Final Report for

OFFICE OF NAVAL RESEARCH

on

Equipment Grant #N00014-87-G-0260

SPRINGNET: A Network of Multiprocessors for Hard Real-Time

by

John A. Stankovic

and

Krithi Ramamritham

October 1988

1. Equipment Purchased

	Item	Manufacturer	Cost
1	Sys1132NY021 (Model 1132 Computer System)	Motorola	$16,875.00
2	MVME136A (68020 based CPU board)	Motorola	8,992.50
1	M68NNTBV68M Unix 5.3	Motorola	1,258.00
1	M68NNTBVNEM (Network Extension (RFS))	Motorola	1,085.00
1	MVME330-2-k-5 (Ethernet Lan Controller, 68010 based)	Motorola	2,100.00
10	IC (M27128A-2F1)	Jameco Electronics	62.50
6	IC	Jameco Electronics	80.94
6	IC (AMZ732A-2DC)	Jameco Electronics	25.50
10	IC EPROMs	Jameco Electronics	134.90
	Total		$30,614.34

Changes from the original request include:

- buying only two processor boards instead of four because of an increase in price,
- not buying the memory boards because memory is now included on the processor board, and
- purchasing IC chips. These chips were needed in order for us to copy one of the processor board chips called "BUG" that provides monitoring and board initialization. This was a minor expense totaling only $303.84.

2. Use of Equipment

The Spring Kernel is a research oriented kernel that contains a new paradigm for real-time operating systems. Simulations have shown the efficacy of its ideas. Our research includes implementing the Spring Kernel on a hardware testbed to validate the simulations and to address the myriad of issues typically glossed over in a simulation. This report briefly indicates the status of the hardware, the testbed development and planned research.

By combining this ONR equipment money with money from NSF and State matching funds, we now have in place 3 nodes of a distributed system and a total of 7 processors. We also have a network bridge, a local ethernet, and a separate development environment of

3 microVaxes. The separate development environment enables us to isolate the SpringNet testbed when testing. Recently, we were awarded a new NSF equipment grant and plan further hardware expansion of the network. This will involve adding 2 more nodes and 10-14 processors, resulting in a 5-node distributed system and approximately 20 processors. We have also been working with Ready Systems in obtaining source code for VRTX and RTSCOPE, two of their products. We will use this software to perform rapid prototyping.

In parallel with studying, purchasing, waiting, implementing and debugging the hardware testbed, siginifcant additional design has been performed on the Spring Kernel. Most importantly, multiprocessor issues have been further developed and scheduling algorithm optimizations have been designed. Note that constructing a hardware testbed from new and state-of-the-art components is not straightforward. For example, we were delayed by at least 3 major errors found in (1) the firmware on DEC bridges, (2) the firmware on the ethernet boards, and (3) in the firmware of the board monitor of the 68020. Finally, actual development and implementation of the first phase of the Spring Kernel, the research to be performed on the testbed, is now underway.

The Spring project has been very active in multiple areas of real-time computing. We have developed significant results in many aspects of real-time scheduling, in real-time operating system design, in real-time transactions, and in time constrained communication protocols. We have also made progress in implementing a software (simulation) testbed, in developing a hardware operating system kernel testbed called SpringNet, and in implementing real-time transactions on a hardware testbed called CARAT. We have also begun a substantial research effort concerning dependable real-time systems. In the following sections we provide a brief overview of the status and plans for the Spring project as a whole.

3. Current Work

3.1 Scheduling and a Software Testbed

Good scheduling algorithms form an essential component of operating systems underlying time-critical applications. Most tasks in time-critical applications have timing constraints such as deadlines or have a need for periodic execution. In addition, tasks typically have criticalness (level of importance), resources requirements, precedence constraints, and placement or affinity constraints.

One of the key notions of our scheme is the notion of *guarantee*. Our scheduling algorithm is designed so that as soon as a task arrives, the algorithm attempts to *guarantee* the task. The guarantee means that barring failures and the arrival of higher criticalness tasks, this task will execute by its deadline, and that all previously guaranteed tasks with equal or higher criticalness will also still meet their deadlines. This notion of *guarantee* underlies our approach to dynamic scheduling and distinguishes our work from other scheduling

schemes. It is also one of the major ingredients for developing flexible, maintainable, predictable, and reliable real-time systems - our major goal.

