Applications of wait/lock-free protocols to real-time systems

Hans Hansson
Department of Computer Engineering
Mälardalens Högsk., S-721 23 Västerås
E-Mail: hansh@docs.uu.se

Marina Papatriantafilou Philippas Tsigas
Chalmers University of Technology
Department of Computing Sciences
S-41296 Göteborg
Email: (ptrianta, tsigas)@cs.chalmers.se

August 1998

The problem and the main idea of the project

In most real-time systems, concurrency and access to shared resources are controlled by *locking*. A serious disadvantage is that locking may give rise to *priority inversion*, i.e. situations in which a high-priority task has to wait for a low-priority task to release a lock. That is a serious problem, because it can make a task miss its deadline, which, in turn can cause various types of disaster in the system.

The conventional methods to cope with this problem are based on having the kernel of the system dynamically adjust priorities to ensure bounds on the waiting times [15, 20, 21]; however, the problem of waiting due to blocking is still there [15, 20, 21], enforcing serialisation on the processes and causing concurrency bottlenecks and possibly also deadlocks.

It is possible to share data and objects, without using locks. Wait/lock-free interprocess communication/coordination permits access to concurrent objects without the use of locking; thus, it offers guarantees not only regarding priority inversion (it eliminates this problem altogether), but also regarding *efficiency* (by allowing maximum concurrency and thus, low completion times) and *fault-tolerance* (no task is blocked due to a task that crashed while holding a lock). The wait-free condition guarantees progress and completion for every job (of every task) regardless of the execution speeds (and priorities) of the other tasks in the system. *Lock-free* implementations have a more relaxed requirement: locking is ruled out, as in wait-free implementations, but repeated interferences (e.g. under extreme load conditions of the system) may cause a given operation to take longer time to complete. Comparing the two, lock-free methods imply less-overhead (i.e. in memory requirements), while wait-free methods guarantee fully predictable behaviour for every task. The so-far experience with non-locking synchronisation has shown that it can offer significant benefits in practice and —more significantly— it is efficient alternative to lock-based synchronisation methods for managing concurrency and access to shared resources. However, there has been only limited (although very successful and promising) effort to study and apply the lock/wait-free synchronisation ideas on real-time systems and in practice in general.

This project is to continue our recently started collaboration to explore the power and applicability of non-locking synchronisation in real-time systems and applications, by combining the experience of the two research groups in theses areas and the useful feedback from the industrial nodes in the project.

Our aim in this project is to study and apply the lock/wait-free methods (each one where appropriately suited) to derive new protocols and algorithms for real-time systems and show how they can improve the system behaviour, compared with lock-based methods. An important area that we plan to explore is on *real-time operating system kernels*, in particular wait/lock-free implementations of *shared data structures*. By introducing such mechanisms, we expect to achieve:

- elimination of priority inversion and of deadlock
- fault-tolerance (as opposed to locking, which implies that the failure of a process which holds a lock will prevent others from making progress)
- scalability and uniformity between uni-processor and multi-processor systems (as opposed to locking, for which one needs to employ different methods)
- better task execution times and schedulability conditions (since absence of locking increases concurrency and prevents priority inversion)
- no change of the structure of the existing operating system kernel.

Related Research

The first steps of the research on non-locking distributed protocols were made about two decades ago [14]. As expected, such a fundamental problem received much further attention (cf. e.g. [9, 10, 17, 19]), part of the effort being towards classifying system models and communication primitives, according to their non-blocking synchronisation power. According to the classification, there exist objects, which, used in so called *universal constructions*, can implement any other object of any class (cf. e.g. [9, 10]). Universal wait-free and lock-free methods are expensive, so a strong stream of research is towards deriving efficient implementations of specific objects (based on the possibility results implied by the universal constructions). Another stream of research effort involves the introduction of new architectural primitives for lock-free process synchronisation, such as the *transactional memory* [11], which allows programmers to define customised operations to be applied to multiple, independently chosen words of memory.

