ITR/SY: A Distributed Programming Infrastructure for Integrating Smart Sensors

NSF Program CCR-0121638

PIs: Umakishore Ramachandran
Kenneth Mackenzie
Steve DeWeerth
Irfan Essa
Thad Starner

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
Phone: (404) 894-5136
FAX: (404) 385-6506
e-mail: rama@cc.gatech.edu
WWW URL: http://www.cc.gatech.edu/~rama

Annual Report via Fastlane August 6, 2004

1 Activities and Findings

1.1 Research and Education

The proposed research is integrating sensing hardware, embedded processing and distributed system support to build a seamless programming infrastructure for ubiquitous presence applications. Fundamental invention and integration of techniques spanning programming idioms [18, 2] and runtime systems for distributed sensors, and building blocks for embedded processing are expected as the primary intellectual contributions of the proposed research. Interfacing these technologies to emerging applications on the one end and novel off-the-shelf sensors at the other end are secondary goals of the proposed research.

This subsection details the research accomplishments this past year (2003-04).

1.1.1 Distributed Systems Technologies

Data Fusion Architecture. Simple in-network data aggregation (or fusion) techniques for sensor networks have been the focus of several recent research efforts, but they are insufficient to support advanced fusion applications. We extend these techniques to future sensor networks and ask two related questions: (a) what is the appropriate set of data fusion techniques, and (b) how do we dynamically assign aggregation roles to the nodes of a sensor network. We have developed an architectural framework, DFuse [9, 10], for answering these two questions. It consists of a data fusion API and a distributed algorithm for energy-aware role assignment. The fusion API enables an application to be specified as a coarse-grained dataflow graph, and eases application development and deployment. The role assignment algorithm maps the graph onto the network, and optimally adapts the mapping at run-time using role migration. Experiments on an iPAQ farm show that, the fusion API has low-overhead, and the role assignment algorithm with role migration significantly increases the network lifetime compared to any static assignment.

Middleware Guidelines for Future Sensor Networks. In the near future, we envision sensor networks to transport high bandwidth, low latency streaming data from a variety of sources (such as cameras and microphones). Sensor networks will be called upon to perform sophisticated in-network processing such as fusion of images, and tracking objects. It is not too difficult to imagine that computational capabilities of network nodes will scale up relative to the fairly limited resources of current motes. However, it is likely that energy will continue to remain a constrained resource in such futuristic sensor networks. Recently, there have been proposals for middleware that provide capabilities for higher-level in-network processing while minimizing energy drain on the network. In this work [21], we analyze the interplay between resource requirements for such compute and communication intensive in-network processing and the resultant implications on the figures of merit of interest to an application including latency, throughput, and lifetime. The workload used is a surveillance application. Middleware capabilities include data fusion, role migration (simple relaying versus in-network processing), and prefetching. Through a simulation-based study, we shed light on the impact of device characteristics such as CPU speed and radio features on application figures of merit. We show, in the presence of prefetching, that radio bandwidth above a threshold may not impact latency for compute-intensive workloads and that the network lifetime is virtually the same irrespective of the radiosŠ power saving mode. We also show that a simple minded cost function may not be sufficient to guide migration decisions in the middleware.

Media Broker. MediaBroker [14] is a distributed framework designed to support pervasive computing applications. Specifically, the architecture consists of a transport engine and peripheral clients and addresses issues in scalability, data sharing, data transformation and platform heterogeneity. Key features of MediaBroker are a type-aware data transport that is capable of dynamically transforming data en route from source to sinks; an extensible system for describing types of streaming data; and the interaction between the transformation engine and the type system. Details of the MediaBroker architecture and implementation are presented in this paper. Through experimental study, we show reasonable performance for selected streaming media-intensive applications. For example, relative to baseline TCP performance, MediaBroker incurs under 11% latency overhead and achieves roughly 80% of the TCP throughput when streaming items larger than 100 KB across our infrastructure.

