GLOSSARY

Absolute jitter The difference between the maximum and the minimum start time (relative to the request time) of all instances of a periodic task.

Acceptance test A schedulability test performed at the arrival time of a new task, whose result determines whether the task can be accepted into the system or rejected.

Access protocol A programming scheme that has to be followed by a set of tasks that want to use a shared resource.

Activation A kernel operation that moves a task from a sleeping state to an active state, from where it can be scheduled for execution.

Aperiodic task A type of task that consists of a sequence of identical jobs (instances), activated at irregular intervals.

Arrival rate The average number of jobs requested per unit of time.

Arrival time The time instant at which a job or a task enters the ready queue. It is also called *request time.*

Background scheduling Task-management policy used to execute low-priority tasks in the presence of high-priority tasks. Lower-priority tasks are executed only when no high-priority tasks are active.

Blocking A job is said to be blocked when it has to wait for a job having a lower priority.

Buffer A memory area shared by two or more tasks for exchanging data.

Capacity The maximum amount of time dedicated by a periodic server, in each period, to the execution of a service.

Ceiling Priority level associated with a semaphore or a resource according to an access protocol.

Ceiling blocking A special form of blocking introduced by the Priority Ceiling Protocol.

Channel A logical link through which two or more tasks exchange information by a message-passing mechanism.

Chained blocking A sequence of blocking experienced by a task while attempting to access a set of shared resources.

Clairvoyance An ideal property of a scheduling algorithm that implies the future knowledge of the arrival times of all the tasks that are to be scheduled.

Competitive factor A scheduling algorithm A is said to have a competitive factor φ_A if and only if it can guarantee a cumulative value at least φ_A times the cumulative value achieved by the optimal clairvoyant scheduler.

Completion time The time at which a job ends to execute. It is also called *finishing time.*

Computation time The amount of time required by the processor to execute a job without interruption. It is also called *service time* or *processing time.*

Concurrent processes Processes that overlap in time.

Context A set of data that describes the state of the processor at a particular time, during the execution of a task. Typically the context of a task is the set of values taken by the processor registers at a particular instant.

Context switch A kernel operation consisting in the suspension of the currently executing job for assigning the processor to another ready job (typically the one with the highest priority).

Creation A kernel operation that allocates and initializes all data structures necessary for the management of the object being created (such as task, resource, communication channel, and so on).

Critical instant The time at which the release of a job produces the largest response time.

Critical section A code segment subject to a mutual exclusion.

Critical zone The interval between a critical instant of a job and its corresponding finishing time.

Cumulative value The sum of the task values gained by a scheduling algorithm after executing a task set.

Deadline The time within which a real-time task should complete its execution.

Deadlock A situation in which two or more processes are waiting indefinitely for events that will never occur.

Direct blocking A form of blocking due to the attempt of accessing an exclusive resource, held by another task.

Dispatching A kernel operation consisting in the assignment of the processor to the task having highest priority.

Domino effect A phenomenon in which the arrival of a new task causes all previously guaranteed tasks to miss their deadlines.

Dynamic scheduling A scheduling method in which all active jobs are reordered every time a new job enters the system or a new event occurs.

Event An occurrence that requires a system reaction.

Exceeding time The interval of time in which a job stays active after its deadline. It is also called *tardiness*.

Exclusive resource A shared resource that cannot be accessed by more than one task at a time.

Feasible schedule A schedule in which all real-time tasks are executed within their deadlines and all the other constraints, if any, are met.

Finishing time The time at which a job ends to execute. It is also called *completion time.*

Firm task A task in which each instance must be either guaranteed to complete within its deadline or entirely rejected.

Guarantee A schedulability test that allows to verify whether a task or a set of tasks can complete within the specified timing constraints.

Hard task A task whose instances must be a priori guaranteed to complete within their deadlines.

Hyperperiod The minimum time interval after which the schedule repeats itself. For a set of periodic tasks, it is equal to the least common multiple of all the periods.

Idle state The state in which a task is not active and waits to be activated.

Idle time Time in which the processor does not execute any task.

Instance A particular execution of a task. A single job belonging to the sequence of jobs that characterize a periodic or an aperiodic task.

Interarrival time The time interval between the activation of two consecutive instances of the same task.

Interrupt A timing signal that causes the processor to suspend the execution of its current process and start another process.

Jitter The difference between the start times (relative to the request times) of two or more instances of a periodic task. See also *absolute jitter* and *relative jitter.*

Job A computation in which the operations, in the absence of other activities, are sequentially executed by the processor until completion.

Kernel An operating environment that enables a set of tasks to execute concurrently on a single processor.

Lateness The difference between the finishing time of a task and its deadline ($L = f - d$). Notice that a negative lateness means that a task completed before its deadline.

Laxity The maximum delay that a job can experience after its activation and still complete within its deadline. At the arrival time, the laxity is equal to the relative deadline minus the computation time ($D - C$). It is also called *slack time.*

Lifetime The maximum time that can be represented inside the kernel.

