Internet: services and opportunities in the 21st century

Giovanni De Micheli

Giovanni De Micheli, is Professor of Electrical Engineering, and by courtesy, of Computer Science at Stanford University. He received his PhD degree from Berkeley in 1980. His research interests include computer-aided design of integrated circuits an systems.

Internet: services and opportunities in the 21st century

When considering the potential market for networked personnal communications and appliances, a plethora of operating systems needs to be developed.

This will be key to the penetration of Internet.

The last decade of this century has been characterized by a few technology leaps that have had a wide impact on society. The most significant change has been the growth of internet as the principal communication means among individuals. This growth has changed our view of the planet, has opened tremendous possibilities for innovation, and has affected the ordinary life of individuals in technologically-advanced parts of this world. The growth rate of internet makes us believe that its services will reach out soon to most individuals in the world, or at least those having access to a telephone.

To understand the growth of the internet we must trace back to the introduction of the personal computer (PC) in the eighties. Computer manufacturers (e.g., IBM) expected all households to acquire a PC and use it, but this dream was frustrated by the lack of realistic usefulness of computers to ordinary people. Personal computers lacked appealing software, and more importantly, communication. This was especially true for home-based computing.

The computer industry in the eighties strived to reach the convergence of computing, communication and consumer electronics. Advances in semiconductor technologies enabled the design of fast processors which can handle graciously the multi-media applications required by entertainment software, like playing audio and movie files. Advances in communications hardware and software enabled rapid file transfers. But still, at the end of the eighties, information retrieval and search was either a task restricted to a specific domain (and requiring appropriate software layers) or a specialist's task.

The introduction of an hypermedia software to the internet, i.e., the world wide web (WWW) [1] made it for the first time easy to post, search and retrieve information. Shortly later, the development of easy-to-use network browsers, such as Mosaic, Netscape and Explorer, solved the problem of enabling an ordinary individual to navigate the network. It is fair to say that Minitel had reached a similar objective many years before the world wide web. But unfortunately Minitel's impact outside France was very limited. Morover, the most noticeable motivation for users of the world wide web is its free access.

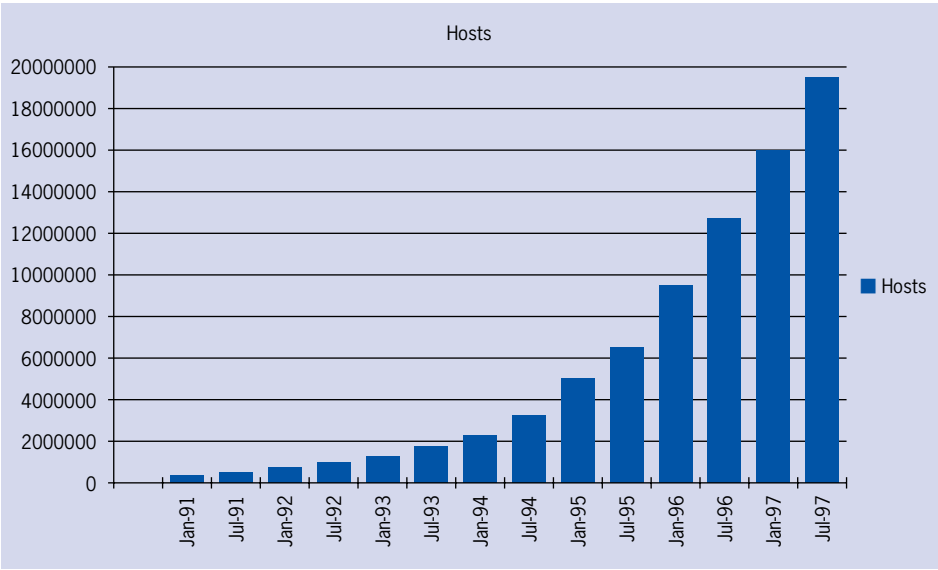

Figure 1 - The world-wide growth of internet sites.

