Wireless Networks 7, 601–607, 2001

Multi-Fidelity Algorithms for Interactive Mobile Applications *

M. SATYANARAYANAN and DUSHYANTH NARAYANAN
School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

Abstract. We introduce the concept of *multi-fidelity algorithms*, which revises the classical notion of an algorithm. Instead of having a fixed output criterion and allowing the resource consumption to vary, we bound the resource consumption and allow the *fidelity* or output criterion to vary. We discuss how multi-fidelity algorithms can improve the latency and battery life of interactive mobile applications. An extension of this idea allows the system to automatically discover *sweet spots*: sharp discontinuities in the fidelity-resource tradeoff space.

Keywords: multi-fidelity algorithms, mobile computing, augmented reality, adaptation, sweet spot, search, Odyssey, Aura

1. Introduction

The concept of an algorithm has proved robust over half a century of advances in the speed and versatility of computing hardware and software. In this paper, we show why *interactive mobile applications* require us to rethink this concept from first principles. Such applications are difficult to support because they place heavy resource demands on hardware that is typically optimized for weight, size and battery life rather than compute power. We show how the notion of an algorithm can be extended to help alleviate this problem, and examine the implications of this shift in viewpoint. The paper is organized in three parts: rationale, research agenda, and related work.

2. Rationale

2.1. Classical view of an algorithm

Informally, an algorithm is a sequence of steps to accomplish some computing task. This task has a precisely-defined *output specification*. A sequence whose execution does not always meet this specification is not considered to be an algorithm for that task. For example, a sorting algorithm must preserve all its input elements but reorder them according to some precisely-defined sort criterion. No deviation from this specification is allowed in a candidate that claims to be a sorting algorithm.

Resources such as time, space or energy needed to accomplish a task are *dependent* variables. As much of each resource is consumed as necessary to meet the output specification. The figure of merit of an algorithm is how sparingly it uses one or more of these resources while meeting the output specification.

* An earlier version of this paper appeared in *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods in Mobile Computing and Communications*, Seattle, WA (August 1999).

2.2. Why a shift in viewpoint is needed

A simple example will help illustrate why mobile computing requires us to revise the classical notion of an algorithm. For brevity, this example is contrived – more realistic examples follow later in the paper.

Imagine a police officer on the beat responding to an assault victim. Using a hand-held mobile computer with a wireless link, the officer helps the victim identify the assailant from a list of wanted suspects. The victim says that the only memorable feature about the assailant was that he had a hairy face. The officer pulls up photographs of suspects, in order of facial hairiness. After a few suspects with full beards, and then a few with mustaches, the remaining suspects are all clean-shaven. Disappointed that the trail is cold, the officer goes with the victim to the precinct office to give a full crime report.

In this scenario, how does the program on the hand-held mobile computer yield images in order of facial hairiness? The instinctive answer for most computer scientists would be to simply sort the array of suspects using "facial hairiness" as the comparison function and then walk down the list. This would directly lead to the use of QuickSort, which is known to have good average case performance.

What this fails to recognize, however, is that only a small part of the sorted output is viewed by the victim. Because the victim gives up long before all the images are seen, most of the work done in the sort is wasted. Perhaps a better approach would have been to just find the hairiest face and present it, then the next, and so on until the victim terminates the process. That way, no work is wasted. Unfortunately, if the victim is stubborn and insists on viewing all the suspects' images, the system would effectively be performing a full sort using SelectionSort! In that case, using QuickSort may indeed have been the right strategy.

The important point is that it is not known *a priori* how many images the victim will want to see. In other words, the output specification is not precise. Not surprisingly, the choice of the best algorithm for this task is unclear.

So far, little in this example is specific to mobile computing. However, suppose another victim had reported a similar attack to this officer earlier in the day, and that facial hairiness was also the most prominent feature of that assailant. In that case, the chances are good that images of the hairiest faces are still cached on the officer's hand-held computer. On a mobile computer, restricting our search to the local cache would save us significant time and energy, since we avoid using the wireless network. On the other hand, searching the entire database would give a slightly better result at a much higher cost. The best approach might be to present the search results from the cached data, with an indication that they are incomplete. Only if the victim wishes to search further need missing images be fetched.

