

# A Dynamic, Real-Time Testbed for Resource Management Technology

David Chelberg, Lonnie Welch, Cynthia Marling, Carl Bruggeman, Douglas Lawrence, David Matolak, Robert Williams, Jae Lew, Arvind Lakshmikumar, Matthew Gillen, Qiang Zhou  
(chelberg,welch)@ohiou.edu,(marling,bruggema)@p1.cs.ohiou.edu,  
(dal,dmatolak,bobw,jlew,kla)@bobcat.ent.ohiou.edu,(mgillen,qzhou)@p1.cs.ohiou.edu  
Ohio University

Barbara Pfarr  
Barbara.Pfarr@gsfc.nasa.gov  
NASA Goddard Space Flight Center

**Abstract—This paper describes a test-bed for technology that unifies agent based computing and adaptive resource management for dynamic real-time systems. We describe a unified framework that combines a hybrid agent based architecture with explicit resource adapting mechanisms.**

## I. INTRODUCTION

Space Missions of the 21<sup>st</sup> Century will be characterized by constellations of distributed spacecraft, miniaturized sensors and satellites, autonomous mobile robots, increased levels of automation and intelligent on-board processing. New technology will be infused to achieve desired levels of autonomy and processing capability. Agent based computing offers the ability to decentralize computing solutions by incorporating autonomy and intelligence into cooperating, distributed applications. It provides an effective medium for expressing solutions to problems that involve interaction with real-world environments and allows modeling of the world state and its dynamics. This model can be then used to determine how candidate actions affect the world, and how to choose the best from a set of actions. Most agent paradigms overlook real-time requirements and computing resource constraints. For a real-time system, the right answer late is wrong. The effectiveness of agent computing can be enhanced by combining it with system-level resource allocation optimization approaches. Adaptive resource management approaches for real-time systems focus on guaranteeing that real-time requirements are met with a particular set of computing resources, even in dynamically changing environments. The authors are performing research which is unifying, for the first time, the agent based computing paradigm with the theory of adaptive resource management for dynamic real-time systems. This effort will produce a resource manage-

ment model for agent based systems that have real-time constraints. The model will be evaluated by applying it to two ground-based testbed prototypes of future NASA systems: a satellite constellation command and control system, and a group of cooperating autonomous mobile robots. The constellation command and control system is discussed in [5]. This paper represents the robot testbed. The effectiveness of adaptive resource management will be applied to a cooperating group of autonomous mobile robots. Scenarios will involve the tasks of learning, planning, vision and navigation in unknown environments. The tasks will have timing constraints associated with them. A distributed set of computers will be managed dynamically to allow the agents controlling the robots to make the best decisions possible under their resource and timing constraints.

The robot agents will perform the RoboCup Challenge. It represents a standard problem on fast-moving multiple robots, which collaborate to solve dynamic problems [7]. It is designed to meet the need of handling real world complexities. In RoboCup, robots compete in a soccer game. RoboCup can be viewed as a multi-agent system, where software agents provide decision support for dispatching and engaging resources. In such a system, knowledge, action and control are distributed where each agent may cooperate, compete or co-exist. Past approaches [13] have been monolithic in their design, wherein an agent (an individual soccer player) is a local process or a group of processes on a single machine. During a game of soccer, computing needs may increase depending on the state of the game. If these processes cannot migrate to a less loaded processor, a single machine could get overloaded, causing an agent to fail to complete its computations in the desired time. In general, without some means of dynamic resource man-

agement, there is an inefficient allocation of resources. Our approach to achieve this objective can be categorized into two main areas, distributed agent based computing and resource management under dynamic, real-time constraints. This paper is a description of the different aspects involved in our RoboCup testbed.

## II. REAL-TIME SYSTEMS

A real-time system may be characterized as being static or dynamic. A static system is one whose resource requirements do not change with changes in its environment. A dynamic system [18] is one whose resource requirements change unpredictably at runtime. This characterization is based on the temporal properties and execution behavior of the system. The majority of real-time computing research has focused on systems whose requirements are evaluated statically. However, RoboCup is a highly dynamic environment. This precludes the accurate characterization of the environment and its properties by static models. In such contexts, temporal and execution characteristics can only be determined accurately by empirical observation or experience (i.e., a posteriori). In most real-time computing models, the execution time of a job is used to characterize workload statically as an integer worst-case execution time [2], [12], [6]. This is often difficult and sometimes impossible. The DeSiDeRaTa (Dynamic, Scalable, Dependable Real-Time System)[17] middleware was designed to address these shortcomings. One of the fundamental innovations of DeSiDeRaTa is the dynamic path paradigm, that is employed for modeling and resource management of distributed, real-time systems.