Electronic mass media and commerce

The explosion of the use of the world wide web lead immediately to the problem of connecting the information provider with the information requestor. Search engines filled this gap. The first search engines, such as Lycos and Yahoo, were developed in the 1995 timeframe. The developers, who understood the critical need of the engines for enabling internet surfing, had two fundamental questions to solve. The first problem was technical, and related to the search procedure: search by pattern matching and affinity, or search by cataloging WWW sites. Both solutions turned out to be winners. The second problem was the business model. Should the search provide revenue or should the search be for free? The key to the success of Yahoo (today's most visited WWW site) was basing the company revenue model on advertisement, and enabling all searches to be free. Thus we can consider Yahoo as a media company.

Overall, the growth of the use of internet has been fueled by two major reasons: electronic entertainment and commerce. Indeed, much of today's use of the internet is for information retrieval: we can access news (newspapers, radio and TV stations, weather) and entertainment, technical information and documentation, and a wide variety of other types of text and hypermedia files. Thus many services provided by the interneat are comparable to, or rival, those provided by standard mass media. This market is potentially very large. Moreover the ubiquitous presence and unregulated structure of the WWW makes available pieces of information in regions where traditionally this information was hard to obtain (e.g., foreign press and radio).

Electronic commerce, i.e., running business on-line, is the second largest factor responsible for internet users' growth. The internet can bring down physical barriers to commerce, such as distance and size of business. At the same times, geographically-distributed customers can find it attractive to shop on-line. It has been reported in January 1998 that the top 52 electronic marketers have had total revenues of US7.3 billion in 1997. At present, this market is dominated by providers of electronic goods (e.g., Dell Computers, Cisco), but the general public has been shopping on line significantly for products like software, books, flowers and music. Electronic commerce is receiving wide interest in the technologically-advanced countries as well as in the developing countries. A UN-based initiative, the global trade-point network, has a program aimed at developing world-wide electronic market, and specifically at helping developing countries by providing local interface to internet, software and connectivity.

Interesting opportunities and challenges surround electronic commerce, for example in the domain of payments. While on-line payment with credit/debit and charge cards is accepted, as well as home banking and electronic transactions on the internet, there is a need of providing a soft equivalent of cash, with the corresponding characteristics of liquidity and anonymity. Experiments on electronic cash (e.g., Digicash) have shown the viability of this approach. If eventually electronic cash is universally accepted, it will require structural changes in the market economics.

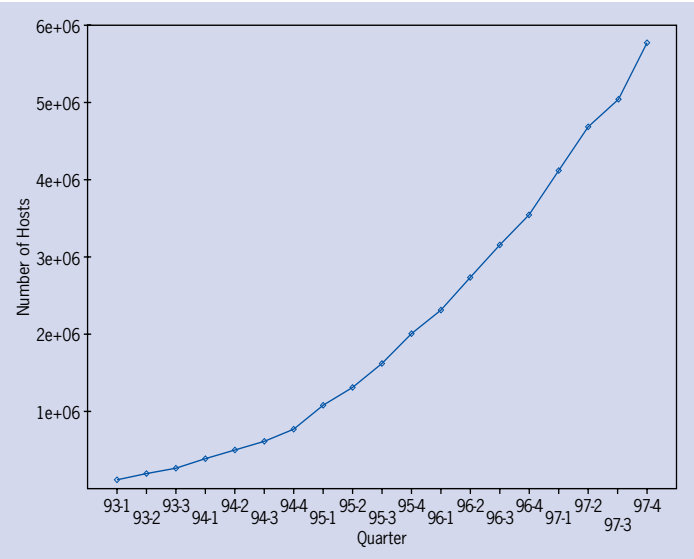

Figure 2 - The growth of internet sites in Europe.

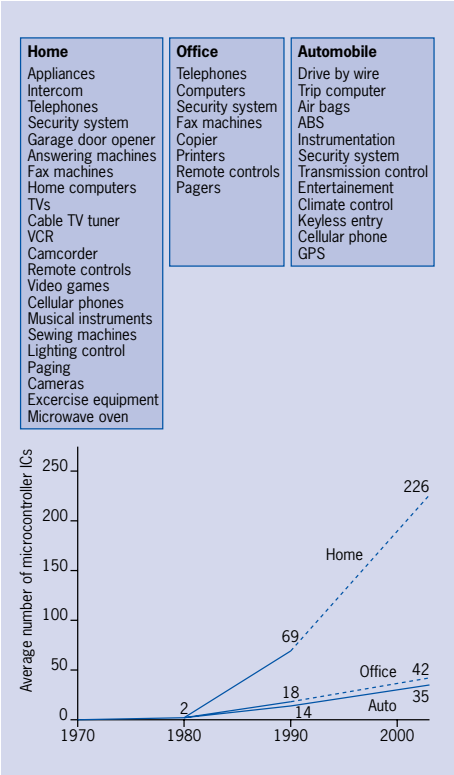

Figure 3 - Penetration of embedded systems.

