1996

Web//ELLPACK: A Networked Computing Service on the World Wide Web

Sanjiva Weerawarana

Elias N. Houstis
Purdue University, enh@cs.purdue.edu

John R. Rice
Purdue University, jrr@cs.purdue.edu

Margaret G. Gaitatzes

Shahani Markus

Report Number:
96-011

WEB//ELLPACK: A NETWORKED COMPUTING SERVICE ON THE WORLD WIDE WEB

S. Weerawarana
E. N. Houstis
J. R. Rice
M. G. Gaitatzes
S. Markus
A. Joshi

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD TR-96-011
February 1996

Web //ELLPACK: A Networked Computing Service on the World Wide Web

**Sanjiva Weerawarana, Elias N. Houstis, John R. Rice,
Margaret G. Gaitatzes, Shahani Markus and Anupam Joshi.**

**Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, USA.
{saw,enh,jrr,mgg,markus,joshi}@cs.purdue.edu**

Recent advances in wired and wireless networking infrastructure are leading to a new form of computing we call *Networked Computing*. Networked Computing is a form of computing where a user takes advantage of networked computing facilities available through the global networks ("the Net"). Networked computing relies heavily on computing resources that are not locally present, but are only available to the user across the Net. Further, it provides an environment where components from different service providers can be combined in an intelligent manner to form a *virtual problem solving environment* for the problem at hand.

This paper describes the //ELLPACK networked computing service that we have made available through the World Wide Web. //ELLPACK is a *problem solving environment* (PSE) for solving partial differential equation (PDE) problems. In this document we describe our vision of networked computing and our experience in making this fairly complex system available over the web as a networked computing service. Our experiences indicate that this is a valid and realistic approach for providing computing services, but also suggest that to make networked computing practical, applications' architectures must change.

1. Introduction

As we leap forward towards the new millennium, we find ourselves in the midst of the networking phase of the information revolution. While the current emphasis in this revolution is mostly on how to better use the network for information access and not on how to use it for computation, we believe that *networked computing* (referred to sometimes as *virtual computing* or *meta-computing*) is the way of the future for computing.

1.1 Networked Computing

Networked computing is a form of computing where vital pieces of software used by some computing process are only *virtually*[1] available to the user via some network. This is in contrast to the current software usage model where one purchases a copy (or copies) of software for use within local hosts, possibly even distributed on a collection of local hosts. With networked computing, the view of software changes from a product to a service where the software developer will provide a computing service to interested parties over the network. The raw computing power to run this service may be purchased from the software service provider, be provided by the end user or may even be purchased from a third party computing service provider.

The networked computing model does not apply to all computing services of course; one will still acquire the software one uses constantly on a more permanent basis. Such software may include the basic operating system and network access software. We do believe however that the degree of permanency will change from the current one-time purchase model to that of "software leases" where one obtains usage rights for a certain period of time. Further, as software is improved upon by the software provider, it will be automatically downloaded to the user's site as long as the software lease is in effect. It is also possible that the compute power (the CPU, memory, disk space etc.) to execute software is purchased (leased) from a third party.

This new model of computing is similar to practice that is ubiquitous in other areas such as accounting/billing services where one uses software from one source along with the labor from another to achieve the desired computation. We envision that this model will eventually become fully automated

1. By *virtually* we mean that the software is only available remotely via over network.

and effectively transparent to the user in more general computing scenarios, after which market forces will make it the dominant paradigm.

The new paradigm is well suited to high performance distributed scientific computing scenarios such as multidisciplinary computing [1] and problem solving environments [2][3]. In such environments where one often studies complex physical phenomena, one applies pieces of disparate software from different sources to complete the desired computations. The networked model where one (remotely) accesses software services is clearly better suited for this type of scenario as it does not require prior, static software "purchases." This leads to more rapid access to new developments and to faster, more effective technology transfer. The networked computing model also brings forth a new degree of distributed computation. The current distributed computing model refers to computing that is distributed within a (parallel, distributed memory) machine or on a local area network. With networked computing, we have true distributed computing with components distributed on to platforms that may be thousands of miles away from the computation owner's location.

Figure 1 illustrates an instance of the vision of networked computing. A user located anywhere in the world accesses a networked computing service such as *Web //ELLPACK* via the network. The service's user interface is exported to the user in some fashion so that the user can easily interact with it. The user formulates the problem via this interface and has it solved at the server site. The server, in turn, accesses other servers (e.g., specialized solvers or parallel machines) via the network during the solution process. Finally, the results as well as the visualization user interface is returned to the user for further processing. To pay for the services, the user may use an electronic payment scheme such as DigiCash [4] or may have some prior *computation bartering* agreement with the service providers.