Load Computation time demanded by a task set in an interval, divided by the length of the interval.

Mailbox A communication buffer characterized by a message queue shared between two or more jobs.

Message A set of data, organized in a predetermined format for exchanging information among tasks.

Mutual Exclusion A kernel mechanism that allows to serialize the execution of concurrent tasks on critical sections of code.

Non-preemptive Scheduling A form of scheduling in which jobs, once started, can continuously execute on the processor without interruption.

Optimal algorithm A scheduling algorithm that minimizes some cost function defined over the task set.

Overhead The time required by the processor to manage all internal mechanisms of the operating system, such as queuing jobs and messages, updating kernel data structures, performing context switches, activating interrupt handlers, and so on.

Overload Exceptional load condition on the processor, such that the computation time demanded by the tasks in a certain interval exceeds the available processor time in the same interval.

Period The interval of time between the activation of two consecutive instances of a periodic task.

Periodic task A type of task that consists of a sequence of identical jobs (instances), activated at regular intervals.

Phase The time instant at which a periodic task is activated for the first time, measured with respect to some reference time.

Polling A service technique in which the server periodically examines the requests of its clients.

Port A general intertask communication mechanism based on a message passing scheme.

Precedence graph A directed acyclic graph that describes the precedence relations in a group of tasks.

Precedence constraint Dependency relation between two or more tasks that specifies that a task cannot start executing before the completion of one or more tasks (called *predecessors*).

Predictability An important property of a real-time system that allows to anticipate the consequence of any scheduling decision.

Preemption An operation of the kernel that interrupts the currently executing job and assigns the processor to a more urgent job ready to execute.

Preemptive Scheduling A form of scheduling in which jobs can be interrupted at any time and the processor assigned to more urgent jobs ready to execute.

Priority A number associated with a task and used by the kernel to establish an order of precedence among tasks competing for a common resource.

Priority Inversion A phenomenon for which a task is blocked by a lower-priority task for an unbounded amount of time.

Process A computation in which the operations are executed by the processor one at a time. A process may consist of a sequence of identical jobs, also called instances. The words *process* and *task* are often used as synonyms.

Processing time The amount of time required by the processor to execute a job without interruption. It is also called *computation time* or *service time.*

Program A description of a computation in a formal language, called a Programming Language.

Push-through blocking A form of blocking introduced by the Priority Inheritance and by the Priority Ceiling protocols.

Queue A set of jobs waiting for a given type of resource and ordered according to some parameter.

Relative Jitter The maximum difference between the start times (relative to the request times) of two consecutive instances of a periodic task.

Request time The time instant at which a job or a task requests a service to the processor. It is also called *arrival time.*

Resource Any entity (processor, memory, program, data, and so on) that can be used by tasks to carry on their computation.

Resource constraint Dependency relation among tasks that share a common resource used in exclusive mode.

Response time The time interval between the request time and the finishing time of a job.

Schedulable task set A task set for which there exists a feasible schedule.

Schedule An assignment of tasks to the processor, so that each task is executed until completion.

Scheduling An activity of the kernel that determines the order in which concurrent jobs are executed on a processor.

Semaphore A kernel data structure used to synchronize the execution of concurrent jobs.

Server A kernel process dedicated to the management of a shared resource.

Service time The amount of time required by the processor to execute a job without interruption. It is also called *computation time* or *processing time.*

Shared resource A resource that is accessible by two or more processes.

Slack time The maximum delay that a job can experience after its activation and still complete within its deadline. At the arrival time, the slack is equal to the relative deadline minus the computation time ($D-C$). It is also called *laxity.*

Soft task A task whose instances should be possibly completed within their deadlines, but no serious consequences occur if a deadline is missed.

Sporadic task An aperiodic task characterized by a minimum interarrival time between consecutive instances.

Start time The time at which a job starts executing for the first time.

Starvation A phenomenon for which an active job waits for the processor for an unbounded amount of time.

Static scheduling A method in which all scheduling decisions are precomputed off-line, and jobs are executed in a predetermined fashion, according to a time-driven approach.

Synchronization Any constraint that imposes an order to the operations carried out by two or more concurrent jobs. A synchronization is typically imposed for satisfying precedence or resource constraints.

Tardiness The interval of time in which a job stays active after its deadline. It is also called *exceeding time.*

Task A computation in which the operations are executed by the processor one at a time. A task may consist of a sequence of identical jobs, also called instances. The words *process* and *task* are often used as synonyms.

Task control block A kernel data structure associated with each task containing all the information necessary for task management.

Tick The minimum interval of time that is handled by the kernel. It defines the time resolution and the time unit of the system.

Timeout The time limit specified by a programmer for the completion of an action.

Time-overflow Deadline miss. A situation in which the execution of a job continues after its deadline.

Timesharing A kernel mechanism in which the available time of the processor is divided among all active jobs in time slices of the same length.

