technische universität dortmund

computer science 12

Master Thesis

Verifying Resource Sharing Protocols in Real-Time Operating Systems

Junjie Shi
Dr.-Ing. Kuan-Hsun Chen
Prof. Dr. Jian-Jia Chen
Technische Universität Dortmund
Email: kuan-hsun.chen@tu-dortmund.de
July 16, 2020

To prevent race conditions or data corruptions, concurrently accessing the same resource is prohibited by exploiting mutual exclusion. Several semaphore-based protocols have been developed to provide a deadlock-free resource synchronization and mitigate the priority inversions. In uni-processor systems, Priority Inheritance Protocol (PIP) and Priority Ceiling Protocol (PCP) are widely accepted and utilized. In modern multiprocessor platforms, many resource synchronization protocols have been proposed, e.g., Multiprocessor Priority Ceiling Protocol (MPCP) [5] and Distributed Priority Ceiling Protocol (DPCP) [6] and implemented on various real-time operating systems (RTOSes), e.g., RTEMS [1] and LITMUS^{RT} [2].

To ensure the correctness of such implementations, dedicated corner cases will be designed and deployed to testify whether the system behaves as expected. However, such case-based validation is usually not sufficient, since it is not possible to test over all the sensitive cases empirically. Alternatively, Gadia et al. [3] use Java to model the implementation of Priority Inheritance Protocol on uniprocessor in the RTEMS and validate the model by using Java Pathfinder to exhaustively detect potential data races, deadlocks, and priority inversions. However, the required effort cannot easily scale to validate the other advanced protocols or even for multiprocessor systems.

Towards this, formal verification has been adopted recently. Gu et al. in [4] develop a practical concurrent OS kernel named CertiKOS, and verify its (contextual) functional correctness with a proof assistant Coq. Although some researches continue to provide certified real-time systems on the same vein, we are more interested to verify the existing implementations on off-the-shelf real-time platforms. We can foresee that such an alternative trait is challenging but potentially effective to the relevant researches.

In this thesis, the student is expected to study the existing implementations of DPCP and MPCP on RTEMS and $LITMUS^{RT}$ at first. Afterwards, all the necessary properties of the assumptions (HW and OS) used by these protocols must be defined and modeled. Based on the proposed models, the student has to formulate the properties of MPCP and DPCP that can be validated, e.g., ceiling priority boost and priority based wait queue. Here, we assume the basic components provided in RTOSes and hardware are correct, e.g., priority based scheduler. At the end, the student is supposed to validate/verify how these necessary properties are matched prove the correctness of considered implementation of PIP, PCP, MPCP, and DPCP on RTEMS and LITMUS^{RT}.

Required Skills:

- Knowledge of C and C++ programming
- Knowledge of Real-Time System
- Knowledge of System Programming and
- Architecture knowledge is beneficial

Acquired Skills after the work:

- Knowledge of Resource Synchronization
- Knowledge of Real-Time Operating Systems
- Design, Analysis, Implementation of system software on real-time system, e.g., version control, coding convention, etc.

References

[1] RTEMS. http://www.rtems.org/.

[2] J. M. Calandrino, H. Leontyev, A. Block, U. C. Devi, and J. H. Anderson. LITMUS^{RT}: A testbed for empirically comparing real-time multiprocessor schedulers. In *Real-Time Systems Symposium (RTSS)*, pages 111–126. IEEE, 2006.

[3] S. Gadia, C. Artho, and G. Bloom. Verifying nested lock priority inheritance in RTEMS with java pathfinder. In K. Ogata, M. Lawford, and S. Liu, editors, *Formal Methods and Software Engineering - 18th International Conference on Formal Engineering Methods, ICFEM, year = 2016,*.

[4] R. Gu, Z. Shao, H. Chen, X. N. Wu, J. Kim, V. Sjöberg, and D. Costanzo. Certikos: An extensible architecture for building certified concurrent OS kernels. In *12th USENIX Symposium on OSDI*, 2016.

[5] R. Rajkumar. Real-time synchronization protocols for shared memory multiprocessors. In *Proceedings.,10th International Conference on Distributed Computing Systems*, pages 116 – 123, 1990.

[6] R. Rajkumar, L. Sha, and J. P. Lehoczky. Real-time synchronization protocols for multiprocessors. In *Proceedings of the RTSS*, 1988.