Actually, the first suggestion for the wait-free approach to real-time communication was first discussed at least two decades ago [22, 23], but was "lost" in the real-time systems community until recently, when it was revived by Kopetz and Reisinger [13], followed by a series of interesting results by Anderson et.al. (including [2, 3, 4, 5]), and the more recent work by Chen and Burns [6], in applications such as automotive system control, real-time database systems, video-conferencing systems and distributed real-time systems.

Project plan

We plan to have two graduate students doing their doctorate research within the project, one at Mälardalens University in Västerås and one at Chalmers University in Göteborg (additionally, we expect the Hard Real-Time Research group at Uppsala University (Andreas Ermedahl) to contribute).

Our aims include regular publications and presence at the important annual international meetings and conferences of the area, namely: the IEEE Real-Time Systems Symposium (RTSS), the Euromicro Conference on Real-Time Systems, the International Conference on Real-Time Computing Systems and Applications (RTCSA), and the Real-Time Technology and Applications Symposium (RTAS).

More precisely, the theme of the work that the two students will get involved in is to explore research advances in the area of lock-free synchronisation and non-blocking concurrent data structures implementation and apply them for gaining in efficiency in the OS kernel level; the target is to develop an OS prototype, with a lock-free (non-blocking) kernel. Key problems that this research will involve are:

- *Snapshot*: i.e. taking an "instantaneous" picture of various independent components in the system, without "freezing" their execution. Snapshot objects are particularly useful and important tools for interprocess communication and coordination. Since they return to the scanner a consistent global state of the system, they provide support for decision algorithms and they are also used to solve a variety of communication and synchronisation problems, e.g., consistency checking in transaction-based systems, distributed debugging, stable property detection (deadlock, stabilisation, termination, etc.), concurrent time-stamping, system monitoring and control, including many real-time applications [16, 18]. We already have some first results on this subject [8], where we build on previous work on the problem and on wait-free implementations with applications in real-time systems [12, 6]; we study lock-based and lock/wait-free solutions both analytically and experimentally (using event-driven simulation) on a real-time system consisting of CAN-bus-connected nodes [1, 24] and we reveal the improvements that can be achieved by using lock-free and, even more, by a new wait-free algorithm.

- *Agreement*: this involves a set of tasks/jobs which are required to reach an irrevocable common decision, even if there are initial differences of opinion about what the decision should be. It is among *the most fundamental* problems in distributed computing, since it lies in the heart of basic problems such as exclusion, resource allocation, symmetry breaking, synchronisation, distributed transaction commit/abort, and, hence, its solutions have direct and important applications in distributed operating systems, control systems and database systems. Moreover, it has been shown [10] that it is a key to the synchronization power of a system. Specific paradigms of the problem that arise in practice in real-time process-control systems [25] include agreement on an estimate of an airplane's altitude based on the readings of multiple altimeters, or carrying out fault diagnosis procedures for some system component —combine individual diagnosis into a common decision about whether or not to replace some component, distributed database systems, among many others. We plan to build on several interesting protocols, derive the required concurrent data structures implementations (linked lists, stacks, queues, etc), implement our new algorithms and study their behaviour, especially from the point of view of how existing features of common architectures in real-time systems can be used to derive more efficient wait-free solutions to problems such as fault-tolerant consistent decision making [10, 17] and how these solutions can be applied and influence the efficiency in these systems, using standard benchmarks for this type of real-time applications ([16, 18]).

The plan of the proposed work involves the following:

- Study of existing real-time operating systems and the architectures for which they are applicable on. The purpose of this is twofold: (a) to identify the data structures whose implementation is a significant factor for the performance of the respective real-time OS (eg run queue for Mach,...) (b) to identify the synchronization capabilities (via the wait-free agreement protocols that they support) of the architectures on top of which these OS are running. These will lead to the identification of the most efficient feasible ways for lock-free implementation of these basic kernel data structures for the respective architectures.