1.2 Relationship to the World Wide Web

The common person today identifies the network revolution with the World Wide Web. The World Wide Web (WWW, Web) [5] is a distributed information system which provides the policies and mechanisms for interconnecting pieces of related information as well as convenient graphical tools for navigating an "infosphere" of connected information. Recently, new developments such as the Java programming language [6] have transformed the Web from purely an information system to a

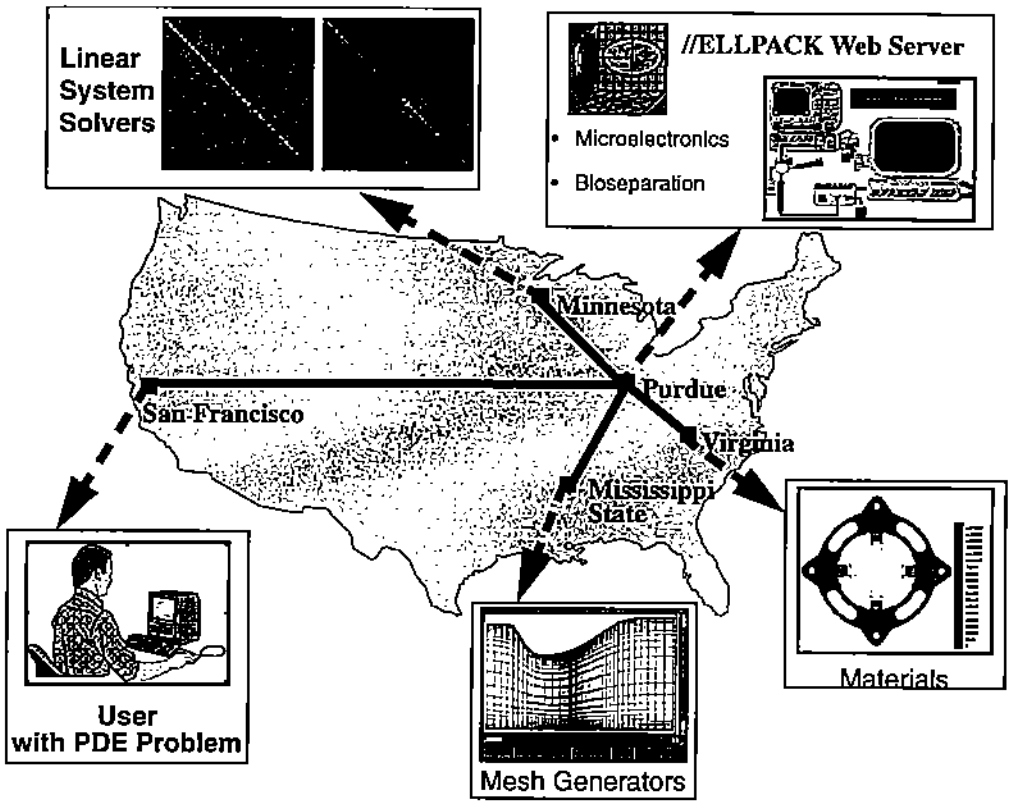

FIGURE 1. A future scenario for high performance, networked scientific computing.

more consumer-oriented environment where one can not only acquire information, but also make purchases and even perform some computing services. The ubiquity of the Web decries that (at least for the near future) any new networked computing models be "Web-friendly." In other words, it takes advantage of the Web protocols, facilities and most importantly software as much as possible. This has several advantages as well - it allows for networked computing services to be easily advertised, searched for and located using the mature, well-developed tools of the Web. It further allows seamless access from different computing platforms ranging from high powered workstations to supercomputers to desktop personal computers to hand-held personal digital assistant type devices. Last, but probably most important, it makes the software accessible to millions of users world wide while demanding little from them in terms of computing knowledge. Hence, we believe that networked computing services that will arrive in the short- to middle-term future will be appropriately Web-friendly and will take advantage of the technologies of the Web.

The Web can also be viewed as an *operating system* for the network [7]. One must then understand the components of the virtual computer presented by the network and then have a set of applications that bring to light the features and benefits of this model of computing. Some of our recent research efforts have been focused on clarifying this view and on realizing applications that exploit its facilities.

1.3 Web //ELLPACK

The first step towards realizing the networked computing environments of the future is to develop networked computing service providers. In this paper, we present *Web //ELLPACK*, a Web-based networked computing service that provides a sophisticated problem solving environment for solving partial differential equation (PDE) problems. *Web //ELLPACK* is a system that allows remote users to access and use our //ELLPACK problem solving environment [8] in a convenient and effective manner. We consider this to be a first generation networked computing environment and present our experience with developing and managing it. *Web //ELLPACK* is now available for public use at http://pellpack.cs.purdue.edu/.

