Protocols for Distributed Termination Detection. *IEEE Transactions on Reliability*, Volume 38, Number 1, pages 103–110, April 1989.

(2) Atreya, Mittal and Garg's algorithm for detecting a locally stable predicate assumes a static computing system. Develop an algorithm for detecting a locally stable predicate that is more suitable for a mobile computing system consisting of stationary base stations and mobile hosts. Your algorithm should have *bounded* message complexity and should be as efficient as possible. Prove the correctness of your algorithm and analyze its performance.

Relevant Publications:

- R. Atreya, N. Mittal and V. K. Garg. Detecting Locally Stable Predicates without Modifying Application Messages. In *Proceedings of the International Conference on Principles of Distributed Systems (OPODIS)*, December 2003.
- K. Marzullo and L. S. Sabel. Efficient Detection of a Class of Stable Properties. Distributed Computing (DC), Volume 8, Number 2, pages 81–91, 1994.
- Y.-C. Tseng and C.-C. Tan. On Termination Detection Protocols in a Mobile Distributed Computing Environment. *IEEE Transactions on Parallel and Distributed Systems*, Volume 12, Number 6, pages 558-566, June 2001.
- J. Matocha. Distributed Termination Detection in a Mobile Wireless Network. In *Proceedings of the ACM Southeast Regional Conference*, pages 207–213, 1998.

(3) Most predicate detection algorithms assume that it is possible to monitor changes in the values of relevant variables efficiently. This may require the application program to be modified. Develop a mechanism to *transparently* monitor changes in the values of relevant variables so that the application does not need to be modified. You may assume that you have access to the source code of the application and can recompile it if necessary. Your mechanism should impose as little overhead as possible. Measure the overhead imposed by your approach experimentally.
(**Suggestions:** You may modify the compiler or use mmap( ) system call supported by Unix and Linux operating systems.)

Relevant Material:

- The source code of gdb.
- The Unix/Linux man page of mmap( ) system call.

(4) Dijkstra's three-state self-stabilizing algorithm assumes that only one machine can move at any time, which is not very realistic. Analyze the algorithm when two or more machines may move simultaneously. Also, give a bound on the number of steps required for the algorithm to stabilize in this case.

Relevant Publications:

- E. W. Dijkstra. Self Stabilization in spite of Distributed Control. *Communication of the ACM (CACM)*, Volume 17, Number 11, pages 643–644, 1974.
- E. W. Dijkstra. A Belated Proof of Self-Stabilization. *Distributed Computing (DC)*, Volume 1, pages 5–6, 1986.
- N. Mittal and V. K. Garg. A Rigorous Proof of $O(n^2)$ Bound on the Number of Moves for Stabilization of Dijkstra's 3-State Algorithm. *Technical Report TR-PDS-2001-005*, Department of Electrical and Computer Engineering, The University of Texas at Austin, 2001.
- S.Dolev. Self-Stabilization. *The MIT Press*, Cambridge, Massachusetts, London, England, 2000.

(5) Chandra and Toueg use a failure detector satisfying certain properties to capture partial synchrony in the system. This makes it easier for an *application programmer* to a write software that is more portable. However, a *system programmer* still needs to implement the failure detector satisfying those properties. To that end, the programmer may need to make assumptions about the underlying system. For example, there is a bound on message delay which may either be known or unknown and so on. Develop algorithms to implement various failure detector proposed by Chandra and Toueg using such assumptions. Also, implement a library of failure detectors using the algorithms developed.

Relevant Publications:

- T. D. Chandra and S. Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM (JACM)*, Volume 43, Number 2, pages 225–267, March 1996.
- D. Dolev, C. Dwork and L.Stockmeyer. On the Minimal Sychronism needed for Distributed Consensus. *Journal of the ACM (JACM)*, Volume 34, Number 1, pages 77–97, January 1987.
- C. Dwork, N. A. Lynch and L. Stockmeyer. Consensus in the Presence of Partial Synchrony. *Journal of the ACM (JACM)*, Volume 35, Number 2, pages 288–323, April 1988.

(6) Sun and Garcia-Molina propose various schemes for partial-lookup approach to store key-to-entries mapping more efficiently on a given set of servers. They, however, assume that the mapping rarely changes and the number of servers is fixed. Conduct experiments to evaluate performance of the various mapping schemes when entries can be added and deleted and the number of servers can change. Based on your results, propose a mapping scheme more suitable for the dynamic scenario and measure its performance experimentally.

Relevant Publications:

- Q. Sun and H. Garcia-Molina. Partial Lookup Services. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 58–67, Providence, Rhode Island, May 2003.

(7) Aguilera *et al* propose a matching algorithm for content-based subscription based on which they develop an efficient multicast protocol. They, however, assume that each subscription is a conjunction of clauses where each clause is a an elementary predicate involving equality test. Develop efficient matching and multicast algorithms for a more general class of subscriptions. Analyze the performance of your algorithms both qualitatively and quantitatively.

Relevant Publications:

- M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra. Matching Events in a Content-Based Subscription System. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 53–61, Atlanta, Georgia, May 1999.
- G. Banavar, T. D. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 262–272, Austin, Texas, May 1999.