Umakishore Ramachandran
http://www.cc.gatech.edu/ rama

I received my Ph. D. in Computer Science from the University of Wisconsin, Madison in 1986. Since then I have been with Georgia Tech, where I am currently a Professor in the College of Computing. My research interests are in the area of architectural design, programming, and analysis of parallel and distributed systems. Currently, in the ubiquitous presence project (funded by NSF ITR), I am investigating software and hardware mechanisms for ubiquitous distributed computing for an environment comprised of distributed sensors, embedded data concentrators, and backend clusters, jointly with researchers from electrical engineering and computer science. I received a Presidential Young Investigator (PYI) Award from the National Science Foundation (NSF) in 1990, the Georgia Tech Doctoral Thesis Advisor award in 1993, and the College of Computing Outstanding Senior Research Faculty award in 1996. Ten students have so far received doctorate under my guidance and are in very distinguished positions in industries and academia. I have dozen students working with me half of whom are undergraduates.

Prior to coming to Georgia Tech, I have participated in the design of three distributed operating systems: Charlotte at University of Wisconsin-Madison, Quicksilver at IBM Almaden, and Jasmin at BellCore. At Georgia Tech, I was associated with the Clouds object-based distributed operating system in which we pioneered the idea of using distributed shared memory as a mechanism for object transport and consistency maintenance. Other projects I have initiated and worked on include Beehive (shared address space mechanisms over a network of workstations), TASS (a top-down approach to scalability study of parallel systems), Timepatch (parallel simulation of multiprocessor caches), and RAW (reconfigurable architecture workbench).

Recent Representative Publications:

- U. Ramachandran, R. S. Nikhil, N. Harel, J. M. Rehg, K. Knobe, "Space-Time Memory: A parallel programming abstraction for interactive multimedia applications," **Proc. ACM Principles and Practices of Parallel Programming**, May 1999, Atlanta, Ga.

- K. Knobe, J. M. Rehg, A. Chauhan, R. S. Nikhil, U. Ramachandran, "Scheduling Constrained Dynamic Applications on Clusters," in **Supercomputing '99**, December 1999.

- R. S. Nikhil, U. Ramachandran, "Garbage Collection of Timestamped Data in *Stampede*," in **Principles of Distributed Computing 2000**, July 2000.

- Sameer Adhikari, Arnab Paul, and U. Ramachandran, "D-Stampede: Distributed Programming System for Ubiquitous Computing," (To Appear in) **22nd International Conference on Distributed Computing Systems (ICDCS)**, Vienna, Austria, July, 2002.

Massively Distributed Programming

Umakishore Ramachandran

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
Phone: (404) 894-5136
FAX: (404) 385-2295
e-mail: rama@cc.gatech.edu
WWW URL: http://www.cc.gatech.edu/~rama

Position Paper for Grand Research Challenges in Computer Science and Engineering

Marc Weiser envisioned *ubiquitous computing* as "computing that is an integral and invisible part of the way people live their lives". The explosive growth of technology, the falling price of computing and communication gear, and the permeation of technology into all aspects of our everyday living is giving credence that this vision will become a reality in the not too distant future. That the future of information technology will be dominated by invisible or pervasive computing is a belief that is being shared by several research groups.

In recent times, (having sat on several panels) I see that every researcher and their brothers and sisters are talking about application scenarios in which there are massive numbers of sensors and actuators (some wired, some wireless) cooperating to accomplish tasks that will make a difference in our everyday lives. For example, the *aware home* project at Georgia Tech talks about wiring a home with sensors and actuators so that they can be used for monitoring (in the good sense!) the activities of its inhabitants, and assisting them in their everyday activities (such as cooking, healthcare, and childcare). The research group at AT&T Cambridge talks about location sensing systems with fine accuracy of pin-pointing objects that can then be used for accomplishing higher level goals such as a "follow me" computing environment in an office setting. There have been a number of examples proposed by researchers for wiring the environment and the participants (ranging from humans to animals in the wilderness) with sensors and actuators to enable both training for and dealing with hazardous real-life scenarios such as fire, terrorist situations, and unobtrusive monitoring of animals in their natural settings.