Reasoning About Time, Location, and Identity in Distributed Pervasive Computing. The pervasiveness of computing is creating opportunities for new kinds of applications. However, the software infrastructure for developing complex pervasive computing applications is far from mature. Examples of complex pervasive computing applications include surveillance, traffic management and mobile commerce. These applications, though seemingly different, have some common requirements from the software infrastructure. Components of the application are physically distributed over space, with all the attendant needs of distributed programming. The components are also distributed over time; thus application level decisions are influenced by live data as well as historical data. The components are temporally dynamic in that the participating entities change constantly over time or an entity may participate in discrete intervals rather than a continuous interval. The components are spatially dynamic in that the participating entities may be mobile or they may change their behavior based on location. The application components that are spread over time and space may have widely heterogeneous computation and communication capabilities. Faults might arise in individual components. The ability of the application to tolerate and react to such faults will depend on the time and location of the fault. Fundamentally, these applications need to reason about events with respect to time, location, and identity in an integrated manner to control application behavior. Time, space, and identity refer, respectively, to the when, where, and who of events that drive the behavior of the application. We develop the system infrastructure to support the ability to reason about time, space, and identity [1]. The system has the following components: reasoning operations, participation protocol, and communication model. The reasoning operations provide a rich set of APIs for navigating the three-dimensional space of space, time, and identity. The participation protocol allows the application components to initiate and maintain distributed and dynamic interactions with one another. The communication model allows the application components to exchange information necessary for the interaction. We evaluate the system along several dimensions. First, we qualitatively show the ease of programming complex pervasive

applications using our system. Second, we quantitatively evaluate the cost of the reasoning operations using a set micro-benchmarks. Third, we model an application (such as surveillance) using our system. Using this model we generate application level workload that is comprised of a many-to-many producer-consumer pipeline. The workload is used to quantify the application level performance of our system.

Application of Error Correcting Codes for Secure Distributed Storage. We have been investigating protocols for large scale fault tolerant secure distributed storage. The two main concerns here are security and availability. We have designed and implemented a new protocol that combines encryption and replication in one set of operations using cryptographic properties of error correction codes [16].

Remote Authentication over Wireless Networks. We study the problem of remote authentication [15] over a long range wireless network using large signature keys such as biometric samples (fingerprint, retinal scans etc.). Because of the large size of these keys, and continual need for authentication, considerable power and bandwidth are consumed by such a process. Authentication being only a background process supporting other transactions, should not take away too much of resources, especially bandwidth and power that are quite critical for small mobile devices. We present LAWN, a Light-weight remote Authentication protocol for Wireless Networks that is based on Error Correcting Codes. LAWN trades computation for communication and can be tuned for any desired security guarantee. While adding only low computational overhead, LAWN enables significant saving in bandwidth [17]. Under a reasonable energy consumption model, we show that this saving results in 70% to 80% saving in power for long-range wireless applications. Deploying LAWN needs extensive experimentation to tune the systems parameters for efficient and error-free operation. Thus we also present a structured and systematic experimental methodology of deriving the systems parameters for deploying LAWN into practice.

State Management in Web Services. We identify a problem for certain applications wishing to use the web service paradigm to enhance interoperability: rapid, robust state maintenance. While many features are available to support session data, special mechanisms for application state maintenance are less well developed. We discuss three different models to solve the problem and compare the advantages and disadvantages of each. Experimental results show that which model to use depends on application requirements. D-Stampede.NET is a platform supporting the development of applications that involve large time-sequenced data communication among heterogeneous clients. We describe our web service implementation along with our state server solution to the application state management problem. A simple demo application is described and measured to validate performance [20].

Stream-based Applications on Stampede. We explore optimization strategies and resulting performance of two stream-based video applications on a cluster of SMPs. The two applications are representative of a class of emerging applications, which we call "stream-based applications", that are sensitive to both latency of individual results and overall throughput. Such applications require non-trivial parallelization techniques in order to improve both latency and throughput, given that the stream data emanates from a single source and that the distribution of the data cannot be done *a priori*. In this work [3], we suggest techniques that address in a coordinated fashion the problems of data distribution and work partitioning. We believe the two problems are related and need to be worked together. To conduct our experiments, we have parallelized two video processing applications using the *Stampede* cluster programming system which provides abstractions for implementing time- and throughput-sensitive applications elegantly and efficiently. For the video texture application we show that we can achieve a speedup of 24.26 on a 112 processor cluster. For the color tracker application, where latency is more crucial, we identify the extent of data parallelism that ensures that the slowest member of the pipeline is no longer the bottleneck for achieving a decent frame rate.