This rest of this paper is organized as follows. The next section describes the //ELLPACK PDE solving system upon which *Web //ELLPACK* is built. Section 3 describes the architecture and implementation of the *Web //ELLPACK* system. Section 4 discusses some related work and indicates how our work differs from it. Section 5 discusses the limitations and future work in this effort and identify some of the technical and non-technical issues that must be resolved before one can fully realize the networked computing model.

2. The //ELLPACK Problem Solving Environment

//ELLPACK is a problem solving environment for solving PDE problems on high performance computing platforms as well as a development environment for building new PDE solvers or PDE solver components. The numerical solvers currently available in //ELLPACK include not only solvers that we have developed, but also a large collection of solvers developed by others. As a problem solving environment, //ELLPACK has a "natural" language for specifying PDE problems and their solution schemes. A complete graphical user interface assists users in specifying the PDE problem and the solution algorithm and also for analyzing computed solutions. Problems may be solved sequentially or in parallel on a variety of supported parallel platforms.

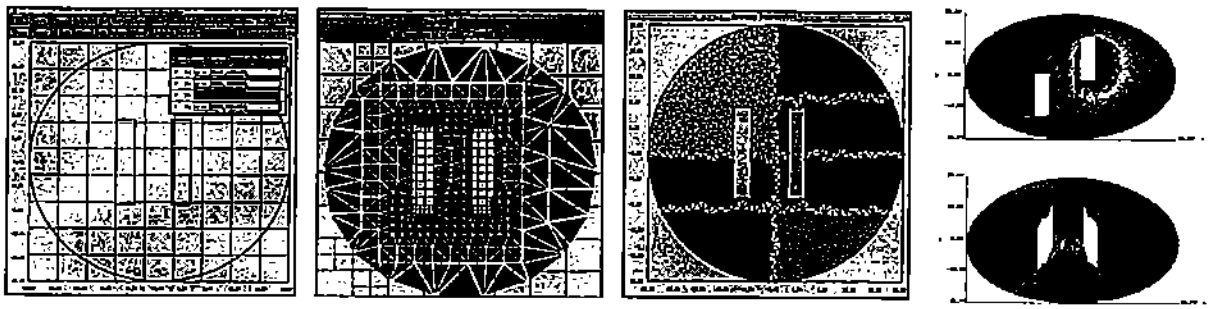

FIGURE 2. A view of some of the graphical tools in //ELLPACK.

The main design objective of //ELLPACK is to create an intelligent software environment where both sequential and parallel PDE solvers can be implemented in a reasonable time. //ELLPACK presents application users with a high level environment to abstractly specify PDE problems and build solvers for them using intrinsic solver components. Knowledgeable users can apply //ELLPACK's solver development facilities to build new solvers which can be made available as intrinsic solver components for application users.

The //ELLPACK system evolved from the well-known ELLPACK system [9]. //ELLPACK has evolved ELLPACK in many ways, including the addition of parallel solution facilities, a complete graphical user interface, symbolic manipulation of PDE problems, finite element solution methods and a framework for the seamless integration of foreign PDE and linear algebraic system solvers. The basic infrastructure of //ELLPACK described below thus extends the infrastructure present in ELLPACK to support the added functionality. Figure 2 shows some of the graphical domain specification, mesh generation, domain decomposition and solution visualization tools available in //ELLPACK.

The usage model adopted by //ELLPACK is based on decomposing the PDE problem and the solution process to their natural constituent parts. In this view, a PDE problem is said to consist of a partial differential equation, the domain over which this model is defined, the conditions that hold at the boundary of the domain and for time dependent problems, the conditions that hold at the beginning of the simulation. For a large class of numerical solution techniques, these components allow the solution process to be decomposed to domain discretization (generating the discrete domain over which the numerical solution is computed), operator discretization (generating the linear or nonlinear

algebraic equations that need to be solved) and solving the resulting system of equations. Other solution schemes combine all of these steps into one composite step that results in the final PDE solution. For parallel solutions, any or all of these steps may be executed on a parallel machine.

The original ELLPACK system was designed to support the specification and solution of second order, 1, 2 and 3-dimensional linear elliptic PDE problems, using primarily finite difference solution techniques. The //ELLPACK system uses this same basic computational infrastructure, but has added many components to better support nonlinear and time dependent problems, and both sequential and parallel finite difference and finite element techniques.

When solving a problem with //ELLPACK, one must specify the PDE problem and the solution algorithm to be applied using the high level PDE language and/or the graphical tools. This PDE model and solution specification is compiled by translating it to a FORTRAN driver program, compiling this driver with a native FORTRAN compiler, and then linking it with the appropriate libraries. After the resulting executable is run, the computed data may be imported into the solution analysis environment for visualization and analysis. The entire process is managed via the top level graphical environment of //ELLPACK.

