Dual Priority Scheduling is Not Optimal (Artifact)

Pontus Ekberg
Uppsala University, Sweden
pontus.ekberg@it.uu.se

Abstract

In dual priority scheduling, periodic tasks are executed in a fixed-priority manner, but each job has two phases with different priorities. The second phase is entered after a fixed amount of time has passed since the release of the job, at which point the job changes its priority. Dual priority scheduling was introduced by Burns and Wellings in 1993 and was shown to successfully schedule many task sets that are not schedulable with ordinary (single) fixed-priority scheduling. Burns and Wellings conjectured that dual priority scheduling is an optimal scheduling algorithm for synchronous periodic tasks with implicit deadlines on preemptive uniprocessors. The related article presents counterexamples to this conjecture, and to some related conjectures that have since been stated. This artifact verifies the counterexamples by means of exhaustive simulations of vast numbers of configurations.

2012 ACM Subject Classification Computer systems organization → Real-time systems; Software and its engineering → Scheduling

Keywords and phrases Scheduling, real time systems, dual priority

Digital Object Identifier 10.4230/DARTS.5.1.1

Related Article Pontus Ekberg, "Dual Priority Scheduling is Not Optimal", in 31st Euromicro Conference on Real-Time Systems (ECRTS 2019), LIPIcs, Vol. 133, pp. 14:1–14:9, 2019.
`https://dx.doi.org/10.4230/LIPIcs.ECRTS.2019.14`

Related Conference 31st Euromicro Conference on Real-Time Systems (ECRTS 2019), July 9–12, 2019, Stuttgart, Germany

1 Scope

The related article lists three counterexamples (task sets) to previously stated conjectures [1, 2, 3, 4] concerning dual priority scheduling. These are given in the paper in Section 2 as Counterexamples 8, 9, and 10. The correctness of these counterexamples can not reasonably be verified by hand, but are instead checked by the accompanying computer program ("the artifact"). Here are short descriptions about the counterexamples. More details are in the paper.

Counterexample 8. This is a task set that is feasible (utilization ≤ 1), but is not dual priority schedulable with any configuration (a setting of priorities and phase change points). The program iterates through all configurations and verifies that none is schedulable by simulating each schedule until a deadline miss.

Counterexample 9. This is a task set that is dual priority schedulable, but not with any configuration where the phase 1 priorities are RM. The program iterates through all the configurations where phase 1 priorities are RM and verifies that none is schedulable by simulating until a deadline miss. It also verifies that there exists another configuration (hardcoded in the program) that is schedulable. This is verified by simulating until the hyper-period with no deadline miss.

Counterexample 10. This is a task set that is dual priority schedulable when priorities are set to be RM+RM, but for which the FDMS [3] heuristic fails to find schedulable phase change points. This is verified by the program by running the FDMS heuristic and noting that it fails to find

a schedulable configuration, and then simulating an RM+RM configuration (hardcoded in the program) until the hyper-period without a deadline miss.

2 Content

The artifact package includes:

- The (only) source code file `dualpriotest.c`
- A `Makefile`
- A short `README`

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is also available at: `https://github.com/pontusekberg/dualpriotest`.

4 Tested platforms

The program should compile with any standards-compliant C99 compiler. It takes a long time to verify counterexamples 8 and 9, but the program uses very little memory. The author has tested all counterexamples with the following compilers and CPUs.

Tested compilers: GCC 8.2.1/5.5.0/4.9.2 and Clang/LLVM 7.0.1
Tested CPUs: AMD Ryzen 7 1700X, AMD Opteron 2220 SE, and Intel Xeon E5520.

5 License

The artifact is available under the MIT License.

6 MD5 sum of the artifact

7b582798bcdb3bbce0502bd31ecb95df

7 Size of the artifact

6.45 KiB

References

1 A. Burns and A.J. Wellings. DUAL PRIORITY ASSIGNMENT: A Practical Method for Increasing Processor Utilisation. In *Proceedings of the 5th Euromicro Workshop on Real-Time Systems*, pages 48–53, 1993. `doi:10.1109/EMWRT.1993.639052`.

2 Alan Burns. Dual priority scheduling: Is the processor utilisation bound 100%? In *Proceedings of the 1st International Real-Time Scheduling Open Problems Seminar (RTSOPS)*, 2010. URL: `https://www.cs.york.ac.uk/ftpdir/reports/2010/YCS/455/YCS-2010-455.pdf#page=9`.

3 Tristan Fautrel, Laurent George, Joël Goossens, Damien Masson, and Paul Rodriguez. A Practical Sub-Optimal Solution for the Dual Priority Scheduling Problem. In *Proceedings of the 13th International Symposium on Industrial Embedded Systems (SIES)*, June 2018. `doi:10.1109/SIES.2018.8442075`.

4 Laurent George, Joël Goossens, and Damien Masson. Dual Priority and EDF: a closer look. In *Proceedings of the Work-in-Progress Session of 35th Real-Time Systems Symposium (RTSS-WiP)*, December 2014. URL: `https://hal.archives-ouvertes.fr/hal-01217433`.