- Gradual incorporation of the new data structure implementations in the existing kernels; testing and evaluation. We believe that it is very important not to build a new OS from scratch, but only modify already existing ones; in this way we will be able first to concentrate on the wait-free/lock-free aspects on real time OS and, second, to measure exactly the improvements; moreover, the expected benefits of the OS may thus be immediately available to the community that is already using it.
- The study and development will throughout the project be guided by case-studies. Initially we will concentrate on using wait-free snapshots in an automotive diagnostics system.

On a more concrete level, the individual roles of the two (cooperating) graduate students will roughly be the following:

- The MdH student will focus on aspects related to the implementation of wait-free techniques in Real-Time kernels. This will, in addition to the actual implementation work, include evaluation and analysis of implementations and applications.
- The Chalmers student will focus on wait-free techniques and algorithms. This includes study of applications and existing kernels, development and analysis of algorithms, as well as evaluation of the use of algorithms.

Milestones

In the initial two year period we expect the following results[1]:

- Year 1:
  1. Report on applicability of wait-free techniques (M+C)
  2. First implementation of snap-shot mechanism in a selected (commercial) real-time kernel (M)
  3. Application study: Wait-free monitoring in the diagnostic system (M+C)
  4. Definition of algorithm for wait-free agreement in real-time (C)
- Year 2:
  1. Selection of an appropriate set of wait-free mechanisms/objects to include in a real-time kernel (M+C)
  2. Evaluation of the wait-free agreement algorithm and the resulting wait-free shared objects (considering real applications) (C)
  3. Implementation of mechanisms in a selected kernel (M)
  4. Evaluation of mechanisms (using realistic data and (possibly) simulation) (M+C)
  5. Application study: a concrete automotive case-study (M+C)
  6. Two licentiate theses (M+C)

[1] In the list of milestones "M" and "C" are used to indicate deliverables mainly involving the Mälardalen and Chalmers graduate student, respectively.

Industrial cooperation

In this project we will cooperate with both application developers and software platform vendors.

Mecel AB in Göteborg (Matts Lindgren) will (possibly together with Volvo) provide us with automotive applications. Initially we intend to study the use of wait-free snapshots to monitor the behaviour of automotive control systems, as a component of a vehicular diagnostics subsystem and/or an automotive "flight recorder".

We are currently discussing cooperation with the real-time kernel vendors (developers) Enea OSE Systems (Lars Österberg) and Volcano Communication Corporation (Antal Rajnak/Ken Tindell). Our ambition is to develop and evaluate prototype wait-free synchronization modules for their kernels (possibly not both).

Resources and preliminary budget

2 graduate students + some mobility support for mutual visits.

References

[1] - Road Vehicles – Interchange of Digital Information – Controller Area Network (CAN) for High Speed Communication. *ISO/DIS 1189*, Feb. 1992.

[2] J.H. Anderson R. Jain and S. Ramamurthy Wait-free object sharing schemes for real-time uniprocessors and multiprocessors. In *Proceedings of the 18th IEEE Real-Time Systems Symposium*, 1997.

[3] J.H. Anderson and S. Ramamurthy A Framework for Implementing Objects and Scheduling Tasks in Lock-Free Real-Time Systems. In *Proceedings of the 17th IEEE Real-Time Systems Symposium*, 1996.

[4] J. Anderson, S. Ramamurthy, and K. Jeffay Real-Time Computing with Lock-Free Shared Objects (Extended Abstract). In *Proceedings of the 16th IEEE Real-Time Systems Symposium*, pp. 28-37, December 1995.

[5] J. Anderson, S. Ramamurthy, M. Moir, and K. Jeffay Lock-Free Transactions for Real-Time Systems. In *Real-Time Database Systems: Issues and Applications*, Kluwer Academic Publishers, pp. 215-234, May 1997.

[6] J. Chen and A. Burns Asynchronous Data Sharing in Muliprocessor Real-Time Systems Using Process Consensus. *10th Euromicro Workshop on Real-Time Systems*, June 1998, also Research report, Department of Computer Science, Un. of York, 1997.

[7] Eric. A. Brewer, Chris N. Dellarocas, Adrian Colbrook, William E. Weihl PROTEUS: A High-Performance Parallel-Architecture Simulator. MIT/LCS/TR-516, September 1991.