Garbage Collection of Timestamped Data. There is an emerging class of interactive multimedia applications that deal with stream data from distributed sources. Indexing the data temporally allows for ordering

individual streams as well as correlation across streams. Stampede programming system facilitates the organization of stream data into *channels*, which are distributed, synchronized data structures containing timestamped data. Channels allow multiple threads to produce and consume items in sparse, and out of timestamp order. These flexibilities are required due to the dynamic and parallel structure of applications. Under these circumstances, a key issue is the "garbage collection" of channel items. In this work [12], we present and compare three different GC algorithms: (1) REF is a simple localized algorithm that keeps a reference count on individual items; (2) TGC, which needs no application knowledge employs a distributed algorithm for computing a low water-mark for timestamp value of interest in the entire application; and (3) DGC, uses an application task graph to determine the relevant timestamps local to each channel and thread. DGC can simultaneously eliminate garbage from channels and unneeded computations from threads. In tests performed using an interactive application, DGC enjoys nearly 30% reduction in the application memory footprint compared to TGC and REF.

1.1.2 Software Caching for Embedded Sensor Processing

This is continuation of the work we reported in the 2001-02 and 2002-03 annual reports to reconcile programmability with cost for embedded processing associated with sensors in a distributed sensing infrastructure.

The problem of hyperblock instruction caching has become particularly burdensome, requiring extensive dataflow analysis of arbitrary binary images. With the ARM platform in particular, this has led to unusual problems due to the instruction set limitations on immediate offset ranges. Typical ARM-based programs embed data addresses as constants in mid-instruction-stream, and accurate isolation of these is essential. When analyzing instruction sequences for block-oriented soft-caching, such address constants must be tagged and propagated during rewrite phases. Two prior undergraduate students, Christopher Parnin and Naila Farooqui, joined the graduate program with some expectation of continuing work on these issues. Due to changes of interest, this work is now being carried out by undergraduate Luke A. Snyder, who won an award for his results [19] thus far during the undergraduate research symposium at Georgia Tech.

Our earlier work on the Energy-Delay metric problem [11], coupled with our early results on performance analysis [4, 5] has led to a general observation and presentation on the energy efficiency of networks used as data storage vessels [6]. These results pose a general challenge to traditional design assumptions about the cost of maintaining local DRAM vs. communicating over a network interface. While the examples used were artifical to illustrate the issues, our results have shown that the soft-cache methodology can be much more energy efficient than traditional designs under not-atypical program behaviors.

Empirical verification of the soft-cache simulations and prelimary power/delay analysis is now in progress. Two ARM-based platforms, the ADI BRH Reference Design and the Intel Sitsang-400 PDA Reference Design, have been fully instrumented for power measurements and study. Detailed microbenchmarks have established baseline power signatures for different components of each platform. These microbenchmark results are now being integrated into a coherent power model that can be used in real-time, correlated to performance counters inside each CPU (the Intel 80200 and the Intel PXA255), will demonstrate total program lifetime energy signatures on a per-component basis (DRAM, network, Flash, etc.).

1.1.3 Applications

EventWeb. While the volume and diversity of multimedia permeating the world around us increases, our chances of making sense of the available information do the opposite. This environment poses a number of challenges which include achieving scalability while accessing all the available media, live and archived, inferring its context, and delivering media to all interested parties with its context attached. We envision [13] a solution to this set of challenges in a novel system architecture. As a starting point, however, we select a previously described framework, EventWeb, suitable for annotating raw multimedia data with context meaningful to end users. We then map it onto a distributed architecture capable of correlating, analyzing, and transporting the volumes of data characteristic of the problem space. This paper first presents the requirements for our architecture, then discusses this architecture in detail, and outlines our current implementation efforts.