2.3. What is a multi-fidelity algorithm?

How can we extend our notion of an algorithm to be a useful concept in scenarios like this? Our approach is to relax the requirement that there be a single output specification. Instead, we allow a *range of possible outcomes*, where the range can be a few discrete states or a continuum.[1]

Intuitively, the range corresponds to different output quality levels or *fidelity levels.* In the sorting example, for instance, if there are M leading elements in sorted order in an array of size N, we could treat the ratio M/N as the measure of fidelity of the result. Perfect fidelity would have M equal to N – the full array is sorted; small values of M/N correspond to low fidelity.

We can now define a *multi-fidelity algorithm*: it is a sequence of computing steps that terminates, yielding a result that falls within a range of acceptable output specifications, called fidelities. Upon termination, a multi-fidelity algorithm indicates the fidelity of the result.

2.4. When is this concept useful?

Multi-fidelity algorithms allow us to formulate problems in ways that were not possible before. Most importantly, *the roles of output specification and resource consumption can be reversed.* In other words, we can now say "Give me the best result you can using no more than X units of resource R." Or, we can let the system make the tradeoff between fidelity and resource consumption by saying "Give me the best result you can cheaply." Depending on the specific state of the computing engine, the network and other environmental attributes, the fidelity of the result may vary from execution to execution.

Interactive applications are a natural application area for multi-fidelity algorithms. To preserve usability, it is often necessary to present a result to the user within a certain period of time. In many cases, a user would rather see a less-than-perfect result soon than suffer from long response times. When such a low-fidelity result is being presented, the user would like some indication of this fact. On occasion, he may repeat his request insisting on a high-fidelity result. More commonly, the user may abort his task after the preview provided by a low-fidelity result. The non-zero probability of task abortion, coupled with user tolerance for low-fidelity results, makes multi-fidelity algorithms well suited for interactive applications.

The intersection of mobile computing and human-computer interaction is a particularly fertile application domain for multi-fidelity algorithms. Augmented reality applications with wearable computers involve particularly stringent constraints: response time is critical for user comfort, and computing resources such as CPU, memory, battery power and wireless bandwidth are at a premium. Multi-fidelity algorithms allow the tradeoff between output quality and resource consumption to be deferred until runtime rather than being wired in *a priori*.

2.5. Why not approximation algorithms?

It is useful to distinguish between multi-fidelity algorithms and a related but distinct concept: *approximation algorithms*. The latter are algorithms that produce results which are provably within some bound of the true result. A subset of this class of algorithms, called *polynomial time approximation schemes* [11], have a tuning parameter analogous to fidelity. Approximation algorithms are typically of interest for intractable (NP-hard) problems, and concentrate on reducing asymptotic complexity. A recent exception is the work by Frieze et al. on approximate methods to do Latent Semantic Indexing [10], a problem for which polynomial time solutions already existed.

In contrast, multi-fidelity algorithms are applicable in many situations where low-order polynomial solutions are available. Sorting, for instance, is $O(N \log N)$ in complexity and yet the example given earlier showed why one might use a multi-fidelity algorithm for this purpose. In the real world, even a reduction of time by a constant factor is extremely valuable. Further, classical complexity measures do not capture energy usage, which is a critical resource in mobile computing.

A multi-fidelity algorithm may be composed of diverse, unrelated algorithms, one of which is dynamically selected based on runtime tradeoffs. Thus approximation algorithms are properly viewed as a special case of multi-fidelity algorithms.