Such complex ubiquitous computing applications, require the acquisition, processing, synthesis, and correlation (often *temporally*) of streaming high bandwidth data such as video and audio, as well as low-bandwidth data such as from a haptic or temperature sensor. Such applications usually span a multitude of devices that are *physically distributed* and *heterogeneous*. Different kinds of sensors (and data aggregators located near them) collect raw data and perhaps do limited processing such as filtering. However, extraction of higher order information content from such raw data requires significantly more processing power. For example, microphones may collect audio data, but higher order processing is needed for voice recognition. Thus there is a continuum of computation and communication resources from tiny computationally limited sensors to high-performance backend clusters. Also such applications are highly dynamic. For example, in a telepresence application participants may join and leave a chat session at different times.

The unique characteristics of this emerging class of massively distributed applications calls for novel solutions. An important and often forgotten challenge is the massively distributed programming problem that such a vision engenders. We present this *distributed programming challenge* as a set of research questions that need investigation:

- **Programming Abstractions.** What are the right programming abstractions that seamlessly transcend the hardware continuum from sensors in the environment to powerful backend clusters in machine rooms? Issues include temporal guarantees for data, efficient support for stream data, and distributed synchronization. The primary goal is to make the development of application programs spanning such a continuum easy. Some of these sensors may be mobile. Many of these sensors may have limited computational capabilities. Yet we may want a uniform programming environment that allows development and deployment of distributed applications spanning this computation and communication resource continuum.

- **Computational Model.** What is the right computational model for dealing with failures? Real failures and perceived failures (due to poor response times) are expected to be normal occurrences in such an environment. Traditional approaches to fault tolerance and recovery may be neither applicable nor scalable to a setting in which there are thousands of sensors and actuators. Fundamentally new computational models are needed that will allow reasoning about software in the presence of failures.

- **Naming and Resource Discovery.** How do we name entities in this massively distributed environment? There have been several recent research ideas at the networking and sensor level to name entities that are not based on static and/or global IDs such as IP addresses. These ideas need to be elevated to the level of programming. Indeed, there may be several layers to this naming problem. For example, a computation may specify that it wants to get video input from the north-west part of campus. Such a high-level programmatic intention has to automagically translate to an appropriate logical *channel* defined by the programming abstraction. The logical channel should translate to the low level address associated with the physical camera that provides this video input.

- **Adaptive Distributed Plumbing.** How do we orchestrate the dynamic connectivity of the components participating in such a massively distributed computation? With advances such as aspect oriented programming, we are just beginning to tackle the problem of plumbing programming components in a dynamic setting. The .NET framework from Microsoft and supporting third party tools are approaching the level of maturity for component-based programming on the internet with the Web as the primary focus. We are not even at the stage of clearly articulating the plumbing problem in massively distributed computations. For one thing, the end point of communication may not always be the same physical device. The camera that was supplying video input in the above naming example may fail. In this case, transparent to the application, the programming system should change the end point of communication to another camera which is available in the vicinity priming the new camera with the state from the old one.

- **Runtime Mechanisms.** What new runtime mechanisms are needed to support such massive distributed programming? To support adaptive plumbing the runtime has to be able to package the state information from a resource constrained or a failed component and move it to another end point. This is just one instance of new runtime mechanisms that may be needed. There may be several new mechanisms needed to support such a programming continuum including program and data transformations, computation elision where applicable, and state reconstruction in the presence of failures.

- **System Evaluation.** How do we evaluate such large-scale systems? There are several axes along which such evaluation needs to be done: ability to meet application level guarantees, availability in the presence of real and perceived failures, and performance and scalability of the runtime mechanisms, While none of these axes are fundamentally new for system evaluation, the scale of future systems poses interesting challenges in coming up with new techniques for carrying them out.