TVWatcher. With the explosion of streaming content in broadcast media, there is a need for a system architecture that automates the capture, filtration, categorization, correlation, and higher level inferencing of such data from distributed sources. TVWatcher [7] is a prototypical example of an application that demonstrates all of the above needs. This application allows user-controlled correlation of live television feed and enables a user to automatically navigate through the available channels to choose the content of interest. Symphony is an architecture for the distributed real-time media analysis and delivery which meets the system requirements for such applications. TVWatcher is built on top of the Symphony architecture, and currently uses closed-captioning information to correlate television programming. Through user studies we show that correlation engine is able to consistently pick significantly useful and relevant content.

1.1.4 Sensor Technologies

Sensor Lab We have provisioned a Sensor Lab within the College of Computing to support our research into flexible software infrastructure for pervasive, heterogeneous sensor networks. Our lab includes modest quantities of a wide variety of devices, both wired and wireless, from simple commercially available (COTS) components to complex research prototypes. The intent of this lab is to serve as a generator of media and sensor streams for use in allied research projects. This lab emphasizes integration of heterogeneous sensors and actuators, including high and low bandwidth devices with varying levels of onboard intelligence.

The Sensor Lab has evolved over time. We have gradually added new technologies and developed new sensor/actuator-based applications. Each new development draws on the experience of the previous prototype or application version. Requirements and research foci have been refined and clarified. Promising technologies and techniques have been adopted as we have become acquainted with them through discovery, colleagues, or conference and journal publications.

Sensored Spaces Devices are currently deployed within the Systems Studio (CCB 201), the Systems Lab (CCB 207) the newly outfitted Systems Smart Conference Room (CCB 252), the Aware Home, a lab facility in Technology Square, and a special media space housed in the Rich Building, the home of the campus computing group (OIT). Additional devices are deployed throughout the second floor of the College of Computing and lounge areas. As we develop confidence and experience with our infrastructure, we seek to extend our deployment to outdoor sensors on the Tech Campus. We are also discussing various possible environmental monitoring projects and accessing sensors deployed by external (OIT, Aerospace Unmanned Autonomous Vehicle Lab) and commercial partners (UPS, Ga Tech Hotel, etc.). See below for more details.

JSTk: Java Sensor Toolkit During the summer of 2004 Research Scientist Phillip Hutto lead a group of undergraduates in the design and development of a Java-based Sensor Toolkit [8]. The system was designed to provide a comprehensive middleware framework for tight integration and coordination of sensor and actuator resources.

The JSTk infrastructure provides high-level, network-aware, user-space "device drivers" that control and mediate their associated devices. Devices differ wildly in their characteristics and capacities so a common core api (interface) will provide an intersection of capabilities with additional device-specific interfaces for unique characteristics. Thus, the infrastructure resembles three OS (Linux) abstractions: the device abstraction, the virtual filesystem interface, and the networking interface. It is layered, like the networking interface, with basic "low-level" capabilities used to implement higher, more powerful abstractions (like persistence, streaming, eventing, etc.).

The design and implementation is object-oriented and utilizes best practices for decoupling, hierarchy, and abstraction (such as the toolkit or framework design pattern), although it also provides a "flattened" procedural interface with limited capabilities for legacy clients. In addition, a web services interface is supported. Note that web service interfaces are, by necessity, procedural so these two issues inter-related.

Each device or group of devices will has a primary software controller (manager, handler, adaptor) called the device *Mediator*. We chose this term because it has a a higher level connotation and includes the notion of "media" which suggests the streaming metaphor. The entire architecture can be thought of as a Virtual Sensor Switch or Virtual Sensor System (VSS) where "sensor" means broadly any sort of device (sensor or actuator or combination) and "virtual" emphasizes the notion of abstraction and that "devices" might be

software entities as well.

As a minimum, the core interface provides capabilities for activating and acquiring the managed device and "reading" and/or "writing" (sending/receiving, sampling/displaying, etc.), and deactivating and releasing the device. A variety of io styles (blocking, non-blocking, async, variable granularity, etc.) are supported along with thread-safe interfaces.