Embedded systems and networked appliances

Most processors and controllers find their use today in embedded applications. Home, offices and vehicles see an increasing number of electronic controllers with embedded software, (see Figure 3). Today, most of these systems are not networked, except for home and office computing and communication systems. Nevertheless, networking will spread soon to most electronically-controlled devices in the home, office and car.

Networked appliances are becoming a reality for several reasons. Most of these appliances rely on software for high-level control. Networking allows their cooperative operation, distance monitoring and ease of maintenance. For example, electric home appliances can cooperate with the goal of keeping the overall electric wattage below a given threshold (for cost reasons). Home and office monitoring has several advantages, but their usefulness increases when a common shared interface, like a network browser accessible ubiquitously, can be used for monitoring and control. Maintenance of networked appliances becomes simpler, because of remote monitoring of malfunctioning, as well as software upgrades that can extend the hardware life while giving the manufacturer the chance of profiting by selective upgrades.

Car navigation systems connected to the internet are popular in Japan, because they give the driver the opportunity of monitoring the traffic and selecting adaptively the best route. For these system to operate, the network has to provide various types of information, such as traffic congestions, police directives, alternate routes.

Collaborative design environments

Given the fact that the world wide web has enabled users to retrieve information ubiquitously, the next step is to research paradigms for processing information in a distributed environment. Distributed computing has been in existence for many years, but the WWW enables the seamless cooperation of software applications running on different platforms toward a common goal. Applications areas may be several, and the solutions may be purposely different. Indeed the WWW provides a support for integration of software applications which is independent of the hardware platform and the operating system. Thus, the embodyments of distributed information retrieval and processing systems may be varied.

We consider in this section a case study for an engineering application of distributed computing: the design process of complex (electronic) systems requiring specialized computer-aided design (CAD) tools. This theme is subject of active research in the USA. Figure 4 shows linkage between different sites providing experimental distributed CAD systems under the auspices of DOD/ARPA.

The wide diffusion of the WWW [1] has prompted several attempts directed to exploit its features in the CAD area. Most approaches exploit the the information-centric perspective of the WWW, addressing the designers' need for accessing geographically disperse and heterogeneous information. The designer can retrieve background material (i.e. algorithms, bibliography, benchmarking information, etc.) and design libraries

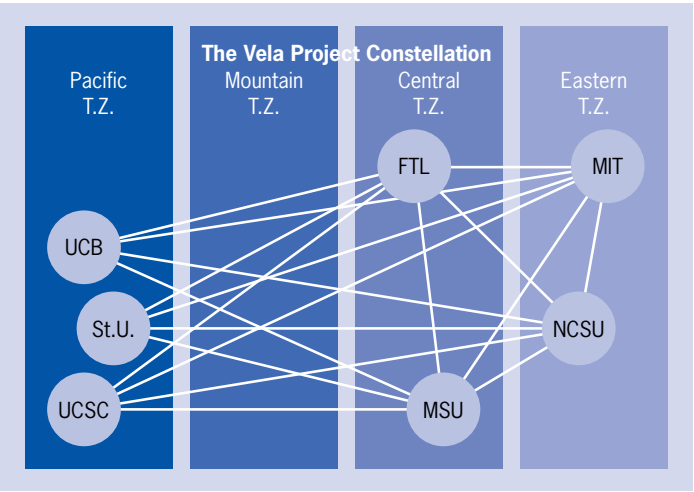

Figure 4 - The sites of the VELA project.

simply by activating a hyperlink on a WWW page. Multiple data formats can be transferred and viewed without the need of complex interactions. Security issues are addressed and privacy is guaranteed by encrypted transactions. The complexity of the communication among remote sites is hidden, and the exchange of data among users becomes straightforward.