3. Research agenda

The shift in viewpoint described in this paper arose from our struggle to understand how best to support a broad range of interactive applications on resource-limited mobile computing hardware, especially wearable computers. Our earlier work on the Odyssey platform for mobile computing had shown us how lowering *data fidelity* could alleviate limitations in critical resources such as network bandwidth [18] or energy [7,8]. For example, when bandwidth is low, a video

[1] Of course, a continuum can only be approximated by a very large number of discrete possibilities in a digital system.

can be transmitted in black-and-white rather than color, or be sent after aggressive lossy compression. Similarly, making the frame size smaller can reduce energy consumption significantly.

This led us to ask whether a meaningful duality exists – are there circumstances where degrading computation rather than data proves useful? The counterpart to data fidelity is *computational fidelity,* which lies at the heart of multi-fidelity algorithms. We realized that we had already used this concept without recognizing it: the Odyssey speech recognition application uses a smaller vocabulary and a more restricted acoustic model when memory or CPU cycles are scarce. The user accepts the lower recognition quality in return for improved recognition speed on limited hardware.

Now that we have a promising conceptual model, our goal is to apply it to the family of problems that motivated this work. Specifically, we plan to extend Odyssey to support interactive applications using multi-fidelity algorithms. We elaborate on this in the following sections, beginning with examples of the kinds of applications we hope to support.

3.1. Examples

3.1.1. Rendering for augmented reality

An architect is designing the renovation of an old warehouse for use as a museum. Using a wearable computer with a head-mounted display, she walks through the warehouse trying out many design alternatives pertaining to placement of doors and windows, placement of interior walls, and so on. For each alternative, the augmented reality software on her wearable computer superimposes the proposed design change on the architect's view of the surroundings. In many cases, an aesthetic or functional limitation immediately becomes apparent and she rejects that alternative. In the few cases that survive this stage of scrutiny, the architect requests a more accurate visualization as well as standardized tests to ensure that the design is structurally sound and meets building codes.

In this application, rendering of 3-D objects is performed frequently. Before an object can be rendered, it needs to be appropriately colored or shaded according to the light sources present as well as the shadowing and reflected light from other objects. A well-known way to do this shading is with a *radiosity computation* [2]. This is highly compute-intensive, but our application requires low latency for good interactive response. Our architect wishes to do "quick-and-dirty" validations of her ideas, and is willing to sacrifice some fidelity in order to be able to try out many ideas interactively.

There are many ways in which the application can control the fidelity of rendering. It can choose between different algorithms such as progressive radiosity and hierarchical radiosity. It can also control the number of polygons used to represent 3-D objects. A representation with fewer polygons can be shaded faster, but will have a lower fidelity compared to the original.

Using a multi-fidelity approach, the application collaborates with Odyssey to choose between these alternatives. Odyssey's guidance is based on the current CPU availability, the latency constraints of the application, and knowledge about the CPU consumption of the application at various fidelities. If the data for rendering the 3-D objects reside on a remote server, then Odyssey must also decide on how much computation to perform at the server and how much at the client, based on CPU and network availability. The current cache state of the wearable computer and the residual energy in its battery are likely to be important influences on this decision.

3.1.2. On-site engineering calculations

An unexpected contingency has arisen at a bridge construction site: excavation for a pier has revealed a different soil type than planned for in the design. A civil engineer is at the site, exploring design modifications to the bridge to cope with this problem. To assist him in his work, he uses a handheld computer running a spreadsheet-like software package. He explores a number of alternatives, examining the impact of each on the strength of the bridge, on the manpower and material costs, and on the delays it will cause to the schedule. During the initial part of his exploration, speed is more important than accuracy of results – results are therefore displayed to him in a font and color indicating low fidelity. As he converges on a few promising alternatives, he requests higher fidelity results: the computations now take much longer, and the presentation of the results indicates the higher fidelity. Once he has selected what appears to be the best choice, this design modification is shipped over a wireless link to his company's supercomputer for full structural analysis and supervisory approval. Once he has received this approval, the engineer gives the modified design to the local personnel. Work can proceed without further delay.