To determine the practicality of the resource management model, we need a benchmark suite and environment for experimental assessment of adaptive QoS and resource management software. The benchmark places load on computation and communication resources by execution of software that performs control system functions (e.g, monitoring and evaluating sensor data, or controlling an actuator). Usability requirements dictate that the benchmark application components be dynamically controllable, relocatable and replicable. Traditional computing benchmarks [14], [8], [19], [15] are inadequate for characterizing dynamic distributed real-time systems. Primarily, they were not designed to exhibit behavior characteristic of control systems. Also, they focus on component level benchmarking, and thus they do not

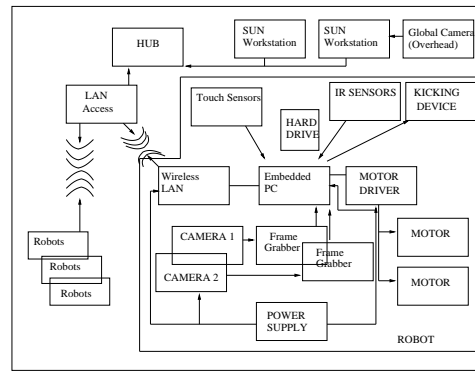


Fig. 1. Block diagram of the entire system

provide the notion of large grain entities with end-to-end timing constraints.

To address these problems, we will use DynBench [16] and RoboCup as benchmarks.

## III. TESTBED

This section gives an overview of the physical and software architecture of robots for the RoboCup challenge problem.

### A. Hardware Design

The physical robot (Figure 1) is made up of an omnidirectional base, a ball-dribbler and a kicker. The control circuitry on board the robot includes a microprocessor, memory and a wireless receiver. Our hardware system can be broadly classified into Mechanical components, Control circuits, Processor and communication devices.

The mechanical design of our robots was developed keeping in mind the agility and speed of the robot. The robots must also be able to kick and control the ball accurately. To provide greater mobility, our robots are equipped with an omnidirectional base. The main features of an omnidirectional system are:

- Ability to move in any direction
- Ability to turn the orientation in any direction

This allows the robots to get from a location  $(x_1, y_1)$  and an orientation  $\Theta_1$  to a new location  $(x_2, y_2)$  and orientation  $\Theta_2$  by following a trajectory while turning continuously. Besides the base, our robot is equipped with a ball kicking mechanism and a dribble bar.

The controls part of the robot can be classified into two main components - the microprocessor and the motor related electronics. Figure 2 shows a schematic of our control system. The player robots do not require a

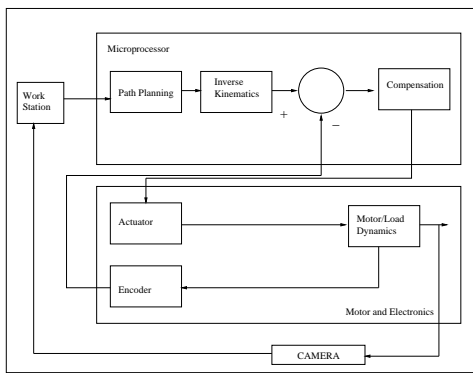


Fig. 2. Control System Block diagram

lot of computational power. Most of the computation is performed on off-board workstation. The off-board computer analyzes the visual information, makes predictions about the player movements and then tells each robot where he is and where he must go. The processing each robot needs to do is:

- Given a final velocity vector, orientation and position, compute and execute the minimum time path from the current velocity vector, orientation and position
- Given a path, follow the path with a given speed. A path is a series of positions/orientations with continuity conditions specified at each point (This is to allow slower speed motions along a well-specified path)
- Keep track of his position, speed and orientation between vision updates
- Decode/process communications. The off-board computer may send messages to specific robots to the effect of *arming* the kicker, or activating it, depending on how the kicker is implemented
- Control the speed of the spinning blade robot kicker

In a robot soccer system, the robots need to communicate with each other and with the base station. This communication needs to be done frequently to help robot share plans and find their goals. Most of these systems use a base station or controller. Wireless LANs allow the robots to transfer information. There are two ways to connect the wireless LAN to the hardware on board. In our case, the processor on board the robots has a PCMCIA interface for the wireless LAN. In some cases, there is a micro-controller on board. For these kind of systems, there are wireless serial modems that connect to the serial port on board. This module provides a lot of avenues

for resource management. Dynamic resource allocation is at the core of high-performance network operations. It involves the real-time management of communication resources, that is, intelligent allocation of frequency channels, spreading-codes and transmission time-slots to network users, in order to share the scarce radio bandwidth efficiently among them. Wireless LANs typically operate on a dedicated bandwidth, which for several of the most popular frequency allocations ranges from 26 to 150 MHz. One of the most popular frequency bands is that centered at 2.4 GHz. The channel capacity can be effectively utilized if the channel allocation is dynamic [4]. This is taken care of by agents that dynamically adjust the characteristics of the data stream to match the changing network conditions.