Moreover design teams are often geographically dispersed, even though they may be working on different parts or facets of the same design. Design reuse of libraries, hard/soft macros, and embedded software may require access to resources that are also geographically dispersed. Fortunately, computer networks such as intranets and the internet provide pervasive reliable links among designers and design data, as well as useful means of storing and classifying information. In addition, network browsers and helpers have simplified the access to remote and distributed information, by providing a uniform mechanism which is easy to learn.

Designers communicate with the tools through a WWW browser, and specialized helpers. Thus the learning-curve for engineers is shortened, as compared to using specialized tools. More importantly, designers need not only to access information and data, but mainly they need to process and modify the data, by executing the available CAD programs. This puts us in an application-centric perspective, where the network and its software become the design environment.

Distributed computer-aided design offers several advantages over traditional, workstation seat-based CAD. First, a designer can access a design system using a computer platform different from that used to run the tools and use a uniform interface to design tools provided by a browser. The designer acts as a client using possibly a low-cost entry system, whereas tool servers provide remote execution of tools. Tool licenses can be reduced to those machines dedicated to the corresponding service. Load balancing across the network can reduce response time and increase designer productivity. Second, CAD tool researchers can be more inclined to perform collaborative research, and to link their tools to those of others who may not be co-located. Interfacing and sharing data and experiences in collaborative research is simplified by having the same interface. Benchmarking and comparisons could become much easier and the dissemination of relevant ideas, even at prototyping stage, could be enhanced. Third, both academic and industrial tool developers could make their products available to the designers' community with the goal of promoting new ideas and methodologies. Academic work would gain larger exposure, while commercial ventures may use remote tool access as a way of promoting commercial products. Last but not least, new commercial paradigms may arise in the CAD industry. Designers could temporarily connect to a tool provider, perform a specific task and be billed on a usage-time basis. Tool providers may be in turn the tool developing companies or other design service brokers.

Simple protocols for remote execution have been described in [10] and [9]. Both approaches focused on batch execution: the user can request services to remote tools, that will perform the required tasks and return the results. The interaction between user and tools is sparse and loose. Effective execution of remote tools requires a much higher level of interaction: the user should be allowed to change the setting of a parameterized tool run, preview partial results, access visual information like waveforms and network schematics, modify networks and update design databases. A few independent research efforts are currently focusing both on the development of new software technologies enabling advanced web-based execution paradigms [5] [4] [8] and on their application to the electronic design automation [12] [6]. The next subsection describes an implementation of a distributed CAD environment at Stanford University.

Case study: a distributed design system

We have developed PPP, an integrated CAD environment with a highly-interactive WWW-based interface. PPP is a distributed system, that links several tools for the synthesis and simulation of low-power CMOS circuits and make them uniformly accessible from the WWW [2]. Internet users may access PPP using their own WWW browsers. No software installation is required and the interface is the same used for WWW navigation.

The primary target of the PPP project is to develop a fully integrated synthesis and simulation environment with short learning curve and architecture-independent WWW-based interface. PPP is a modular system composed of interacting tools that may run on different machines and be provided by different vendors or contributors. The user accesses PPP using his/her own WWW browser, with no additional requirements on hardware or software installation. The tools can reside on remote servers or on the local users' machine. This is fully transparent to the user. The graphical user interface (GUI) of PPP is exactly the same used for WWW navigation and no additional effort needs to be spent in familiarizing with a new GUI when the user first accesses the system.

In the design of PPP, we focused on two key ideas: plug-and-play modularity and architecture-independent interactive remote execution. Plug-and-play modularity allows new tools to be integrated in PPP with little effort and no modifications. The impact on other tools already embedded in the environment is negligible. Remote execution can be seen as an advanced caching strategy. Upon invocation, a resource in PPP can maintain its state throughout a sequence of interactions. The state information is not necessarily saved on files to avoid the penalty of iterated disk accesses (or, worse, of sending data across the internet), but it is kept in memory because the resource does not terminate its execution after every interaction with the user.

From the user's point of view, PPP appears as a remote HTTP server and each user can access it by simply inserting a pointer to the PPP URL in his/her bookmark file. This is the only setup operation required. Figure 5 represents the user interaction with PPP. The rectangles represent different machines connected to the internet. PPP is the core of a distributed architecture. It manages two different kinds of interfaces: connection to the client and connection to the resources (tools and databases). The arrows represent internet protocols (FTP, HTTP, mail, etc).