In order to realize the //ELLPACK computational environment, we have adopted three levels of programming with standardized data structures and interfaces among the various PDE objects involved in the solution process.

At the highest level, the graphical user interface provides application users with knowledge-based, object-oriented editors to define problem components, specify the solution process and perform various post-processing analyses. The problem and solution specifications are expressed in terms of a high level PDE language, which is used to represent the PDE objects produced by the graphical editors. The //ELLPACK language processor is used to compile this high level problem and solution specification into a driver program that invokes various library modules to realize the user's solution process.

This architecture is implemented in //ELLPACK in terms of seven subsystems. These subsystems represent the solution process that application users follow. The *PDE Problem Specification Sub-*

system and the *PDE Solution Specification Subsystem* provide users with graphical editors, "foreign" system templates, the //ELLPACK language and embedded FORTRAN code. The *PDE Libraries* implement sequential and parallel solver components that are available to users via the solution specification subsystem. They include the ELLPACK solver library, the //ELLPACK solver library and "foreign" solver libraries such as FIDISOL, VECFEM, PDECOL and PDEONE. The various components are interconnected using standardized interfaces which support the "plug-and-play" approach provided by //ELLPACK. The *Knowledge Based Framework* assists users with the selection of an appropriate solution process for a given problem. The *Language Processor* uses the high level PDE language specification to generate a driver program that implements the solution scheme given in the high level specification. Furthermore, it is used to integrate new PDE solver components to the // ELLPACK system. The *Execution Subsystem* provides a framework for executing //ELLPACK programs. It helps users compile and execute programs on all the hardware and software platforms that //ELLPACK supports by managing the complexities associated with sequential and multi-platform parallel execution. The *Analysis Subsystem* provides users with graphical tools for visualizing and analyzing computed solutions and for collecting and visualizing performance data.

3. Web //ELLPACK

The goal of the *Web //ELLPACK* service is to allow remote users to access and use the //ELLPACK system in a safe, secure and effective manner. With the current architecture of //ELLPACK, this means that the service would be required to use one of our machines instead of a remote user's machine.

Our design was guided by the following principles: 1) We do not want to give outside users direct access to our machine(s) for obvious security reasons. 2) We do want to control who has access to the software, at least for the purpose of accountability. 3) We want to give users privacy; i.e., one user should not be able to freely browse other users' files.

To satisfy these constraints, we developed an account-oriented model where users "log in" to their "account" and then access the software. We chose to create the "accounts" within the data space of a custom Web server. We maintain access control using standard Web security constraints. To main-

tain security, we used several levels of Unix security mechanisms to prevent remote users from harming our machines.

The *Web //ELLPACK* service consists of the following components: a Sun SparcStation LX workstation, a special WWW server, the //ELLPACK system, and administration and maintenance software. In the rest of this section, we describe the operation of the system and its security mechanisms.

3.1 Operation

The machine pellpack.cs.purdue.edu is currently providing the *Web //ELLPACK* service to the Internet community. The top level page of this web site is shown in Figure 3. There are three tasks the user may do at this point: run a demonstration of *Web //ELLPACK*, request an account to use the service, or log in and use the service.

To run //ELLPACK (as a demonstration or otherwise), the user needs to have the X window system operational on his/her machine. In order for the machine providing the *Web //ELLPACK* service to display X windows on the user's display, the user must instruct their own machine to permit this action. The command for doing so is conveniently provided to the user in a set up page in both demonstration and actual runs.

Once the appropriate permissions have been set up, the demonstration is started by pressing the "Run" button. The demonstration //ELLPACK system then runs on the WWW service machine and displays its windows on the user's display. Figure 4 shows the demonstration in operation. The WWW browser is blocked until the execution is complete. Once the operation is complete, the user is presented with a page that allows them to send comments to the //ELLPACK developers.

For normal operation, the user must first request access to use the *Web //ELLPACK* service by filling a (Web) *form* and submitting it to the service administrators. We currently process these requests manually so that we have direct knowledge of who accesses the service. Once the *Web //ELLPACK* service is more stabilized, we expect to enable automatic processing of these requests. Processing the request requires us to provide the user with an initial login identifier and an initial password which the user may use to create an account for themselves. Then, we add this login to the WWW server access control file that controls who has access to the account creation page.