In this example, the multi-fidelity algorithms are all numerical in nature. Depending on the specific problem, there are many ways in which different levels of fidelity can be obtained. For example, a successive approximation algorithm may terminate after just a few iterations to yield a low-fidelity result. A simulated annealing algorithm may change the coarseness of its mesh as well as the number of iterations to yield different fidelities. The number of terms in a series expansion can be varied to achieve a desired fidelity. Indeed, the concept of fidelity is probably most easily applied to numerical algorithms.

3.1.3. Scientific visualization

A visualization program presents earthquake simulation data as an animation. The transformation is highly compute intensive and proceeds in three pipelined stages: sampling the data onto a regular grid, computing isosurfaces, and rendering a 2-D view of the 3-D grid. The user specifies a region of interest, and a desired frame rate. The application then queries the system to find the appropriate configuration and fidelity level. As the animation continues, conditions in the system change; whenever the application needs to adapt

to the new state of the system, it receives a callback and is supplied with new fidelity parameters.

In this application, fidelity can be reduced by downsampling the input grid of data. In a distributed implementation of the application, the split of functionality between client and server is important. The optimal split depends on the current availability of client and server CPU as well as network bandwidth. Thus the system has to decide what the best split is, and what fidelity is sustainable while maintaining the desired frame rate.

Although scientific visualization is not closely related to mobile computing, it is an important application domain that can benefit from a multi-fidelity approach. Through visualization, large data sets such as MRI scans and astronomical measurements can be grasped more effectively by users. The Quake visualizer [1] is an instance of this kind of application.

3.2. Work in progress

We are in the early stages of designing extensions to Odyssey on Linux to support multi-fidelity algorithms. Odyssey is a component of *Aura*, an umbrella project of broad scope in ubiquitous computing. We believe that the ability to effectively run multiple concurrent applications is key to a good mobile computing environment. Our previous experience with Odyssey [18] has shown that to support concurrency effectively, we need centralized system support for resource monitoring and arbitration between applications.

Our goal is to support the kinds of applications described in the previous section. We plan to implement these extensions, gain hands-on usage experience, and then critically evaluate and refine the design in the light of this feedback. We see four major components to this work.

First, we need to develop an API that enables applications to communicate their constraints to Odyssey and to receive notifications to trigger fidelity changes. Odyssey already provides such an API to support data fidelity, but our work so far indicates that a new set of system calls will be needed to support computational fidelity.

Second, the code to support this API has to be designed and implemented. This code will have two major responsibilities: monitoring the availability of resources such as energy, bandwidth, cache space and CPU cycles; and allocating these resources to concurrent applications, one or more of which may be executing multi-fidelity algorithms. We expect that minor extensions to the existing resource monitoring code in Odyssey will suffice, but that the code for resource allocation will need to be completely redesigned.

Third, we will have to design and implement an interface that allows a user to express his constraints and preferences to Odyssey. Feedback about the fidelity of results can also be given through this interface. It can also be the focal point for collecting information to help predict likely user behavior. For example, as we have seen in the sorting example of section 2.2, it is valuable to have a good estimate of the likelihood of user abort for each application. It is not yet clear to us whether this support should be integrated with each application, or if there should be a single GUI for the whole system.

Finally, we need to develop the multi-fidelity applications that will enable us to evaluate our design. Our strategy is to start from existing applications and to modify them. Since their source code is most readily available to us, the initial applications we explore will be augmented reality rendering, described in section 3.1.1, and visualization, described in section 3.1.3.

Throughout this process, we also need to develop and refine an evaluation methodology for multi-fidelity systems. Since, to our knowledge, this is the first system of its kind, we need to develop new ways of measuring and reporting the performance of the system, and the effects of various design decisions.

Currently, we have developed a multi-fidelity API and a working prototype that implements this API. We have tested the system with two applications: the radiosity application described in section 3.1.1, and a web browser application that fetches degraded (lossily compressed) images over the web. We have also implemented *predictors* for CPU and energy usage. In order to make good adaptation decisions, we need to know an application's resource consumption at various fidelities, so that we can pick the most appropriate fidelity. Our approach is to log the past behavior of the application, and use the log history to make predictions about future resource usage. Our initial results [17] are promising; they indicate that *history-based prediction* is a feasible and simple way to learn the relationship between application fidelity and resource usage.