Managing or controlling the device from a user-space mediator must rely on OS support. Special OS-level techniques (such as a special user id that "owns" the device) are employed to avoid subverting access control. Otherwise, only "advisory" access control can be provided. That is, a client might access the device using a low-level OS interface, bypassing the access control checks in the mediator. This is an OS-dependent aspect of the architecture and one of the main benefits that would result from moving the mediator into kernel space (as, for example, a Linux kernel module).

Once we start considering the "network face" of the mediator, a variety of standard networking issues come into play. Ideally transport and security protocols should be selectable and interoperability should be provided. By default, we provide TCP command and data streams. UDP and RTP are supported by our use of the Java Media Framework. Custom transports can be included with some effort. The broad range we are attempting to cover makes it difficult to support a single transport. Some applications require reliability, others don't. Efficient transport of audio and video often requires the ability to drop a sample or frame from time to time.

In addition, once mediators are scattered across machines, some sort of naming/location mechanism is required. We provide a relatively simple, XML-based registry, initially centralized. Our design supports building or campus-scale systems and registry requests are relatively infrequent compared to data movement so a single, fast registry probably will suffice and can be federated for further scalability.

Failure-handling is another "cross-cutting concern" for network-aware applications. The registry implements leasing for failure detection and robust clean-up.

SLCP: Sensor Lab Control Panel The Sensor Lab Control Panel (SLCP) is a graphical application that allows authorized users to inspect known devices, determine their status, interrogate their capabilities, sample sensor data or issue actuator commands as appropriate, display results, persist sample or configuration data, and, generally, monitor and control devices. The system allows management of known (registered) devices and supports detection and discovery of devices running suitable client software (protocol stack). Sensors and actuators may be physical or virtual. The system supports concurrent users as possible and aims to be easily usable by casual (but knowledgeable) users such as students and faculty within the College of Computing.

Major features of SLCP include: 1) the ability to manually register/unregister device types and devices; 2) the ability to view the status of known devices; 3) the ability to discover new (unknown) devices of known types (e.g. handhelds, laptops); 4) the ability to select device sets (for display, actuation, etc.); 5) the ability to sample and display sensor data in a variety of formats; 6) the ability to issue commands to actuators; 7) the ability to activate, sample, persist (log) data streams; and 8) the ability to monitor active devices with minimal intrusion.

Outdoor/External Deployment As a next step in our sensor infrastructure research we have been seeking interesting applications and deployments from real-world partners, including deploying sensors in outdoor settings. First, we hope to deploy some exterior cameras near the ”wireless corridor” on the Georgia Tech campus to experiment with hardening and physically securing devices and to investigate bandwidth restrictions. These cameras will be placed in such as way as to minimize privacy concerns. Second, we are in discussions with various potential partners to use our infrastructure to control existing high-bandwidth sensors in real-world settings on and off campus. Possibilities include accessing surveillance cameras used by the campus computing group (OIT) to monitor workstation labs on campus, security cameras in the Georgia Tech Hotel, cameras in loading docks and other commercial facilities (UPS) and possibly devices on unmanned aerial vehiclas (Tech UAV Lab). Third, we are in discussions with members of the Environmental Studies Department and the School of Public Health at Emory University for possible off-campus environ-

mental monitoring applications. The researcher at the School of Public Health is also allied with the Centers for Disease Control. We look forward to the challenges of outdoor and real-world deployment and the usual feedback we will receive.

1.2 Training and Development

We continue to attract bright and interested graduates and undergraduates to research projects in our group. Undergraduate participation in research within the College is facilitated by the excellent UROC program (`www.cc.gatech.edu/program/uroc`), coordinated by Amy Bruckman. A variety of institute-wide programs are also available (`www.undergraduateresearch.gatech.edu`) including a special fund sponsored by the president of Georgia Tech (PURA) and several NSF-sponsored projects. We were pleased to support four undergraduate on ITR-related projects during the Spring semester of 2004. They were: Ken Edwards (TVWatcher), Zachary Crowell (EventWeb), Garret Boyer (hardware-related), and Ilya Bagrak (MediaBroker). For details of the PURA program, along with a list of recipients, see the website.