We began our explorations into specific scheduling algorithms by developing algorithms for scheduling simple tasks and then progressively extended our algorithms to deal with tasks having more complex structures and requirements. Following this approach we have developed a number of variants of our scheduling algorithms, the differences between the variants arising from the factors they take into account. In the process we have developed a software simulation testbed to evaluate these various algorithms. The testbed is continually being improved to include greater functionality and greater modularity. A goal that we have is to make the testbed suitable for distribution to other researchers in real-time systems. The software and hardware testbeds will be coordinated so that they can validate each other.

In the basic version of our scheduling algorithm, only timing constraints, i.e., tasks' computation times and deadlines were taken into account [5]. We considered both periodic tasks and nonperiodic tasks. After evaluating this algorithm [10] [11], we extended it to handle, among other things, resource requirements of tasks. This is a significant accomplishment because handling resources is a complicated problem ignored by most researchers. Our work as described in [21] presents a non-preemptive algorithm for guaranteeing tasks that have deadlines and need resources in exclusive mode. In [24], we consider the situation where resources can be used in both shared as well as exclusive modes. Preemptive scheduling on a node is the subject of [22]. Extensions to the scheme for cooperation among nodes to explicitly handle the presence of resources are discussed in [18], [6], and [19].

In parallel with the extensions involving resource constraints, we considered extensions to the basic algorithm to include precedence constraints among tasks. In our approach [3,4], a task consisting of subtasks related by precedence constraints is scheduled in an atomic fashion. We assume that the computation costs of subtasks as well as the communication costs between subtasks are known when a task arrives at a node. Nodes attempt, in parallel, to schedule subtasks within the constraints imposed by precedence relationships; thus, once guaranteed, subtasks can be executed in parallel at different nodes. [4] reports on evaluation of this scheduling strategy as well as its efficacy in different situations.

We have also evaluated two algorithms which integrate both deadline constraints and criticalness factors in making sceduling decisions [1,2]. Any realistic scheduling algorithm must consider the importance of the tasks along with their timing constraints.

New simulations have also been performed to study the value of our algorithms with two distinct types of multiprocessor architectures [9]. These results have affected the design of the multiprocessor nodes of SpringNet – our distributed systems testbed. We have also worked out many optimizations for our basic algorithms and have begun testing them on the software testbed. The most effective of these will be implemented in the Spring kernel.

In summary, our work on scheduling continues, as we seek integrated solutions that take

into account the complex characteristics of tasks in time-critical systems and the nature of resources that these tasks require.

3.2 Operating System Support and a Hardware Testbed

Next generation hard real-time systems require greater flexibility and predictability than is commonly found in today's systems. These future systems include the space station, integrated vision/robotics/AI systems, collections of humans/robots coordinating to achieve common objectives (usually in hazardous environments such as undersea exploration or chemical plants), and various command and control applications. The Spring kernel [14] is a research oriented kernel, based on a new OS paradigm that is designed to form the basis of a flexible, hard real-time operating system for such applications. Our unique scheduling approach provides for an on-line, dynamic guarantee of deadlines for essential tasks. Essential tasks have deadlines and are important for the operation of the system, but they do not cause catastrophies if their deadlines are missed. Note that critical tasks that do cause catastropies are treated (guaranteed) separately (primarily by preallocating resources). Our dynamic approach results in many benefits which have been detailed in our various papers. The main contributions in our approach are the scheduling algorithms themselves, the design of the kernel that enables predictability of execution time, and the synergism between the scheduling algorithm and the kernel design. The Spring kernel is being implemented on a network of multiprocessors (68020 based) called SpringNet. The hardware consisting of a 3 node system (and a total of 7 processors) is now in place.