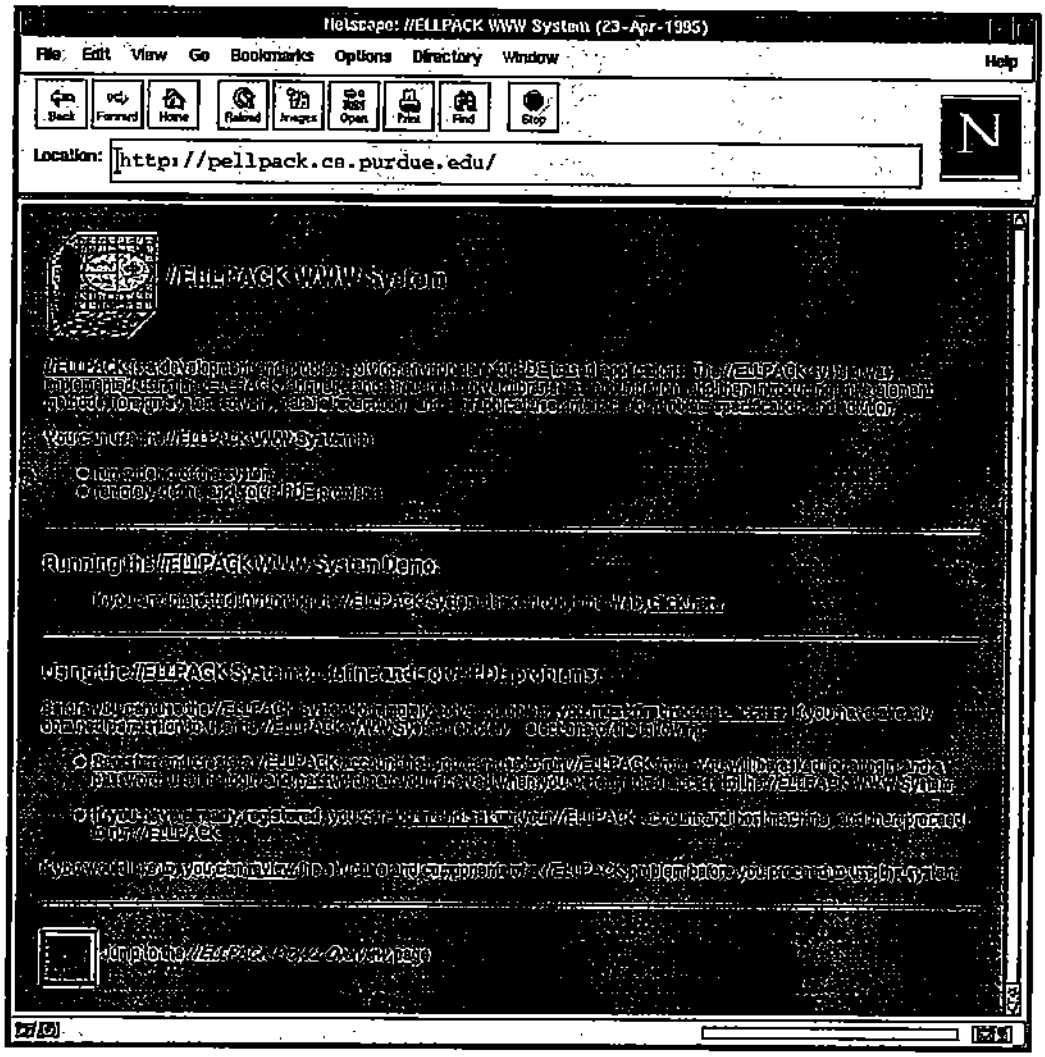

FIGURE 3. Homepage of the *Web //ELLPACK* service

We allow the user to create their own account as a matter of convenience for the user. Once the user receives the initial login and password, (s)he visits the account creation page using these tokens and sets up an account for themselves. The account information is submitted to us via another form (Figure 5). This request is processed automatically by creating a "home" directory for the user within the WWW server's data space and by creating an access control file there that restricts access only to this user.