Many of the ongoing ITR-related projects are partially staffed by students working in the context of the Systems Hackfest. This is a group of undergraduates who participate in various research projects for pay, course credit, or just for fun. Hackfest is supervised by Research Scientist Phil Hutto and runs throughout the year. Summer sessions are most productive and have recently involved 6-10 students. Students meet briefly in a weekly session to report progress and plan milestones for the coming week. The group meeting allows cross-fertilization of project ideas and helps to educate the students. In addition, it provides an opportunity for group brainstorming on design and debugging issues. Weekly project meetings are focused on specific research tasks and often involve relevant faculty, grad students and staff.

During the last year undergraduates have participated in the following projects: SensorLab, TVWatcher, MediaBroker, and EventWeb. We are also pleased by the number of undergraduates in our group who continue on to graduate study both here at Georgia Tech and at other top schools.

We believe the Hackfest is an excellent opportunity for initiating undergraduates into the form and substance of academic research. In addition, the size and maturity of the inter-related research efforts provides a fertile matrix for varied interactions and training. Each group – undergraduates, Masters students, PhD students, research scientists and senior faculty – have regular opportunities for cross-group interactions. For example, undergraduates can look to senior faculty for vision and research goals, to research scientists for design advice, to graduate students for technical assistance and literature questions, and to each other for day to day collegiality.

1.3 Outreach

The PI (Professor Ramachandran) was the program co-chair for the *International Workshop on Future Trends in Distributed Computing Systems (FTDCS)*, 2004, held in Suzhou, China. Professor Ramachandran was also one of four invited speakers at a two-day workshop held at National Chiaotung University in Taiwan on pervasive computing, in December 2003. In addition, he presented invited talks at several universities and Samsung Research Lab in South Korea. As a result of the Samsung visit, there is an ongoing dialogue to establish a joint research relationship between our group and Samsung.

During the summer of 2004 we have established comprehensive websites for all of our related projects and activities including personnel and publications. These websites offer extended abstract-style overviews of each project and discuss work in progress. We believe this effort will contribute to our research group's visibility.

Through the auspices of the Center for Experimental Research in Computer Systems (CERCS) we continue to invite and host key individuals from academia and commercial research labs engaged in complementary research. Recent visitors include: Prof. Krithi Ramamritham from IIT Bombay; Prof. Gilles Muller from Ecole des Mines de Nantes, France; Mathai Joseph from Tata research Development and Design Centre; Prof. Peter Steenkiste from Carnegie Mellon University; and Dr. Sugata Ghosal, IBM India Research Lab.

The annual CERCS NSF/IUCRC Workshop on Experimental Research in Computer Systems was held in October 2003 and gave an opportunity for our group to interact with a distinguished list of advisory board members such as Philip Bernstein (Microsoft), Felipe Cabrera (Microsoft), Alan Ganek (IBM), Dennis Gannon (Indiana University), Daniel Reed (UNC Chapel Hill), and Raj Yavatkar (Intel).

We continue to place student members of our research group in interesting project-related internships, graduate programs and industry jobs. PhD student Xiang Song is spending the Summer and Fall of 2004 working with Dr. Raj Kumar at HP Labs in Palo Alto on grid infrastructure for "appliance computing." PhD student Namgeun Jeong has been working for several months at the Federal Reserve Bank in Atlanta with their cluster computing group. PhD student Bikash Agarwalla has recently returned to Georgia Tech after a year working at HP Labs in Palo Alto on interactive grid schedulers. Undergraduate Ilya Bagrak is currently pursuing a Master's degree at the University of California at Berkeley after graduating from Tech. Undergraduate Zachary Crowell is currently doing an internship at Microsoft; and Yavor Angelov is pursuing his PhD under Professor Ramachandran while employed at Microsoft. Former Master's student Derick Pack is joining a Naval Research laboratory in Charleston, South Carolina.

2 Publications and Products

2.1 Publications

See the references at the end of this document.