3.3. Open questions

The work described in the previous section merely scratches the surface of the rich problem space defined by multi-fidelity algorithms. We see many deeper issues, both theoretical and experimental, that will have to be explored to gain a full understanding of this area. These issues are of such diversity and breadth that only a community-wide effort is likely to address them successfully. We describe some of these issues in the rest of this section. The list of questions posed below is not exhaustive, but gives a flavor of the research problems in this area.

3.3.1. What is a good fidelity metric?

Choosing the right fidelity metric is important, but we currently have no systematic way to do this. Assigning fidelity values in an ad hoc way makes it difficult to evaluate and compare multi-fidelity algorithms. A more scientific approach would be valuable.

In some cases, such as numerical and sorting algorithms, the deviation from the ideal result is easily quantified and meaningful as a measure of fidelity. In other cases, such as algorithms for querying databases of images [6] and financial time sequences [14], it is possible to define a Euclidean distance measure that can serve as the metric of fidelity.

But there are cases, such as rendering or visualization algorithms, where assigning fidelity is harder because it depends on user perception. It would be helpful to have a clear taxonomy of algorithms based on fidelity metrics, together with guidelines for defining the appropriate metric for each class of algorithm.

3.3.2. How are multi-fidelity algorithms compared?
If we have two multi-fidelity algorithms, *A* and *B*, for the same task, which one should we choose? With conventional algorithms, we simply pick the one with lower algorithmic complexity. With multi-fidelity algorithms, this is problematic, since we lack a common basis for comparison.

If algorithm A always uses fewer resources for the same fidelity than algorithm B, we know that A is better than B. What if A is better at some fidelities, and B at others? What if the fidelity levels of A and B do not correspond in a one-to-one fashion? Overall, it would be of great value to find a rigorous approach to comparing multi-fidelity algorithms.

3.3.3. Can multi-fidelity computations be composed?
So far we have considered the case of computations with a variable output specification, but with a fixed, or immutable, input. What if the input could have variable fidelity as well? Imagine a cascaded computation, consisting of a pipeline of several multi-fidelity computations. The fidelity of each computational stage determines the input fidelity of the next.

The input fidelity might degrade the output fidelity much as a noisy input signal creates a noisy output. In other cases, the input fidelity might simply place a ceiling on the output fidelity: a rendering algorithm cannot produce a high-resolution 2-D image from a low-resolution 3-D model.

What is the right way to characterize the effect of input fidelity on a computation? How can we express the overall fidelity of cascaded computation in terms of the fidelities of its various components?

3.3.4. Can the system find sweet spots?
We have already seen that multi-fidelity algorithms allow us to fix resource consumption and allow the output specification to vary. In fact, they let us go a step further: we can free the system to make the tradeoff between the two, based on its knowledge of the current state of the system and the environment. In effect, we let the system find the "sweet spot" – the highest fidelity attainable with modest resource usage.

For example, section 2.2 has the example of a query on a database of faces. By degrading the query to operate only on the cached portion of the database, we sacrifice a small amount of fidelity in exchange for a large savings in time and energy. This degraded query corresponds to a knee, or sweet spot, on the fidelity-performance curve. For interactive applications, having the system make the tradeoff is often the right choice – the user does not have stringent requirements on either performance or fidelity, but simply wants a good compromise between the two. We need methods to automatically identify these sweet spots.

4. Related work

The defining characteristic of a multi-fidelity algorithm is the broadening of output specification from a single outcome to a range of acceptable outcomes. A key consequence of this relaxation is the possibility of role reversal – resource consumption can now be the independent variable in a computation, with fidelity being the dependent variable. To the best of our knowledge, this is the first paper to advocate such a broad change to the concept of an algorithm, and to identify its importance for interactive applications.