Multiple users are allowed to concurrently access PPP, similarly to a traditional WWW server. Figure 5 depicts a centralized structure. The PPP kernel manages all interactions between the users and the tools.

PPP is build upon a layered architecture, following a well-known software engineering paradigm. The three layers are: application, communication and interface. The organization of PPP is shown in Figure 6. The top two layers are embedded in the kernel, while the application layer is distributed. Application is at the bottom of the hierarchy and manages the interaction with the tools and the information sources.

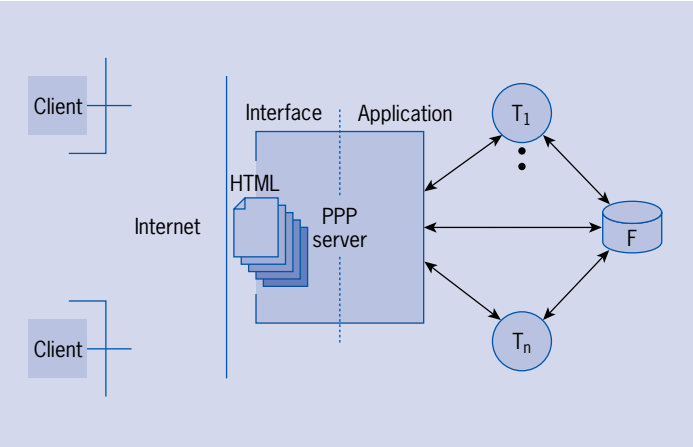

Figure 5 - The client-serveur model of the PPP environment.

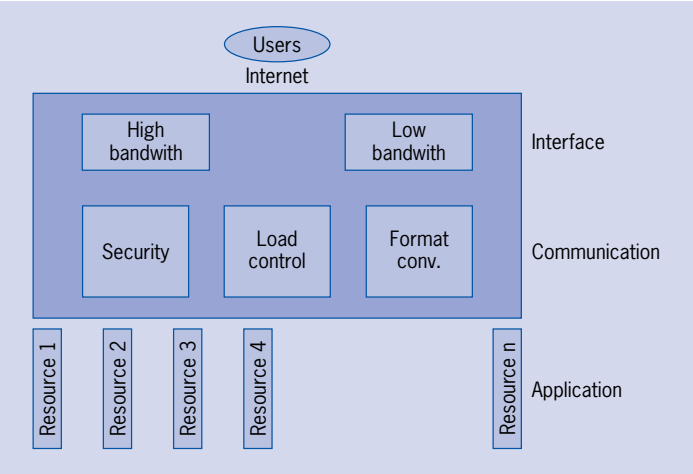

Figure 6 - Software layers in PPP.

Application layer

The synthesis and simulation tools embedded in PPP represent the bulk of the application layer and are called resources. No constraint is posed on the target architecture and on the programming language used for the implementation of the resources. The resources may run on different machines. Two levels of interaction between resources and the above layers are possible. The simplest interaction is stateless connection: the tool performs some manipulation on the input data, produces an output and terminates the execution. Information about previous invocations may be stored in files. Unfortunately, retrieving the state from files imposes heavy performance penalties when highly interactive operation is required (for example, during an user-controlled optimization session). Stateless connection resembles the batch execution features described in [10] [9].

For applications where higher interactivity is required, we provide a novel paradigm called interactive remote execution. Once started, the resource does not terminate its execution after every command, but it simply waits for another command to be issued. The resource becomes a server that awaits for service requests from the client (the user) keeping complete information about past requests and their results. Resources that provide interactive remote execution must satisfy some requirements on their external interface. In particular, it must be possible to interactively receive data from another process without terminating the execution. Interactive remote execution is required for running optimization and validation tools that perform user-assisted tasks.

Communication layer

The communication layer represents the core of the PPP architecture. Its purposes are the following:

- process users' requests and translate them in a format that is accepted by the resources;
- collect the results produced by the resources and provide them to the users in a readable/downloadable format;

■ manage data files provided by the users and/or created by the resources and store them where retrieval will be faster;
■ provide security. Different users work on private space and should not be allowed to interact unless they explicitly require to work on a shared design. Access identification should be enforced;
■ manage server's resources. If PPP runs on multiple machines, the communication layer can direct the requests of the users toward unloaded machines.