2.2 Web Site

Please visit the project web site at www.cc.gatech.edu/~rama/ubiq-presence/

3 Contributions

The activities we are currently undertaking have resulted in a significant number of publications and software artifacts. These are listed in the references at the end of this report.

3.1 Human Resources

We have roughly 17 graduate students, 12 undergraduate students, 4 research scientists, 1 post-doc, 1 visiting faculty member, and 1 technical staff member working in areas related to this project.

3.2 Research and Education

The research artifacts from the project are finding their way into graduate courses and we have signficant undergraduate participation in project-related research. We have funded about 8 undergraduates through the Research Experience for Undergraduates grant supplement, sponsored a similar number of independent undergraduate research projects for course credit (CS 4903) and have sponsored three capstone senior design projects (CS 3901) that each result in a poster presentation at the annual Undergraduate Research Symposium. One senior design project is in progress related to MediaBroker Federation (James Kim) and we anticipate a SensorLab project (Robert Thomas) in Spring 2005.

SensorLab resources were used in Spring 2004 for a project in an ECE graduate seminar (ECE 8883A Sensor Enabled Embedded Systems) taught by Visiting Professor Mark Smith from HP Labs. SensorLab resources are used regularly in two CoC graduate seminars, CS 8803E Pervasive Computing with Distributed Sensors (taught by Prof. Ramachandran) and CS 7470B Mobile and Ubiquitous Computing (taught by Prof. Gregory Abowd). SensorLab software will be used for a project in a graduate course on Distributed Systems (CS 559) taught by Phillip Hutto at Emory University in Fall 2004. We hope that as the SensorLab software matures, it will receive even more exposure and use at Georgia Tech and other schools.

4 Special Requirements

The total budget for this 5-year proposal is $1.35M. However, due to fiscal constraints NSF chose to front-load the total budget by awarding $750,000 in the first year. The second and third year increments were $200,000. The understanding with the program manager (Dr. Helen Gill) is that our spending plan for the award will be more balanced over the 5-year period despite the front-loaded nature of the allocation. Hence, we have over $500,000 still remaining from the first three years of allocation.

References

[1] Sameer Adhikari. *Programming Idioms for Pervasive Computing*. PhD thesis, College of Computing, Georgia Institute of Technology, December 2004.

[2] Sameer Adhikari, Arnab Paul, Bikash Agarwalla, and Umakishore Ramachandran. D-stampede: Distributed programming system for ubiquitous computing. *IEEE Transactions on Parallel and Distributed Systems*, 2003. Expanded version of a paper that appeared in the 22nd International Conference on Distributed Computing Systems (ICDCS), submitted to *IEEE TPDS*, under revision.

[3] Yavor Angelov, Umakishore Ramachandran, Kenneth Mackenzie, James Matthew Rehg, and Irfan Essa. Experiences with optimizing two stream-based applications for cluster execution. *Journal of Parallel and Distributed Systems*, 2003. Under revision.

[4] J.B. Fryman, C.M. Huneycutt, H.S. Lee, K.M. Mackenzie, and D.E. Schimmel. Energy efficient network memory for ubiquitous devices. Technical Report GIT-CERCS-03-05, CERCS, Gerogia Institute of Technology, 2003.

[5] J.B. Fryman, H.S. Lee, C.M. Huneycutt, N.F. Farooqui, K.M. Mackenzie, and D.E. Schimmel. Softcache: A technique for power and area reduction in embedded systems. Technical Report GIT-CERCS-03-06, CERCS, Gerogia Institute of Technology, 2003.

[6] Joshua B. Fryman, Chad M. Huneycutt, Hsien-Hsin S. Lee, Kenneth M. Mackenzie, and David E. Schimmel. Energy efficient network memory for ubiquitous devices. In *IEEE MICRO*, Sept/Oct 2003.

[7] David Hilley, Ahmed El-Helw, Matthew Wolenetz, Irfan Essa, Phillip Hutto, Thad Starner, and Umakishore Ramachandran. Tv watcher: Distributed media analysis and correlation. Technical Report GIT-CERCS-04-25, Center for Experimental Research in Computer Systems, Georgia Institute of Technology, 2004.