Time slice A continuous interval of time in which a job is executed on the processor without interruption.

Utilization factor The fraction of the processor time utilized by a set of periodic tasks.

Utility function A curve that describes the value of a task as a function of its finishing time.

Value A task parameter that describes the relative importance of a task with respect to the other tasks in the system.

Value Density The ratio between the value of a task and its computation time.

REFERENCES

[ABDNB96] P. Ancilotti, G. C. Buttazzo, M. Di Natale, and M. Bizzarri. A flexible tool kit for the development of real-time applications. In *Proceedings of the IEEE Real-time Technology and Application Symposium*, pages 260–262, June 1996.

[ABDNS96] P. Ancilotti, G. C. Buttazzo, M. Di Natale, and M. Spuri. A development environment for real-time applications. *Journal of Software Engineering and Knowledge Engineering*, 6(3):91–99, September 1996.

[ABR+93] N.C. Audsley, A. Burns, M.F. Richardson, K. Tindell, and A.J. Wellings. Applying new scheduling theory to static priority preemptive scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.

[ABRW91] N.C. Audsley, A. Burns, M.F. Richardson, and A.J. Wellings. Hard real-time scheduling: the deadline-monotonic approach. In *Proceedings of Eighth IEEE Workshop on Real-Time Operating Systems and Software*, May 1991.

[ABRW92] N.C. Audsley, A. Burns, M.F. Richardson, and A.J. Wellings. Hard real-time scheduling: The deadline monotonic approach. In *IEEE Workshop on Real-Time Operating Systems*, 1992.

[AL86] L. Alger and J. Lala. Real-time operating system for a nuclear power plant computer. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1986.

[AS88] R. J. Anderson and M. W. Spong. Hybrid impedance control of robotic manipulators. *IEEE Journal of Robotics and Automation*, 4(5), October 1988.

[B+93] J. Blazewicz et al. *Scheduling in Computer and Manifacturing Systems*. Springer-Verlag, 1993.

[BAF94] G. C. Buttazzo, B. Allotta, and F. Fanizza. Mousebuster: a robot for catching fast objects. *IEEE Control Systems Magazine*, 14(1):49–56, February 1994.

[Baj88] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):996–1005, August 1988.

[Bak91] T.P. Baker. Stack-based scheduling of real-time processes. *Journal of Real-Time Systems*, 3, 1991.

[BDN93] G.C. Buttazzo and M. Di Natale. Hartik: a hard real-time kernel for programming robot tasks with explicit time constraints and guaranteed execution. In *Proceedings of IEEE International Conference on Robotics and Automation*, May 1993.

[BFR71] P. Bratley, M. Florian, and P. Robillard. Scheduling with earliest start and due date constraints. *Naval Research Quarterly*, 18(4), 1971.

[BH73] Per Brinch Hansen. *Operating System Principles.* Prentice-Hall, 1973.

[BKM+92] S. Baruah, G. Koren, D. Mao, A. Raghunathan B. Mishra, L. Rosier, D. Shasha, and F. Wang. On the competitiveness of on-line real-time task scheduling. *Journal of Real-Time Systems*, 4, 1992.

[BL97] G. Buttazzo and G. Lipari. Scheduling analysis of hybrid real-time task sets. In *Proceedings of the IEEE Euromicro Workshop on Real-Time Systems*, 1997.

[Blo77] Arthur Bloch. *Murphy's Law.* Price/Stern/Sloan Publishers, Los Angeles, California, 1977.

[Blo80] Arthur Bloch. *Murphy's Law Book Two.* Price/Stern/Sloan Publishers, Los Angeles, California, 1980.

[Blo88] Arthur Bloch. *Murphy's Law Book Three.* Price/Stern/Sloan Publishers, Los Angeles, California, 1988.

[BR91] S. Baruah and L.E. Rosier. Limitations concerning on-line scheduling algorithms for overloaded real-time systems. In *Eighth IEEE Workshop on Real-Time Operating Systems and Software*, 1991.

[BRH90] S.K. Baruah, L.E. Rosier, and R.R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Journal of Real-Time Systems*, 2, 1990.

[BS93] G.C. Buttazzo and J. Stankovic. Red: A robust earliest deadline scheduling algorithm. In *Proceedings of Third International Workshop on Responsive Computing Systems*, 1993.

[BS95] G.C. Buttazzo and J. Stankovic. Adding robustness in dynamic preemptive scheduling. In D.S. Fussel and M. Malek, editors, *Responsive Computer Systems: Steps Toward Fault-Tolerant Real-Time Systems*. Kluwer Academic Publishers, 1995.

[BS97] G. Buttazzo and F. Sensini. Deadline assignment methods for soft aperiodic scheduling in hard real-time systems. In *Submitted to IEEE Euromicro Workshop on Real-Time Systems*, 1997.

[BSR88] S. Biyabani, J. Stankovic, and K. Ramamritham. The integration of deadline and criticalness in hard real-time scheduling. In *Proceedings of the IEEE Real-Time Systems Symposium*, 1988.