### B. Software Design

The architecture proposed by us is applicable to both the simulator league and the small sized robot league. The existing RoboCup client models do not adequately take into account the real-time constraints faced by the soccer agents on the field. This leads to an overworked and sub-optimal system. To perform these activities under real-time conditions, an architecture is proposed that puts the entire RoboCup subsystem under the control of a resource manager. This manager monitors the performance of various players on the field and balances the processor load based on their capacities. In the proposed model, we introduce the concept of a *Client manager* that resides in a level between the server and the clients. This client manager, based on the input from the resource manager will decide the resources needed by every client. The client manager communicates with the server using the libraries native to the server and with the clients using DeSiDeRaTa's communication libraries.

Our architecture is a layered one. This is similar to a subsumption architecture [3], [1]. Figure 2 illustrates this architecture and the various layers. Level 0 has only two functions - detect/avoid obstacles and gather statistics. These statistics indicate the number of collisions that occurred and the number of exceptions sent back to level 1. If there are no obstacles, the player continues on his current path. Otherwise, he throws an exception to level 1. Level 1 deals with the factors that affect the action and performance of the client. This includes object tracking and exception handling from level 0. Level 2 pertains to the individual planning adopted by the clients. Some

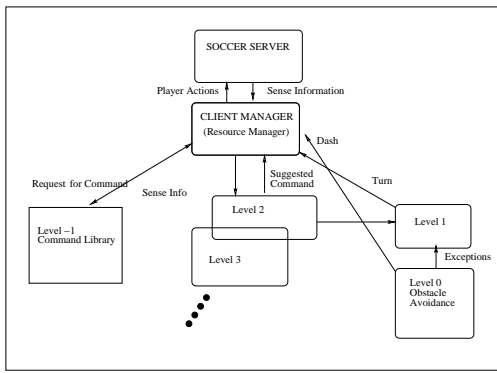


Fig. 3. Layered Architecture

of the actions we identified under this category were the actual strategy planning (offense, defense and goalie), player communication, utility of reasoning and prediction. In level 3, we perform the action evaluation of the clients. This includes modeling the opponent, sharing the knowledge learned and computing the resources needed to reason. All the levels following level 3 would follow the same communication path. The main advantage of this scheme is that the clients may be distributed across different machines. The resource manager would still be managing both the CPU and the network loads.

The objective of a vision system is to capture frames from an overhead camera and extract information from the picture for our strategy module. That involves recognizing objects in the image (like the ball, or another robot) and determining where they are in the world coordinate system. Robot soccer is a difficult environment for image processing. Primarily, it has to be done in real-time so that the game has not changed significantly when the robot is ready to react to the input from the vision system. To meet this demand, processing one frame should take less than  $(1/30)$  of a second. Another issue is the size of the camera. Besides the overhead camera, the goalie robot is equipped with an on board camera. Our on board camera has to be small and light to fit on the robot. Small and light cameras tend to have low resolution and unstable color reproduction. There are two major techniques that allow us to identify the location of objects from an image. They are *edge detection* and *thresholding*. Edge detection [9] recognizes edges in the picture, and takes into consideration a priori knowledge about the shapes of the objects on the field. Edge detection is primarily a technique intended for greyscale images, but can be adapted to color images too. The two

main disadvantages with edge detection techniques are:

- They don't take advantage of the color information for identifying objects
- They are sensitive to noise

Thresholding [10] is readily applied to color images, and it does take advantage of the color information. Objects in an image are distinguished by their color. Pixels with similar colors are grouped together by comparing each pixel in the image to a set of threshold values. Thresholding suits the robot soccer environment well, because different types of objects have different colors. Also, this method is very efficient, since comparing an image to a set of color ranges would be faster than edge detection. The task of the vision system is to supply the strategy module with information about the position of the objects it can see relative to a given robot. After thresholding, the point on each group of pixels closest to the center of the image is identified. This gives us the distance between various points in an image. Since the camera is calibrated, we know the correlation between the distances in the image and the actual distances between objects on the field. This would enable a robot to know the whereabouts of every object on the field. However, if some players are blocking the field landmarks and the ball, estimating the relative position and orientation becomes difficult. In such cases, the players of a team would need to cooperate and belief networks [11] would be used to resolve the uncertainty.