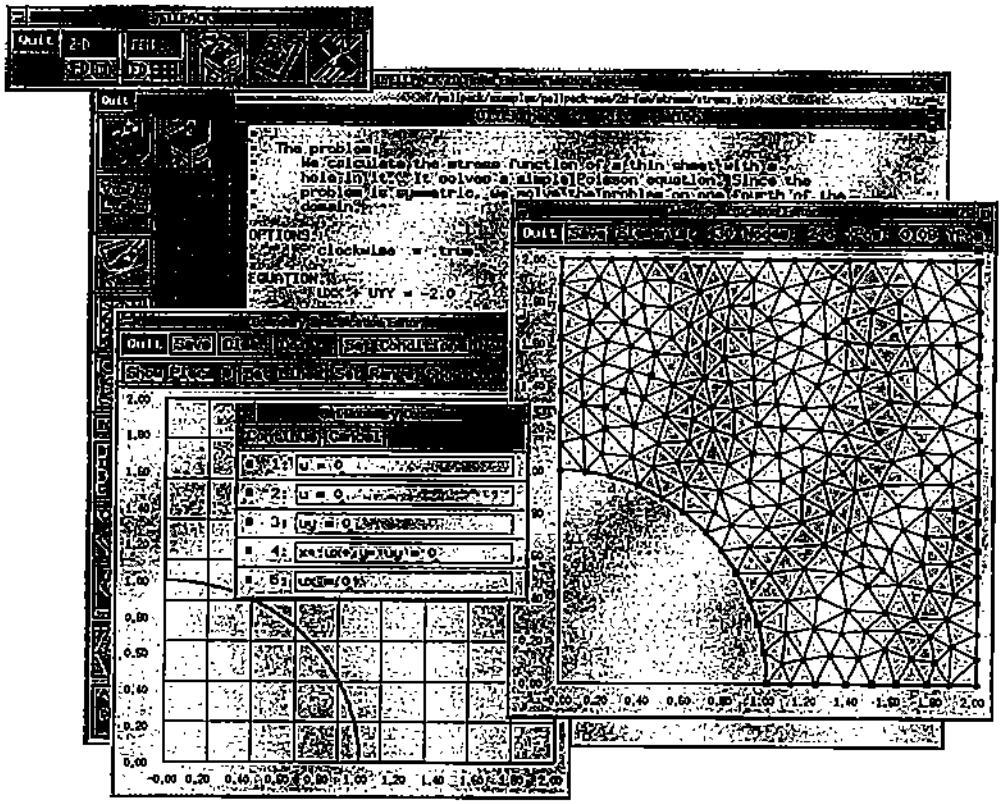

FIGURE 4. The *Web //ELLPACK* demonstration in operation.

Once the account has been set up, the user may log in any time and use the //ELLPACK system. The home directory works like a normal home directory; i.e., the user may use it as a persistent working area and may save programs and results there. Files may be off-loaded from this directory using a web browser, but we currently do not support uploading files to this directory. We expect to support file uploading using the support for file uploads in HTML3 [10] forms.

3.2 Security and Access Control

The primary concern of anyone providing any Internet-wide service is that of security. The security concerns in the *Web //ELLPACK* service include ensuring that users do not get access to files outside of the service boundaries, that they have restricted access outside of their home directory, that

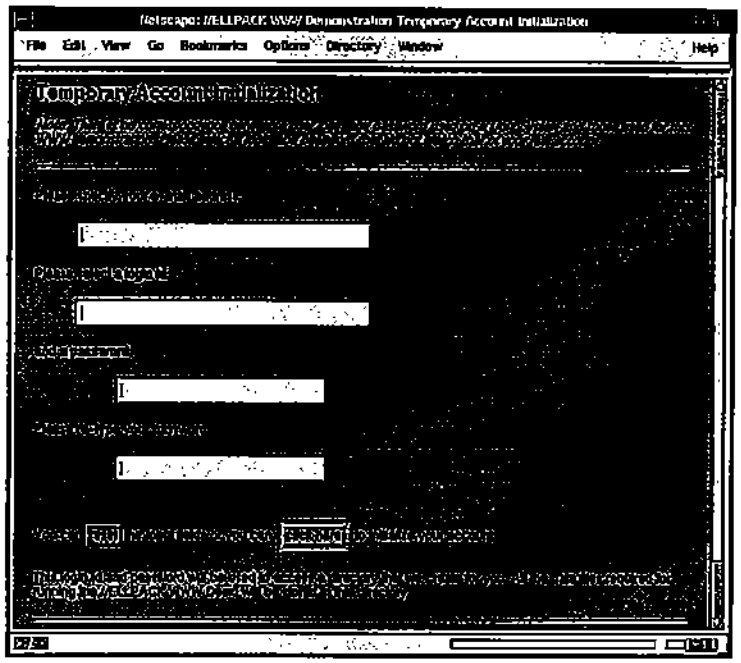

FIGURE 5. The *Web //ELLPACK* account initialization form.

they cannot compromise the system in any way and that they cannot access and compromise the local network in any way.