[But91] G.C. Buttazzo. Harems: Hierarchical architecture for robotics experiments with multiple sensors. In *IEEE Proceedings of the Fifth International Conference on Advanced Robotics ('91 ICAR)*, June 1991.

[But93] G.C. Buttazzo. Hartik: A real-time kernel for robotics applications. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1993.

[But96] G. C. Buttazzo. Real-time issues in advanced robotics applications. In *Proceedings of the 8th IEEE Euromicro Workshop on Real-Time Systems*, pages 77–82, June 1996.

[CC89] H. Chetto and M. Chetto. Some results of the earliest deadline scheduling algorithm. *IEEE Transactions on Software Engineering*, 15(10), 1989.

[CL90] M. Chen and K. Lin. Dynamic priority ceilings: A concurrency control protocol for real-time systems. *Journal of Real-Time Systems*, 2, 1990.

[Cla89] D. Clark. Hic: An operating system for hierarchies of servo loops. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1989.

[CSB90] H. Chetto, M. Silly, and T. Bouchentouf. Dynamic scheduling of real-time tasks under precedence constraints. *Journal of Real-Time Systems*, 2, 1990.

[Cut85] M. R. Cutkosky. *Robot Grasping and Fine Manipulation.* Kluwer Academic Publishers, 1985.

[DB87] P. Dario and G. C. Buttazzo. An anthropomorphic robot finger for investigating artificial tactile perception. *International Journal of Robotics Research*, 6(3):25–48, Fall 1987.

[Der74] M.L. Dertouzos. Control robotics: the procedural control of physical processes. *Information Processing*, 74, 1974.

[Dij68] E. W. Dijkstra. Cooperating sequential processes. In F. Genuys, editor, *Programming Languages.* Academic Press, New York, 1968.

[DRSK89] A. Damm, J. Reisinger, W. Schwabl, and H. Kopetz. The real-time operating system of mars. *Operating System Review*, 23(3):141–157, July 1989.

[DTB93] R.I. Davis, K.W. Tindell, and A. Burns. Scheduling slack time in fixed priority pre-emptive systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1993.

[Foh93] G. Fohler. Realizing changes of operational modes with pre run-time scheduled hard real-time systems. In H. Kopetz and Y. Kakuda, editors, *Responsive Computer Systems*, pages 287–300. Springer-Verlag, 1993.

[Foh95] G. Fohler. Joint scheduling of distributed complex periodic and hard aperiodic tasks in statically scheduled systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 152–161, December 1995.

[GB95] T.M. Ghazalie and T.P. Baker. Aperiodic servers in a deadline scheduling environment. *Journal of Real-Time Systems*, 9, 1995.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.

[GLLK79] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling theory: a survey. *Annals of Discrete Mathematics*, 5, 1979.

[GR91] N. Gehani and K. Ramamritham. Real-time concurrent c: A language for programming dynamic real-time systems. *Journal of Real-Time Systems*, 3, 1991.

[Gra76] R.L. Graham. Bounds on the performance of scheduling algorithms. In *Computer and Job Scheduling Theory*, pages 165–227. John Wiley and Sons, 1976.

[HHPD87] V.P. Holmes, D. Harris, K. Piorkowski, and G. Davidson. Hawk: An operating system kernel for a real–time embedded multiprocessor. Technical report, Sandia National Laboratories, 1987.

[HLC91] J.R. Haritsa, M. Livny, and M.J. Carey. Earliest deadline scheduling for real-time database systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1991.

[Hor74] W. Horn. Some simple scheduling algorithms. *Naval Research Logistics Quarterly*, 21, 1974.

[Jac55] J.R. Jackson. Scheduling a production line to minimize maximum tardiness. Management Science Research Project 43, University of California, Los Angeles, 1955.

[JS93] K. Jeffay and D.L. Stone. Accounting for interrupt handling costs in dynamic priority task systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 212–221, December 1993.

[JSM91] K. Jeffay, D. F. Stanat, and C. U. Martel. On non-preemptive scheduling of periodic and sporadic tasks with varying execution priority. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 129–139, December 1991.

[JSP92] K. Jeffay, D.L. Stone, and D. Poirier. Yartos: Kernel support for efficient, predictable real-time systems. In W. Halang and K. Ramamritham, editors, *Real-Time Programming*, pages 7–12. Pergamon Press, 1992.

[Kar92] R. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? *Information Processing*, 92(1), 1992.

[KB86] O. Khatib and J. Burdick. Motion and force control of robot manipulators. In *Proceedings of IEEE Conference on Robotics and Automation*, 1986.

[KDK+89] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabla, C. Senft, and R. Zainlinger. Distributed fault-tolerant real-time systems: The mars approach. *IEEE Micro*, 9(1), February 1989.

[KIM78] H. Kise, T. Ibaraki, and H. Mine. A solvable case of the one machine scheduling problem with ready and due times. *Operations Research*, 26(1):121–126, 1978.