In the case of other sensing strategies, the agent should find the ball and identify the target. Besides vision, typical sensors used in mobile robot research are range finders, sonars, and bumper sensors. However, it is difficult for them to discriminate the ball and the target unless special equipment, such as a transmitter, is set inside the ball or target, or a global positioning system is additionally used to inform the positions of all agents. The simplest case is a global positioning system, without on-board sensing.

#### IV. CONCLUSIONS

Resource management is vital to dynamic systems operating in real-time. This paper describes a testbed being developed for evaluation of resource management middleware. We have described the various modules that constitute the RoboCup system and discussed the applicability of the resource manager to a few of them. Our system will allow assessment of how effectively re-

source managers balance the load between the different machines running the clients so as to ensure real-time constraints and achieve the most efficient allocation of computational resources. The agents being developed by us are fully cognizant of the resources available to them and the constraints of operation. These agents will efficiently be controlled by a meta-agent which manages their time and resources.

## REFERENCES

- [1] Eyal Amir and Pedrito Maynard-Reid II. Logic based subsumption architecture. In *Sixteenth International Joint Conference on Artificial Intelligence*, 1999.
- [2] T.P. Baker. Stack-based scheduling of real-time processes. *Journal of Real-time Systems*, 1991.
- [3] Rodney A. Brooks. A robust layered control system for a mobile robot. Technical report, MIT, 1985.
- [4] D. C. Cox and D. O. Reudink. A comparison of some channel assignment strategies in large-scale mobile communication systems. In *IEEE Transactions on Communications*, volume 20, pages 190–195, April 1972.
- [5] Ryan Detter, Lonnie Welch, Barbara Pfarr, Brett Tjaden, and Eui-Nam Huh. Adaptive management of computing and network resources for spacecraft systems. In *The 3<sup>rd</sup> Annual Military and Aerospace Applications of Programmable Devices and Technologies Conference*, September 2000.
- [6] J.S. Breese E.J. Horvitz and M. Henrion. Decision theory in expert systems and artificial intelligence. *International Journal of Approximate Reasoning*, pages 247–302, 1998.
- [7] Yasuo Kuniyoshi Hiroaki Kitano, Minoru Asada and Itsuki Noda. *RoboCup: A Challenge Problem for AI and Robotics*, volume I of *RoboCup 97: Robot Soccer World Cup*, chapter 1, pages 1–20. Springer-Verlag, 1998.
- [8] H.J.Curnow and B.A.Wichmann. A synthetic benchmark. In *Computer Journal*, volume 19, pages 43–49, January 1976.
- [9] J.Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.
- [10] William K.Pratt. *Digital Image Processing*. Wiley-Interscience, 1991.
- [11] Hongjun Li. An introduction to belief networks. Technical Report TR 99-58, University of Maryland at College Park, 1999.
- [12] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 1973.
- [13] Masayuki Ohta. *Learning Cooperative Behaviors in RoboCup Agents*, volume I of *RoboCup 97: Robot Soccer World Cup*, chapter 8, pages 412–420. Springer-Verlag, 1998.
- [14] R.P.Weicker. Dhrystone: A synthetic systems programming benchmark. In *Communications of the ACM*, volume 27, pages 1013–1030, 1984.
- [15] R.P.Weicker. An overview of common benchmarks. In *IEEE Computer*, volume 23, pages 65–75, December 1990.
- [16] Lonnie Welch and Behrooz A.Shirazi. A dynamic real-time benchmark for assessment of qos and resource management technology. In *The IEEE Real-Time Technology and Applications Symposium*, pages 36–45, June 1999.
- [17] Lonnie Welch, Behrooz Shirazi, Binoy Ravindran, and Carl Bruggeman. Desiderata: Qos management technology for dynamic,scalable,dependable,real-time systems. In *The 15th Symposium on Distributed Computer Control Systems*, pages 7–12, September 1998.
- [18] Lonnie Welch, Behrooz Shirazi, Binoy Ravindran, and Carl Bruggeman. Specifications and modeling of dynamic,distributed,real-time systems. In *The IEEE Real-Time Systems Symposium*, IEEE Computer Society Press, pages 72–81, December 1998.
- [19] W.J.Price. A benchmark tutorial. In *IEEE Micro*, pages 28–43, October 1989.