We ensure that these constraints are maintained by using several levels of security as provided by the Unix operating system. First, the WWW server and the //ELLPACK software is run within a restricted file system. A restricted file system in Unix (created using the *chroot*(2) system call) is a mechanism by which we can isolate a set of directories into a virtual file system. Only files within the restricted environment are visible to any process running inside the virtual file system, while this set of directories is visible as a normal directory for regular processes. We run the WWW server and all of the //ELLPACK software from inside this virtual file system. To accomplish this, it is necessary to have a *copy* of all the software that these processes require within the virtual file system. This is a tedious process as this includes software such as a FORTRAN compiler, basic commands such as */bin/rm* as well as innocuous files such as */dev/zero*. The set of system level files that need to be copied into this environment is extremely ad hoc as there is no listing of them readily available. To make matters worse, this set may differ slightly from one processor to another, even from the same manu-

facturer, as hardware dependent files need to be present as well. Executing within the virtual file system guarantees (in as much as Unix correctly implements the *chroot*(2) system call) that a process running inside the virtual file system cannot look out.

The second objective is to ensure that users can only access their own files and other public files. We implement this by using the Web's access control mechanisms. The HTTP protocol [11] which drives the Web includes a security policy called "Basic" which allows one to require correct username / password pairs before providing access to some file or directory. We use this mechanism to ensure that users have full access only to their own directories and to directories which are not protected by some other username / password pair.

The next level of security is to run the WWW server and the users' //ELLPACK processes as the Unix user *nobody*. This is a Unix login which typically carries no access privileges for any file or resource of any significance. For example, processes running as nobody cannot access the network devices. With this, we ensure that even if a remote user somehow compromises //ELLPACK and access the Unix process running it, then all he or she can do is something which is relatively harmless.

Finally, we can fully isolate the server from other local area network traffic by placing the machine in an isolated subnetwork. This guarantees that even if the user were to break into this machine in some way, he or she would not be able to gather useful information by sniffing the network.

We believe that these measures address the security and access control concerns of *Web //ELLPACK*.

4. Related Work

Placing an X window system program on the Web and allowing others to run it is a fairly routine activity now on the World Wide Web (e.g., [12]). However, providing a networked computing service is not just a matter of placing an executable on the Web. It requires setting up all the policies and mechanisms that we have described above in addition to developing strategies for how to maintain and manage such a service. We have not found any work similar to ours that attempts to provide a regular, long term service rather than a one-time demonstration.

The Matrix Market [13] is a recently announced Web service that allows its uses to browse a database of matrices and to selectively inspect and download them. While this service currently does not provide any direct computation support, we are collaborating with the developers to build a "roving matrix generation" service by developing a series of Java applets for this purpose. This is also a different type of service from the model we present, however, as the service provider simply provides the software (Java applet) which will be downloaded and run on the user's machine.

In some of our own previous work [14] we have developed a system that allows one to build a notebook interface for a problem solving environment using WWW tools. This interface, while not providing a computation service, allows users to access and interact with computation services that are remotely accessible via the Web. The work reported here is complimentary to that as here we are concentrating on providing the networked computing services that tools such as our notebook interface can access and interact with.

There is a large body of work on distributed computing and distributed problem solving. The multitude of conferences, workshops and journals dedicated to these topics serve as evidence. We argue that networked computing differs from traditional distributed computing or distributed problem solving in several ways. First, distributed computing typically deals with computation distributed in a local area. While network technology makes the local vs. remote issue mostly transparent to the programmer and to the user, there are other issues such as accounting and security that are completely different in a global setting. Also, the research issues in building software for locally distributed computation are significantly different from the research issues in building software for globally distributed computation. We outline some of these issues in the next section.

5. Discussion

In this paper we have presented our vision of a new model of computing called Networked Computing as well as a first step we have taken to build such environments. The *Web //ELLPACK* system is now available for public use and we welcome feedback from its users. Several users have located the service even though we have not announced it publicly as yet. There are approximately 20 registered users of which about 10 are active. While the feedback we have received has been generally encouraging, it is clear that most users are not satisfied with the response speed of the software operating

over the Internet. In order to fully realize our vision, it is imperative that the speed of the Internet be at least an order of magnitude faster. Given the amount of commercial interest on the Internet, we fully expect that major improvements in the network infrastructure are forthcoming shortly.

This work represents a step towards realizing the vision of networked computing. There are many benefits to be gained by this mode of operation. We identify the following as being significant:

- *Generality* - any machine connected to the Internet can access a software service without concerns about language or machine compatibility.
- *Interaction* - the user can interact with the software as if it were directly available locally.
- *Access to high performance computers and other special hardware* - unlike today's situation where one needs hardware to be available locally, networked computing allows hardware resources to be consumed remotely as well. While it is possible to access such machines remotely via remote login even today, using them effectively is not convenient due to differences, and hence difficulties, with software setup and management.
- *No code portability problems* - software developers and their customer no longer need to worry about porting program code between different hardware / operating system platforms.

There are several technical and non-technical issues that need to be resolved before the networked computing model can be fully realized. We briefly discuss some of these below.