We are aware of three previous extensions to the concept of an algorithm that are related to this work. The first, approximation algorithms, has already been discussed in section 2.5. As explained there, approximation algorithms are focused almost exclusively on intractable problems and can be viewed as a subset of multi-fidelity algorithms.

Anytime algorithms [3] and their generalization, any-dimension algorithms [16], represent a second important extension to the concept of an algorithm. An anytime algorithm can be interrupted at any point during its execution to yield a result – a longer period before interruption yields a better result. Any-dimension algorithms are similar, except that they allow more general termination criteria. Since a range of outcomes is acceptable, these classes of algorithms can be viewed as multi-fidelity algorithms. However, the requirement that execution be interruptible at any point in time is an added constraint that restricts their generality. Hence, anytime and any-dimension algorithms are also subsets of the more general class of multi-fidelity algorithms.

The third extension, *imprecise computations* [4,5,12], support graceful degradation of real-time systems under overload conditions. Each computation is modeled as a mandatory part followed by an optional anytime part that improves the precision of the result. The real-time scheduler ensures that the mandatory portion of every task meets its deadline, and that the overall error is minimized. Since they allow multiple outcomes, imprecise computations are clearly instances of multi-fidelity algorithms. However, their restricted structure and real-time focus makes them a subset of the latter class.

From a broader perspective, the concept of *Quality of Service* (QoS) in networking and real-time systems bears some resemblance to the notion of fidelity. There are two main differences. First, QoS is typically viewed as a property of the environment, not the application. Second, fidelity is a broader concept of quality than QoS, which is typically used to indicate parameters such as bandwidth, jitter, and so on.

Finally, *randomized algorithms* [15] such as the Miller–Rabin primality test possess a property akin to fidelity. When such an algorithm terminates, there is a non-zero probability that the answer is incorrect. However, by repeating the algorithm many times, one can increase the probability of discovering the correct result. Thus, increasing the number of trials can be viewed as increasing the fidelity of the result.

5. Conclusion

The importance of *adaptation* in mobile computing systems is now widely recognized [9,13,19]. Only through effective adaptation can applications on a mobile computer overcome challenges such as unpredictable variation in network quality, wide disparity in the availability of remote services, limitations on local resources imposed by weight and size constraints, and concern for battery power consumption. Multi-fidelity algorithms are a natural fit for adaptive systems because they tolerate a range of outcomes, thereby offering many new degrees of freedom for adaptation.

In this context, interactive mobile systems pose special challenges as well as opportunities. On the one hand, meeting usability requirements is made more difficult by the resource constraints of mobile computing. On the other hand, the fact that a human user is generating requests and receiving output allows flexibility that would not otherwise be possible. Humans tend to be noisy sources of input and tolerant sinks of output. They trigger computations, then abort them at whim; they formulate a query, then realize that they meant something different after seeing some of the results; they are annoyed by slow response, but are often happy with a less than perfect result. By freeing the system to be more flexible in its responses, multi-fidelity algorithms are a good match for the idiosyncrasies of human users.

The change in viewpoint represented by multi-fidelity algorithms is simple, yet surprisingly powerful. While the full consequences of this shift remain to be worked out, we are convinced that multi-fidelity algorithms will play a key role in the future of interactive mobile computing systems.

Acknowledgements

Avrim Blum, Christos Faloutsos, Mor Harchol-Balter, David Petrou, Dan Siewiorek, and Andrew Willmott gave us valuable comments on an earlier version of this paper. Their input substantially improved its content and presentation.

This research was supported by the Air Force Materiel Command (AFMC) under DARPA contract number F19628-96-C-0061. Additional support was provided by Intel and IBM. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of AFMC, DARPA, Intel, IBM, Carnegie Mellon University, or the US Government.

References

[1] M. Aeschlimann, P. Dinda, L. Kallivokas, J. López, B. Lowekamp and D. O'Hallaron, Preliminary report on the design of a framework for distributed visualization, in: *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, Las Vegas, NV (June 1999).

[2] M.F. Cohen and J.R. Wallace, *Radiosity and Realistic Image Synthesis* (Academic Press Professional, Boston, MA, 1993).