[8] Phillip Hutto, Simon Chen, Ryan Graciano, James Kim, Thomas Shanks, Rex Sheridan, Robert Thomas, Matthew Wolenetz, and Umakishore Ramachandran. Sensorlab: A Java sensor toolkit (jstk). *Third IEEE International Conference on Pervasive Computing and Communications (PerCom'05)*, March 2005. Under submission.

[9] Rajnish Kumar, Matthew Wolenetz, Bikash Agarwalla, JunSuk Shin, Phillip Hutto, Arnab Paul, and Umakishore Ramachandran. Dfuse: A framework for distributed data fusion. In *Proceedings of the First International Conference on Embedded Networked Sensor Systems (SenSys'03)*, pages 114–125, November 2003.

[10] Rajnish Kumar, Matthew Wolenetz, Brian Cooper, Bikash Agarwalla, JunSuk Shin, Phillip Hutto, Arnab Paul, and Umakishore Ramachandran. Dynamic data fusion for future sensor networks. *IEEE Transactions on Sensor Networks*, 2004. Under submission.

[11] H.S. Lee, J.B. Fryman, A.U. Diril, and Y.S. Dhillon. The elusive metric for low-power architecture research. In *Proceedings of the Workshop on Complexity-Effective Design*, 2003.

[12] Hasnain A. Mandviwala, Nissim Harel, Kathleen Knobe, and Umakishore Ramachandran. Distributed garbage collection for timestamped data. *IEEE Transactions on Parallel and Distributed Systems*, 2003. Expanded version of two papers that appeared in earlier conferences, *ICPP 2002* and *LCPC 2002*, under submission.

[13] Martin Modahl, Ilya Bagrak, , Matthew Wolenetz, Ramesh Jain, and Umakishore Ramachandran. An architecture for eventweb. In *Proceedings of the 10th IEEE International Workshop on Future Trends in Distributed Computing Systems (FTDCS'04)*, pages 95–101, May 2004.

[14] Martin Modahl, Matthew Wolenetz, Phillip Hutto, and Umakishore Ramachandran. Mediabroker: An architecture for pervasive computing. In *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, pages 253–262, March 2004.

[15] Arnab Paul. *Application of Error Correcting Codes to Distributed Pervasive Computing*. PhD thesis, College of Computing, Georgia Institute of Technology, December 2004.

[16] Arnab Paul, Sameer Adhikari, and Umakishore Ramachandran. Design of a secure and fault tolerant environment for distributed storage. Technical Report GIT-CERCS-04-02, CERCS, Georgia Tech, January 2004.

[17] Arnab Paul, Rajnish Kumar, and Umakishore Ramachandran. Computation-communication trade-off for power and bandwidth saving for remote authentication over wireless networks. Technical Report GIT-CERCS-04-01 (submitted to IEEE INFOCOM 2005), CERCS, Georgia Tech, January 2004.

[18] Umakishore Ramachandran, Rishiyur Nikhil, James Matthew Rehg, Yavor Angelov, Arnab Paul, Sameer Adhikari, Kenneth Mackenzie, Nissim Harel, and Kathleen Knobe. Stampede: A cluster programming middleware for interactive stream-oriented applications. *IEEE Transactions on Parallel and Distributed Systems*, November 2003.

[19] Luke Snyder, Joshua Fryman, Chad Honeycutt, and Umakishore Ramachandran. Poster presentation: Better cache design through memory reference behavior. In *Undergraduate Research Symposium, College of Computing, Georgia Tech*, 2004. Third Place Award.

[20] Xiang Song, Namgeun Jeong, Phillip Hutto, Umakishore Ramachandran, and James Rehg. State management in web services. In *Proceedings of the 10th IEEE International Workshop on Future Trends in Distributed Computing Systems (FTDCS'04)*, May 2004.

[21] Matthew Wolenetz, Rajnish Kumar, Junsuk Shin, and Umakishore Ramachandran. Middleware guidelines for future sensor networks. In *Proceedings of the First Workshop on Broadband Advanced Sensor Networks*, 2004. To appear.