All functionalities listed above are in some degree implemented in PPP. Each user works in a separate space, where input files and results are stored. User access authentication is enforced by a password mechanism. All format conversions are performed transparently. Since the resources run on different machines, the communication layer must be implemented as a distributed multiprocess application. In the current implementation, scheduling of resources, conflicts and mutual exclusion are managed using simple message files on a shared file system.

Interface layer

Interface is the uppermost layer, with which the user interacts. The largest part of the interface layer is directly supported by the WWW browser. If the interaction is limited to simple menu-based communication, the form feature provided by HTML [3] is sufficient. The same holds for small images and graphs. This is however not sufficient in general. In many cases, the user wants to specify the input or examine the output in more complex ways.

We call high-bandwidth the kind of interaction for which the WWW browser does not provide satisfying performance. Examples of high-bandwidth interactions are transfers of large input files (i.e. the networks that the user wants to simulate/optimize) and interactive graphical output (such as waveform display or network browsing). In these cases alternate mechanisms for interaction must be provided. For user-originated communication such as the transfer of input files, we currently use the FTP protocol.

For resource-originated communication, such as waveform display, more advanced mechanisms are needed. The resource generates output data files, the communication layer then takes care of transforming the data in a graphical output that is sent to the user. The user can view the graphical output using helper programs or directly through the limited image display capability of the WWW browser.

Interactive remote execution

When interactive remote execution is required, the process of linking the tool to PPP is less straightforward. The standard input and output channels must be re-directed to a script that manages the interaction with the interface layer. The connection with the tool is established when the user selects the interactive logic optimization option. Upon connection, on the server side the tool is started and it remains idle waiting for the user's commands. On the client side a form is displayed to allow the user to enter commands.

Once connection is established, the communication between user and tool follows the interactive remote execution paradigm. When the user types in the commands and issues them, on the server side a script is executed that parses the commands and submits them to the tool. When the tool completes the command execution, it returns diagnostic messages to the script that translates them in HTML format and forwards them to the client side. It is important to observe that the tool does not terminate execution when a command completes. Hence all data structures and current execution status are kept in memory on the machine running the tool. Moreover, since the execution is interactive, the user is not allowed to send new commands to the server until the last issued command has completed.

Interactive remote execution is efficient when the user needs to issue a large number of commands with short execution time, and wants to take decision based on the results of the commands. Batch execution is more suitable for a usage pattern based on long-executing commands that do not require user intervention. For instance, interactive remote execution may be preferred when the user is customizing and experimenting with the optimization scripts. Once the best optimization strategy has been chosen, it is advisable to bundle all commands in a single script and run it in batch mode.

Bandwidth management

A fundamental issue in the design of all layers of PPP is bandwidth management. Although local area networks (LAN) can usually provide the bandwidth required for most data transfers, PPP is not limited to run on LANs. When some of the connections are supported by a wide area network (WAN), the bandwidth of the data transfer has to be reduced as much as possible. In the current implementation, the only critical connection is between the PPP kernel and the client. On this connection, bandwidth-critical applications are those involving massive data transfer and a high degree of interactivity. In this case, the level of interaction with the resource is so high that we may want to transfer the resource itself on the client's side.