[8] A. Ermedahl, H. Hansson, M. Papatriantafilou and Ph. Tsigas Wait-free snapshots in real-time systems: algorithms, correctness and performance. *Fifth International Conference on Real-Time Computing Systems and Applications*, Hiroshima, October 1998.

[9] M. Herlihy A Methodology for Implementing Highly Concurrent Data Objects. *Proceedings of the ACM Symposium on Principles and Practice of Parallel Programming (PPoPP '90)*, pp. 197–206, 1990.

[10] M. Herlihy Wait-Free Synchronization. *ACM Transactions on Programming Languages and Systems*, Vol. 11, No. 1, January 1991, pp. 124-149.

[11] M. Herlihy, J.E.B. Moss Transactional Memory: Architectural Support for Lock-Free Data Structures. *Proceedings of of the 17th IEEE/ACM International Symposium in Computer Architectures (ISCA '93)*, 1993.

[12] L.M. Kirousis, P. Spirakis and Ph. Tsigas Reading Many Variables in One Atomic Operation: Solutions with Linear or Sublinear Complexity. *IEEE Transactions on Parallel and Distributed Systems*, **5**(7), pp. 688-696, July 1994.

[13] H. Kopetz and J. Reisinger The Non-Blocking Write Protocol NBW: A Solution to a Real-Time Synchronization Problem. In *Proceedings of he 14th Real-Time Systems Symposium*, pp. 131-137, 1993.

[14] Leslie Lamport Concurrent Reading and Writing. *Communications of the ACM*, Vol. 20, No. 1, Nov. 1977, pp. 806–811.

[15] J.P. Lehoczky and L. Sha and J.K. Strosnider Aperiodic Responsiveness in Hard Real-Time Environments. In Proceedings of the IEEE Real-Time Systems Symposium, pp. 262–270, 1987.

[16] C.D. Locke, L. Lucas, and J. Goodenough Generic Avionics Software Specification, Technical Report CMU/SEI-90-TR-8, Software Engineering Institute, Carnegie Mellon University, December 1990.

[17] M. Papatriantafilou and Ph. Tsigas Wait-Free Consensus in *In-Phase* Multiprocessor Systems. In *Proceedings of the Seventh IEEE Symposium on Parallel and Distributed Processing (SPDP'95)*, pp. 312-319, IEEE Press, 1995.

[18] C.-S. Peng, K.J. Lin, and C. Boettcher Real-Time Database Benchmark Design for Avionics Systems. In *Proceedings of the First International Workshop on Real-Time Databases: Issues and Applications*, pp. 92-99, March 1996.

[19] S.A. Plotkin Sticky bits and universality of consensus. In *Proceedings of the Eighth ACM Symposium on Principles of Distributed Computing (PODC '89)*, pp. 159-176, 1989.

[20] R. Rajkumar Synchronization in Real-Time Systems – A Priority Inheritance Approach. *Kluwer Academic Publications*, 1991.

[21] L. Sha, R. Rajkumar, and J. P. Lehoczky Priority Inheritance Protocols: An Approach to Real-Time Synchronization. In *IEEE Transactions on Computers*, vol. 39, pp. 1175-1185, Sep. 1990.

[22] P. Sorensen A Methodology for Real-Time System Development. *PhD Thesis*, University of Toronto, 1974.

[23] P. Sorensen and V. Hemacher A Real-Time System design Methodology. *INFOR*, 13(1), Feb 1975, pp.1–18.

[24] K.W. Tindell and H. Hansson and A.J. Wellings Analysing Real-Time Communications: Controller Area Network (CAN). In *Proceedings of IEEE RTSS'94*, pp. 259–263, 1994.

[25] J.H. Wensley, L. Lamport, J. Goldberg, M.W. Green, K.N. Levitt, P.M. Melliar-Smith, R.E. Shostak and C.B. Weinstock SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control. *Proceedings of the IEEE*, Vol. 66, No. 10, October 1978, pp. 1240–1255.