[KKS89] D.D. Kandlur, D.L. Kiskis, and K.G. Shin. Hartos: A distributed real-time operating system. *Operating System Review*, 23(3), July 1989.

[KS86] E. Kligerman and A. Stoyenko. Real-time euclid: A language for reliable real-time systems. *IEEE Transactions on Software Engineering*, 12(9), September 1986.

[KS92] G. Koren and D. Shasha. D-over: An optimal on-line scheduling algorithm for overloaded real-time systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, 1992.

[L+94] J.W.S. Liu et al. Imprecise computations. In *Proceedings of the IEEE*, January 1994.

[Law73] E.L. Lawler. Optimal sequencing of a single machine subject to precedence constraints. *Managements Science*, 19, 1973.

[Lip97] G. Lipari. Resource constraints among periodic and aperiodic tasks. RETIS LAB, TR-97 01, Scuola Superiore S. Anna, Pisa, Italy, February 1997.

[LK88] I. Lee and R. King. Timix: A distributed real-time kernel for multi-sensor robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1988.

[LKP88] I. Lee, R. King, and R. Paul. Rk: A real-time kernel for a distributed system with predictable response. MS-CIS-88-78, GRASP LAB 155 78, Department of Computer Science, University of Pennsylvania, Philadelphia, PA, October 1988.

[LL73] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery*, 20(1), 1973.

[LLN87] J.W.S. Liu, K.J. Lin, and S. Natarajan. Scheduling real-time, periodic jobs using imprecise results. In *Proceedings of the IEEE Real-Time System Symposium*, December 1987.

[LLS+91] J.W.S. Liu, K. Lin, W. Shih, A. Yu, J. Chung, and W. Zhao. Algorithms for scheduling imprecise calculations. *IEEE Computer*, 24(5), May 1991.

[LNL87] K.J. Lin, S. Natarajan, and J.W.S. Liu. Concord: a system of imprecise computation. In *Proceedings of the 1987 IEEE Compsac*, October 1987.

[Loc86] C.D. Locke. *Best-effort Decision Making for Real-Time Scheduling.* PhD thesis, Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA, 1986.

[LRK77] J.K. Lenstra and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, (5):287–326, 1977.

[LRKB77] J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, (1):343–362, 1977.

[LRT92] J.P. Lehoczky and S. Ramos-Thuel. An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1992.

[LSD89] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1989.

[LSS87] J.P. Lehoczky, L. Sha, and J.K. Strosnider. Enhanced aperiodic responsiveness in hard real-time environments. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1987.

[LTCA89] S.-T. Levi, S.K. Tripathi, S.D. Carson, and A.K. Agrawala. The maruti hard real-time operating system. *Operating System Review*, 23(3), July 1989.

[LW82] J. Leung and J.W. Whitehead. On the complexity of fixed priority scheduling of periodic real-time tasks. *Performance Evaluation*, 2(4), 1982.

[Nat95] Swaminathan Natarajan, editor. *Imprecise and Approximate Computation.* Kluwer Academic Publishers, 1995.

[PS85] J. Peterson and A. Silberschatz. *Operating Systems Concepts.* Addison-Wesley, 1985.

[Raj91] R. Rajkumar. *Synchronization in Real-Time Systems: A Priority Inheritance Approach.* Kluwer Academic Publishers, 1991.

[Rea86] J. Ready. Vrtx: A real-time operating system for embedded microprocessor applications. *IEEE Micro*, August 1986.

[RS84] K. Ramamritham and J.A. Stankovic. Dynamic task scheduling in distributed hard real-time systems. *IEEE Software*, 1(3), July 1984.

[RTL93] S. Ramos-Thuel and J.P. Lehoczky. On-line scheduling of hard deadline aperiodic tasks in fixed-priority systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1993.

[SB94] M. Spuri and G. Buttazzo. Efficient aperiodic service under earliest deadline scheduling. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1994.

[SB96] M. Spuri and G.C. Buttazzo. Scheduling aperiodic tasks in dynamic priority systems. *Journal of Real-Time Systems*, 10(2), 1996.

[SBG86] K. Schwan, W. Bo, and P. Gopinath. A high performance, object-based operating system for real-time robotics application. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1986.

[SBS95] M. Spuri, G.C. Buttazzo, and F. Sensini. Robust aperiodic scheduling under dynamic priority systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1995.

[SGB87] K. Schwan, P. Gopinath, and W. Bo. Chaos–kernel support for objects in the real-time domain. *IEEE Transactions on Computers*, 36(8), August 1987.

[Sha85] S. Shani. *Concepts in Discrete Mathematics.* Camelot Publishing Company, 1985.

[SLCG89] W. Shih, W.S. Liu, J. Chung, and D.W. Gillies. Scheduling tasks with ready times and deadlines to minimize average error. *Operating System Review*, 23(3), July 1989.