3.3 Real-Time Transactions and CARAT

We have developed an integrated approach for real-time transactions based on locking. By an integrated approach we mean that we consider the value of competing transactions and their deadlines in the concurrency control protocol, in the deadlock resolution scheme, (we have both deadlock detection and deadlock prevention algorithms), in the CPU priority, in the recovery strategy, and in I/O scheduling. All this has been implemented on the CARAT testbed. This testbed is a completely functioning distributed database testbed consisting of 5 microVaxes. (None of this equipment was supported by ONR.) We are currently in the process of obtaining performance data for these new real-time protocols. We have also developed some ideas on how to use optimistic concurrency control rather than locking in supporting real-time transactions. This has not yet been implemented. This section was included in this report only for completeness and was not supported by ONR.

3.4 Communication Protocols

Distributed real–time systems will have multiple, distributed tasks cooperating to achieve their goals. An important issue in having such tasks satisfy their timing constraints is the ability to deliver messages on-time. Such message transmissions must be integrated with scheduling. This is one of the major open problem areas for distributed, real–time systems referred to as the *end-to-end* problem. For example, process A might want to communicate with process B which is physically remote. All the steps involved in this communication, namely running process A, executing the send message primitive, invoking the OS, physically transferring the message, receiving the message, invoking the receiving task, processing the message, and replying must be accomplished within a deadline. This requires an integrated scheduling and resource allocation policy.

There are at least two broad ways of dealing with scheduling in the presence of message delays. The first is based on utilizing information about the maximum delay that a message will encounter [8]. Thus, if the nodes in a distributed time-critical system are connected by a local area network and the channel access protocol is designed to guarantee message delivery within bounded time then communicating tasks can be scheduled assuming bounded message delivery delays. The second method to deal with scheduling tasks in the presence of message delays is to compute a deadline for each message delivery from the deadline requirements of the tasks and then employ a communication protocol that transmits messages so that they are delivered before their deadlines. We have developed and evaluated two new classes of protocols termed *virtual time CSMA protocols* [20], [23] and time constrained window protocols [24]. It still remains to better integrate these new protocols with the scheduling algorithm.

4. Planned Work

Even though we believe that we have made a number of substantial contributions in the area of scheduling in time-critical systems, a number of problems still remain. These include:

- Developing integrated scheduling schemes for nonperiodic tasks which have deadlines, resource requirements, criticalness, precedence constraints, and placement constraints.
- Scheduling such complex nonperiodic tasks in the presence of complex periodic tasks.
- Coping with resources other than those on individual nodes, in particular, the communication subnet.
- Scheduling tasks with precedence constraints on multiple nodes; in an end-to-end scheduling scheme, the scheduling of these tasks will have to be done in conjunction with the scheduling of messages along the communication subnet.

- Stragegies for scheduling tasks with a wide spectrum of timing constraints, i.e., where tasks have a large range of deadlines.
- Scheduling schemes for soft real-time tasks coexisting with hard real-time tasks.

We plan to study each of these scheduling problems. We also plan to complete the implementation of the Spring kernel and then test the various scheduling algorithms in the SpringNet environment. For real-time transactions we plan to evaluate the locking based real-time transaction protocols we developed and then to fully develop, implement, evaluate and compare an optimistic concurrency control approach for real-time transactions. We also intend to improve our time constrained communication protocols, and integrate them with the Spring kernel and with the scheduling algorithms. In addition, we will develop a real-time virtual circuit which will be more suitable to hard real-time scheduling than the time constrained communication protocols developed to date.

REFERENCES

[1] S. Biyabani, "The Integration of Deadlines and Criticalness in Hard Real–Time Scheduling," Masters Thesis, Univ. of Mass., August 1988.

[2] S. Biyabani, J. Stankovic, K. Ramamritham, "The Integration of Deadlineand Criticalness in Hard Real–Time Scheduling," *Proc Real-Time Systems Symposium*, Dec. 1988.

[3] S. Cheng, J.A. Stankovic, and K. Ramamritham, "Dynamic Scheduling of Groups of Tasks with Precedence Constraints in Distributed Hard Real-Time Systems," *Proc. Real-Time Systems Symposium*, Dec 1986.

[4] S. Cheng, "Dynamic Scheduling Algorithms for Distributed Hard Real-Time Systems," *Ph.D. Thesis*, University of Massachusetts, May 1987.

[5] K. Ramamritham and J.A. Stankovic, "Dynamic Task Scheduling in Hard Real-Time Distributed Systems," *IEEE Software*, pp. 65-75, July 1984.