[3] T. Dean and M. Boddy, An analysis of time-dependent planning, in: *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, Saint Paul, MN (AAAI Press/MIT Press, August 1988) pp. 49–54.

[4] W. Feng and J.W.S. Liu, An extended imprecise computation model for time-constrained speech processing and generation, in: *Proceedings of the IEEE Workshop on Real-Time Applications*, New York, NY (May 1993) pp. 76–80.

[5] W. Feng and J.W.S. Liu, Algorithms for scheduling tasks with input error and end-to-end deadlines, Technical report UIUCDCS-R-94-1888, University of Illinois at Urbana-Champaign (September 1994).

[6] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele and P. Yanker, Query by image and video content: The QBIC system, Computer 28(9) (September 1995) 23–32.

[7] J. Flinn and M. Satyanarayanan, PowerScope: A tool for profiling the energy usage of mobile applications, in: *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA (February 1999).

[8] J. Flinn and M. Satyanarayanan, Energy-aware adaptation for mobile application, in: *Seventeenth ACM Symposium on Operating Systems Principles (SOSP'99)*, Kiawah Island, SC (December 1999) pp. 48–63.

[9] G.H. Forman and J. Zahorjan, The challenges of mobile computing, IEEE Computer 27(4) (April 1994).

[10] A. Frieze, R. Kannan and S. Vempala, Fast Monte-Carlo algorithms for finding low-rank approximations, in: *IEEE Symposium on Foundations of Computer Science (FOCS)*, Palo Alto, CA (1998).

[11] M.R. Garey and D.S. Johnson, *Computers and Intractability* (Freeman and Co., New York, 1979).

[12] D. Hull, W. Feng and J.W.S. Liu, Operating system support for imprecise computation, in: *Flexible Computation in Intelligent Systems: Results, Issues, and Opportunities*, Cambridge, MA (November 1996).

[13] R.H. Katz, Adaptation and mobility in wireless information systems, IEEE Personal Communications 1(1) (1996).

[14] F. Korn, H.V. Jagadish and C. Faloutsos, Efficiently supporting ad hoc queries in large datasets of time sequences, in: *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, Tucson, AZ (May 1997).

[15] R. Motwani and P. Raghavan, *Randomized Algorithms* (Cambridge University Press, Cambridge, UK, 1995).

[16] D.J. Musliner, E.H. Durfee and K.G. Shin, Any-dimension algorithms, in: *Proc. Workshop on Real-Time Operating Systems and Software* (May 1992) pp. 78–81.

[17] D. Narayanan, J. Flinn and M. Satyanarayanan, Using history to improve mobile application adaptation, in: *Proceedings of the Third IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2000)*, Monterey, CA (December 2000).

[18] B.D. Noble, M. Satyanarayanan, D. Narayanan, J.E. Tilton, J. Flinn and K.R. Walker, Agile application-aware adaptation for mobility, in: *Proceedings of the 16th ACM Symposium on Operating Systems and Principles*, Saint-Malo, France (October 1997).

[19] M. Satyanarayanan, Mobile information access, IEEE Personal Communications 3(1) (February 1996).

Mahadev Satyanarayanan is the Carnegie Group Professor of Computer Science at Carnegie Mellon University. The Coda and Odyssey systems for mobile information access have been developed over the last decade under his leadership. Earlier, he was a principal architect and implementor of the Andrew File System. He received the Ph.D. in Computer Science from Carnegie Mellon in 1983, after Bachelor's and Master's degrees from the Indian Institute of Technology, Madras.

E-mail: satya@cs.cmu.edu

Dushyanth Narayanan is a Ph.D. candidate at the School of Computer Science, Carnegie Mellon University. He works on the Odyssey project headed by M. Satyanarayanan. His thesis work deals with system support for mobile, interactive applications. He received his Bachelor's degree in 1995 from the Indian Institute of Technology, Madras.
E-mail: bumba@cs.cmu.edu