[SLR88] L. Sha, J.P. Lehoczky, and R. Rajkumar. Solutions for some practical problems in prioritized preemptive scheduling. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1988.

[SLS95] J. K. Strosnider, J. P. Lehoczky, and L. Sha. The deferrable server algorithm for enhancing aperiodic responsiveness in hard-real-time environments. *IEEE Transactions on Computers*, 4(1), January 1995.

[Spu95] M. Spuri. *Earliest Deadline Scheduling in Real-Time Systems.* PhD thesis, Scuola Superiore S. Anna, Pisa, Italy, 1995.

[SR87] J. Stankovic and K. Ramamritham. The design of the spring kernel. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1987.

[SR88] J. Stankovic and K. Ramamritham, editors. *Tutorial on Hard Real-Time Systems.* IEEE Computer Society Press, 1988.

[SR89] J. Stankovic and K. Ramamritham. The spring kernel: A new paradigm for real-time operating systems. *Operating System Review*, 23(3), July 1989.

[SR90] J.A. Stankovic and K. Ramamritham. What is predictability for real-time systems? *Journal of Real-Time Systems*, 2, 1990.

[SR91] J.A. Stankovic and K. Ramamritham. The spring kernel: a new paradigm for real-time systems. *IEEE Software*, May 1991.

[SRL90] L. Sha, R. Rajkumar, and J.P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9), September 1990.

[SRS93] C. Shen, K. Ramamritham, and J. Stankovic. Resource reclaiming in multiprocessor real-time systems. *IEEE Transactions on Parallel and Distributed Computing*, 4(4):382–397, April 1993.

[SSDNB95] J.A. Stankovic, M. Spuri, M. Di Natale, and G. Buttazzo. Implications of classical scheduling results for real-time systems. *IEEE Computer*, 28(6), June 1995.

[SSL89] B. Sprunt, L. Sha, and J. Lehoczky. Aperiodic task scheduling for hard-real-time systems. *Journal of Real-Time Systems*, 1, July 1989.

[Sta88] J.A. Stankovic. Misconceptions about real-time computing. *IEEE Computer*, 21(10), October 1988.

[SZ92] K. Schwan and H. Zhou. Dynamic scheduling of hard real-time tasks and real-time threads. *IEEE Transactions on Software Engineering*, 18(8):736–748, August 1992.

[TK88] H. Tokuda and M. Kotera. A real-time tool set for the arts kernel. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1988.

[TLS95] T.S. Tia, J.W.S. Liu, and M. Shankar. Algorithms and optimality of scheduling aperiodic requests in fixed-priority preemptive systems. *Journal of Real-Time Systems*, 1995.

[TM89] H. Tokuda and C.W. Mercer. Arts: A distributed real-time kernel. *Operating System Review*, 23(3), July 1989.

[TT89] P. Thambidurai and K.S. Trivedi. Transient overloads in fault-tolerant real-time systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1989.

[TWW87] H. Tokuda, J. Wendorf, and H. Wang. Implementation of a time-driven scheduler for real-time operating systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1987.

[Whi85] D. E. Whitney. Historical perspective and state of the art in robot force control. In *Proceedings of IEEE Conference on Robotics and Automation*, 1985.

[WR91] E. Walden and C.V. Ravishankar. Algorithms for real-time scheduling problems. Technical report, University of Michigan, Department of Electrical Engineering and Computer Science, Michigan (USA), April 1991.

[Zlo93] G. Zlokapa. Real-time systems: Well-timed scheduling and scheduling with precedence constraints. Ph.D. thesis, CS-TR 93 51, Department of Computer Science, University of Massachusetts, Amherst, MA, February 1993.

INDEX

A

Absolute Finishing Jitter, 80
Absolute Release Jitter, 79
Accidents, 2
Actuators, 304
Ada language, 19
Adversary argument, 235
Aperiodic service
  Background scheduling, 110
  Deferrable Server, 116
  Dynamic Priority Exchange, 150
  Dynamic Sporadic Server, 155
  EDL server, 163
  IPE server, 168
  Polling Server, 112
  Priority Exchange, 125
  Slack Stealer, 138
  Sporadic Server, 132
  TB* server, 171
  Total Bandwidth Server, 160
Aperiodic task, 27, 51
Applications, 1, 301
Arrival time, 26
ARTS, 325, 340
Assembly language, 2
Asynchronous communication, 290
Audsley, 92, 98
Autonomous system, 307
Average response time, 7–8, 11

B

Background scheduling, 110
Baker, 208
Baruah, 102, 228, 234, 241
Best-effort, 38
Biyabani, 227
Blocking factor, 194
Blocking, 181
Bouchentouf, 71
Braking control system, 307
Bratley, 64
Burns, 140
Busy period, 103
Busy wait, 16–18
Buttazzo, 150, 160, 229, 245