[6] K. Ramamritham, J.A. Stankovic, and W. Zhao, "Distributed Scheduling of Tasks with Deadlines and Resource Requirements," submitted to *IEEE Trans. on Computers*, Oct. 1988.

[7] K. Ramamritham, J. Stankovic, W. Zhao, "Meta-Level Control in Distributed Real-Time Systems," *Int. Conf. on Distributed Computing Systems*, Sept. 1987.

[8] K. Ramamritham, "Channel Characteristics in Local Area Hard Real-Time Systems," *ISDN and Computer Networks*, 1987.

[9] P. Shiah, "Real-Time Multiprocessor Scheduling," Masters Thesis, in preparation.

[10] J.A. Stankovic, "Stability and Distributed Scheduling Algorithms," *IEEE Trans. on Software Engineering*, Vol SE-11, No. 10, Oct 1985.

[11] J.A. Stankovic, K. Ramamritham, and S. Cheng, "Evaluation of a Flexible Task Scheduling Algorithm for Distributed Hard Real-Time Systems," Special Issue on Distributed Computing, *IEEE Transactions on Computers*, pp. 1130-1143, December 1985.

[12] J.A. Stankovic and L. Sha, "The Principle of Segmentation," Technical Report, 1987.

[13] J.A. Stankovic, "Decentralized Decision Making for Task Allocation in a Hard Real-Time System," to appear in *IEEE Transactions on Computers*.

[14] J.A. Stankovic and K. Ramamritham, "The Design of the Spring Kernel," *Proc Real-Time Systems Symposium*, Dec. 1987.

[15] J. A. Stankovic and W. Zhao, "On Real-Time Transactions," *ACM SIGMOD Record*, 1988.

[16] J. A. Stankovic, "Misconceptions of Real-Time Computing: A Serious Problem for Next Generation Systems," *IEEE Computer*, October 1988.

[17] J. A. Stankovic and K. Ramamritham, *Hard Real-Time Systems*, Tutorial Text, IEEE Press, 1988.

[18] W. Zhao and K. Ramamritham, "Distributed Scheduling Using Bidding and Focussed Addressing," *Proc. Real-Time Systems Symposium*, pp. 103-111, Dec 1985.

[19] W. Zhao, "A Heuristic Approach to Scheduling with Resource Requirements in Distributed Systems," *Ph.D. Thesis*, Feb 1986.

[20] W. Zhao and K. Ramamritham, "A Virtual-Time CSMA Protocol for Hard Real-Time Communication," *Proc. Real-Time Systems Symposium*, Dec 1986.

[21] W. Zhao, K. Ramamritham, and J. A. Stankovic, "Scheduling Tasks with Resource Requirements in Hard Real-Time Systems," *IEEE Transactions on Software Engineering*, May 1987.

[22] W. Zhao, K. Ramamritham, and J.A. Stankovic, "Preemptive Scheduling under Time and Resource Constraints," *Special Issue on Real-Time Systems, IEEE Transactions on Computers*, Aug 1987.

[23] W. Zhao and K. Ramamritham, "Virtual Time CSMA Protocols for Hard Real-Time Communication," *IEEE Transactions on Software Engineering*, 1987.

[24] W. Zhao and K. Ramamritham, "Simple and Integrated Heuristic Algorithms for Scheduling Tasks with Time and Resource Constraints," *Journal of Systems and Software*, 1987.

[25] W. Zhao, J. Stankovic, K. Ramamritham, "A Multi-Access Window Protocol for Transmission of Time Constrained Messages," *Int. Conf on Distributed Computing Systems*, June 1988.

[26] W. Zhao, J. Stankovic, K. Ramamritham, "A Window Protocol for Transmission of Time Constrained Messages," submitted to *IEEE Transactions on Computers*, Dec. 1987.

[27] W. Zhao, and J. Stankovic, "Performance Analysis of FCFS and Improved FCFS Scheduling Algorithms for Dynamic Real-Time Computer Systems," submitted to *IEEE Transactions on Computers*, March 1988.

[28] Zlokapa, G., "A Multiprocessor Architecture for Real-Time Systems," unpublished memo, University of Massachusetts, May 1985.