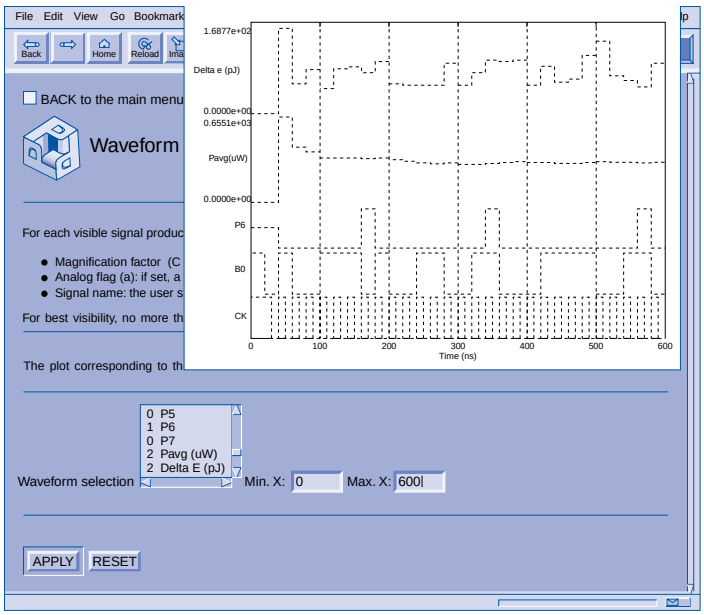

Figure 7 - Waveform display in PPP.

An example of critical application is waveform display (see Figure 7). A solution where the client receives compressed images of the waveforms is satisfactory only if the user does not need to dynamically update the waveform view very frequently. The advantage of this approach is that the user does not need any dedicated support for waveform display.

More aggressive bandwidth reduction can be achieved without requiring the user to install dedicated programs. To this purpose, it is necessary to enable the transmission of executable code across the network. Languages such as Java [5] and TCL [11] allow this kind of interaction.

These considerations leads us to envision an engineering CAD system as consisting of two type of tools: point tools and service tools. Point tools are specific high-valued tools that perform specific tasks, run on dedicated machines and usually execute computing-intensive tasks. Service tools are low-valued tools, that should be portable and thus best implemented as applets that can be downloaded on the client to better serve the user. All application-specific helpers should then be portable, cost-free tools.

Conclusions

The internet provides the basic layer for communication, for enabling distance monitoring and control, as well as for distributed data acquisition and computing. When considering personal computers and software applications running on internet, it is immediate to notice that a gracious interaction of the operating system with the software applications is key to their effective and successful implementation. Thus, providers of widespread operating systems for personal computers will be able to control the software market for WWW applications, because they can control the way in which applications interact with the software layers closer to the hardware itself. A monopoly situation in operating systems will affect adversely the development of novel applications on the WWW, because it will suffocate deviation from mainstream business models and will marginalize applications that do not blend with the operating system itself.

On the other hand, the growth potential for internet goes beyond connecting personal computers. When considering the potential market for networked personal communicators and appliances, a plethora of operating systems with different characteristics needs to be considered.

Some of the functions typical of the operating systems will be best implemented as newtwork software, and eventually operating system and newtwork software design will blend.

In conclusion, the know-how and control of network hardware and software will be key to the development of internet software and to its penetration.

Reference

[1] T. Berners-Lee et al., “The World Wide Web,” Communications of the ACM, vol. 37, no. 8, pp. 76-82, 1994.

[2] A. Bogliolo, L. Benini, G. De Micheli and B. Ricco, “Gate-Level Power and Current Simulation of CMOS Integrated Circuits,” IEEE Transactions on VLSI, Vol.5, No.4, December 1997, pp.473-488.

[3] “HTML Working and Background Materials,” http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html.

[4] CDR, Stanford University, “JATLite,” http://java.stanford.edu/java_agent/html/, 1997.

[5] Sun Microsystems, “The Java language environment: white paper,” http://java.stanford.edu/java_agent/html/, 1997.

[6] H. Lavana, A. Khetawat, F. Brglez and K. Kozminski, “Executable Workflows: A Paradigm for Collaborative Design on the Internet,” 34rd Design Automation Conference, 1997.

[7] D. Lidsky and J. Rabaey, “Early Power Exploration - A World Wide Web Application,” 33rd Design Automation Conference, 1996.

[8] C. J. Petrie, “Agent-Based Engineering, the Web, and Intelligence,” IEEE Exp. Intelligent Systems and their Appl., December 1996.

[9] P. G. Ploger et al., “WWW Based Structuring of codesigns,” International Symposium on Systems Synthesis, pp. 138-143, 1995.

[10] M. J. Silva and R. H. Katz, “The case for design using the World Wide Web,” 32nd Design Automation Conference, pp. 579-585, 1995.

[11] J. K. Ousterhout, “Tcl and the Tk toolkit,” Addison-Wesley, 1994.

[12] The WELD Group, University of California, Berkeley, “WELD project: Web-Based Electronic Design”.