C

CAB, 291
Cache, 13
Carey, 227
Ceiling blocking, 204
Ceiling, 201
Chained blocking, 199
CHAOS, 325
Chen, 185
Chetto, 71, 163
Chorus, 323
Clairvoyant scheduler, 35
Clairvoyant scheduling, 234
Clark, 291
Communication channel, 291
Competitive factor, 234
Complete schedule, 63
Completion time, 27
Computation time, 27

Concurrency control protocols, 221
Context switch, 24, 255
Control applications, 301
Control loops, 301
Cost function, 40
Critical instant, 79
Critical section, 31, 181
Critical time zone, 79
Criticalness, 27
Cumulative value, 43, 231
Cyclical Asynchronous Buffers, 291

D

D-over algorithm, 248
D-over, 248
Dashboard, 307
Davis, 140
Deadline Monotonic, 96
Deadline tolerance, 245
Deadline, 8, 27
  firm, 231
  hard, 8
  soft, 8
Deadlock prevention, 201, 205, 216
Deadlock, 200
Deferrable Server, 116
Dertouzos, 57
DICK, 253, 260
Ding, 92
Direct blocking, 188
Directed acyclic graph, 28
Dispatching, 23, 271
DMA, 13
  cycle stealing, 13
  timeslice, 13
Domino effect, 37, 225
Driver, 15
Dynamic Priority Exchange, 150
Dynamic priority servers, 149
  Dynamic Priority Exchange, 150
  Dynamic Sporadic Server, 155
  EDL server, 163
  IPE server, 168
  TB* server, 171
  Total Bandwidth Server, 160
Dynamic scheduling, 35
Dynamic Sporadic Server, 155

E

Earliest Deadline First, 56, 93
Earliest Due Date, 53
EDL server, 163
Eligibility, 335
Empty schedule, 63
Environment, 302
Ethernet, 329, 336
Event, 6, 17
Event-driven scheduling, 109
Exceeding time, 27, 246
Exclusive resource, 31, 181
Execution time, 27
Exhaustive search, 63
Exponential time algorithm, 34

F

Fault tolerance, 12, 326–327
Feasible schedule, 25
Feedback, 304
Finishing time, 27
Firm task, 109, 231
First Come First Served, 111
Fixed-priority servers, 110
  Deferrable Server, 116
  Polling Server, 112
  Priority Exchange, 125
  Slack Stealer, 138
Fohler, 330
Friction, 308

G

Graceful degradation, 230, 245
Graham, 44
Graham's notation, 51
Guarantee mechanism, 36
Guarantee, 243
Gulf War, 3

H

Hard real-time system, 8
Hard task, 8, 26
Haritsa, 227
HARTIK, 291, 325, 345
HARTOS, 325
Heuristic function, 66, 334
Heuristic scheduling, 35
Hierarchical design, 313
Hit Value Ratio, 249
Horn's algorithm, 56
Howell, 102
Hybrid task sets, 109
Hyperperiod, 103

I

Idle state, 256
Idle time, 24
Imprecise computation, 38
Instance, 27
Interarrival time, 109
Interference, 98–99, 172
Interrupt handling, 15
Intertask communication, 289
IPE server, 168

J

Jackson's rule, 53
Jeffay, 62, 102, 299
Jitter, 79
Job, 27

K

Karp, 242
Kernel primitive
  activate, 280
  create, 260
  end_cycle, 281
  end_process, 283
  kill, 283
  sleep, 260
Kernel, 253
Koren, 248

L

Language, 12, 19
Lateness, 27
Latest Deadline First, 68
Lawler, 68
Laxity, 27
Layland, 82
Lehoczky, 92, 116, 125, 138, 186, 201
Leung, 96
Lifetime, 265
Lin, 185
List management, 272
Liu, 82, 142
Livny, 227
Load, 228
Locke, 227

M

Mach, 227
MACH, 324
Mailbox, 290
Maintainability, 12

MARS, 325
Martel, 62
MARUTI, 325
Maximum lateness, 40
Memory management, 19
Message passing, 289
Message, 290
Metrics, 41, 230
Mode change, 330
Motorola, 327
Multimedia, 38
Murphy's Laws, 4
Mutual exclusion, 18, 31, 181, 284

N

Nested critical section, 187
Non-idle scheduling, 62
Non-preemptive scheduling, 62
Non-real-time task, 109
NP-complete, 34
NP-hard, 34

O

Off-line scheduling, 35
On-line guarantee, 36
On-line scheduling, 35
Optimal scheduling, 35
OS9, 323
Overhead, 296
Overload, 225

P

Partial schedule, 63
Patriot missiles, 3
Peak load, 12
Performance, 40, 43, 230
Period, 28, 78
Periodic task, 27, 77
Phase, 28, 78
Polling Server, 112
Polynomial algorithm, 34
Precedence constraints, 28, 68
Precedence graph, 28
Predecessor, 28
Predictability, 12
Preemption level, 209
Preemption, 24
Preemptive scheduling, 35
Priority Ceiling Protocol, 201
Priority Exchange Server, 125
Priority Inheritance Protocol, 186
Priority inversion, 184
Process, 23
Processor demand, 102
Processor utilization factor, 80
Programming language, 12, 19
Pruning, 64
PSOS, 323
PUMA, 336
Push-through blocking, 189