- *Performance of the user interface*: In order to achieve good performance over the highly loaded Internet, in addition to improving network capacity, it is likely that one will need to better use the network. There is a clear trade off in user interface performance between exporting code to the user's machine and executing code on the server. Our existing prototype (which executes all of its user interface code on the server) shows that communicating each mouse click and key click back to the server for processing provides unsatisfactory interactive performance due to network delays. Our analysis of //ELLPACK indicates that almost all of the interaction can be run locally by exporting a moderate amount of code implemented in an architecture neutral, network transportable programming language such as Java. The user interface, however, does use tools that are both time consuming to execute and which are too large to export. Examples are MAXIMA (used to transform mathematical equations) and domain processors (used to create meshes or grids in

geometric domains). These tools usually require pauses even without a network and the added delay due to networks is likely to be acceptable.

- *Software architecture of networked PSEs*: The performance of the user interface is only one of the aspects of software architecture of networked PSEs that needs to be addressed. Clearly it is necessary to re-engineer existing software before it can be made into a fully functional network service. We are continuing our research in this direction.
- *Dynamically extending system capabilities*: An added twist to networked PSEs is that ideally the remote user should be able to augment the capabilities of the system with his/her own components as well. The //ELLPACK system has this model and we are currently studying ways to allow this facility. In collaboration with researchers at the University of Tennessee, Knoxville and the Oak Ridge National Laboratory, we are experimenting with the NetSolve system [15] as a possible mechanism for implementing this facility.
- *Software ownership and fair use*: We prevent the copying of our software by placing the program code in directories which are not accessible via the Web. However, the question of "fair use" is still murky as we may, in theory, allow thousands of people to use our single machine copy of a privately owned or commercial code. While most current license agreements do not address this issue, we strongly believe that they will soon.
- *Payments for computing services*: The issue of who pays the computing costs needs to be resolved in some way. We currently ignore this issue completely by providing the service free of charge, but its easy to realize that when such services become commonplace that it will be essential for having costing and payment schemes.

We believe that addressing these and other important issues will lead to the creation of networked computing service providers and to the realization of the networked computing model. Given the economic preference for service-oriented schemes as opposed to product-oriented schemes, we believe that this will lead to a better world of software.

6. References

1. T.T. Drashansky, A. Joshi, J.R. Rice, E.N. Houstis, S. Weerawarana, "A MultiAgent Environment for MPSEs," Technical Report CSD-TR-96-013, Department of Computer Sciences, Purdue University, 1996.

2. E. Gallopoulos, E.N. Houstis, J.R. Rice, "Computer as a Thinker/Doer: Problem-Solving Environments for Computational Science," *IEEE Computational Science and Engineering*, **1**(2), 1994, pp. 11-23.

3. S. Weerawarana, "Problem Solving Environments for Partial Differential Equation Based Applications (Ph.D. Thesis)," Technical Report CSD-TR-94-058, Department of Computer Sciences, Purdue University, 1994.

4. About DigiCash - home page, http://www.digicash.nl/digicash/.

5. T. Berners-Lee, R. Cailliau, J.-F. Groff and B. Pollermann, "World Wide Web: The Information Universe," *Electronic Networking: Research, Applications, and Policy*, 2(1), Spring 1992, pp. 52-58.

6. Java: Programming for the Internet, http://www.javasoft.com/.

7. A.P. Black, J. Walpole, "Objects to the rescue! or httpd: the next generation operating system," *Operating Systems Review*, October, 1994, pp. 91-95.

8. S. Weerawarana, E.N. Houstis, J.R. Rice, A.C. Catlin, "//ELLPACK: A System for Simulating Partial Differential Equations," *Proc. IASTED Intl. Conf. on Modelling and Simulation*, Colombo, Sri Lanka, 1995, pp. 122-126.

9. J.R. Rice and R.F. Boisvert, *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, 1985.

10. HyperText Markup Language (HTML): Working and Background Materials, http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html.

11. Hypertext Transfer Protocol -- HTTP/1.0, http://www.w3.org/pub/WWW/Protocols/HTTP1.0/draft-ietf-http-spec.html.

12. An Electronic Primer on Geometric Constraint Solving, http://www/homes/cmh/electrobook/intro.html.

13. Matrix Market, http://math.nist.gov:80/MatrixMarket/.

14. S. Weerawarana, A. Joshi, E.N. Houstis, J.R. Rice, A.C. Catlin, "Notebook Interfaces for Networked Scientific Computing: Design and WWW Implementation," Technical Report CSD-TR-95-048, Department of Computer Sciences, Purdue University, 1995.

15. H. Casanova, J.J. Dongarra, "NetSolve: The Solution Engine (DRAFT)", draft report.