Q

Quality of service, 38
Queue operations
  extract, 274
  insert, 272
Queue, 24
  idle, 256
  ready, 24, 256
  wait, 32, 182, 256

R

Rajkumar, 186, 201
Ramamritham, 65, 226
Ramos-Thuel, 138
Rate Monotonic, 82
Ready queue, 24, 256

Real Time, 4
Receive operation, 290
Reclaiming mechanism, 154, 244
Recovery strategy, 247
Recursion, 20
RED algorithm, 245
Relative Finishing Jitter, 80
Relative Release Jitter, 79
Release time, 77
Residual laxity, 245
Resource access protocol, 181
Resource constraints, 31, 181, 186
Resource reclaiming, 154, 245, 247
Resource, 31, 181
Response time, 79
Richard's anomalies, 44
RK, 325, 336
Robot assembly, 315
Robotic applications, 301
Robust scheduling, 243
Rosier, 102
RT-MACH, 324
RT-UNIX, 324
Running state, 23

S

Schedulable task set, 25
Schedule, 24
  feasible, 25
  preemptive, 25
Scheduling algorithm, 23
  D-over, 248
  Deadline Monotonic, 96
  Earliest Deadline First, 56, 93
  Earliest Due Date, 53
  Horn's algorithm, 56
  Jackson's rule, 53
  Latest Deadline First, 68
  Rate Monotonic, 82
  Robust Earliest Deadline, 245
Scheduling anomalies, 44
Scheduling policy, 23
Scheduling problem, 34
Scheduling, 271
  best effort, 243
  dynamic, 35
  guarantee, 243
  heuristic, 35
  non-preemptive, 35
  off-line, 35
  on-line, 35
  optimal, 35
  preemptive, 35
  robust, 243
  static, 35
Schwan, 227
Search tree, 63
Semaphore Control Block, 262
Semaphore queue, 256
Semaphore, 18, 32, 181, 284
Send operation, 290
Sensory acquisition, 301
Server capacity, 112
Sha, 92, 116, 125, 186, 201
Shankar, 142
Shared resource, 31, 181
Shasha, 248
Signal, 32, 182, 286
Silly, 71
Slack Stealer, 138
Slack time, 27
Sleep state, 260
Soft task, 8, 26
Sporadic Server, 132
Sporadic tasks, 109
Spring algorithm, 66
Spring, 325, 331
Sprunt, 132
Spuri, 150, 160, 185
Stack Resource Policy, 208
Stack sharing, 218
Stanat, 62

Stankovic, 65, 226, 229, 245
Start time, 27
Static scheduling, 35
Stone, 102, 299
Strosnider, 116, 125
Synchronization, 284
Synchronous communication, 289
System call
  activate, 280
  create, 260
  end_cycle, 281
  end_process, 283
  kill, 283
  sleep, 260
System ceiling, 212
System tick, 265

T

Tactile exploration, 317
Tardiness, 27, 246
Task Control Block, 261
Task instance, 27
Task states, 256
  delay, 256
  idle, 256
  ready, 256
  receive, 257
  running, 256
  sleep, 260
  waiting, 256
  zombie, 258
Task, 23
  active, 23
  firm, 231
  ready, 23
  running, 23
TDMA, 329
Thambidurai, 227
Tia, 142
Tick, 265
Time resolution, 265
Time slice, 25
Time, 4
Time-driven scheduling, 109
Time-overflow, 94–95, 267
Timeliness, 12
Timer interrupt, 266
Timing constraints, 26
TIMIX, 325
Tindell, 140
Total Bandwidth Server, 160
Transitive inheritance, 190
Trivedi, 227
Turing machine, 34

U

UNIX, 324
Utility function, 43, 230
Utilization factor, 80

V

Value density, 230, 242
Value, 27, 230
Vehicle, 307
VRTX32, 323
VxWorks, 221, 323–324

W

Wait, 32, 182, 285
Waiting state, 32, 182
Whitehead, 96
Workload, 228
Worst-case scenario, 36

Y

YARTOS, 325

Z

Zhou, 227
Zlokapa, 227
Zombie state, 258

Giorgio C. Buttazzo graduated in 1985 in Electronic Engineering at the University of Pisa (Italy) and in 1987 received a M.S. degree where he also worked on active perception and real-time control at the G.R.A.S.P. (General Robotics and Active Sensory Processing) Laboratory of the University of Pennsylvania. In 1991, he received a Ph.D. degree in robotics at the Scuola Superiore S. Anna of Pisa. He is currently Assistant Professor of Computer Engineering at the Scuola Superiore S. Anna of Pisa. His main research areas include real-time computing, dynamic scheduling algorithms, sensor-based control, advanced robotics, and neural networks.