Web-Enabled DSP/Microcontroller Simulators

Project Report

Chuanjun Wang

chjwang@cs.utexas.edu

http://www.cs.utexas.edu/users/chjwang/

EE382c - Embedded Software Systems

Dr. Brian Evans

Spring 1998

Abstract

The Web-Enabled Simulation (WEDS) framework from the University of Texas at Austin complements the Web-Enabled Electronic Design(WELD) system at the University of California at Berkeley by adding support for the simulation, debugging, and design of software for embedded processors. WEDS is a client/server framework that current offers interfaces to one microcontroller simulator, two digital signal processor (DSP) simulators, and one DSP board. WEDS consists of GUI Java applets running on the client side, a multi-threaded TCP/IP communication server (Java application) running on the server, and a group of simulators and debuggers for DSPs and microcontrollers. WEDS makes use of sockets, multiple threads, and Unix interprocess communication. In this project, we expand WEDS by redesigning the Java application server and the protocol between Java clients and the server, providing security and privacy by proving user file protection on the server and using authentication and RSA encryption/decryption for communication, and dynamically configuring both the server and its clients. The WEDS framework's flexibility, scalability, portability and security are greatly enhanced.

1 Introduction

Computer-Aided Design (CAD) tools are essential for designing integrated circuits and their importance is increasing as circuit designs become more and more complex. With the development of microelectronics technology like deep sub-micron technology, new Electronical Design Automation(EDA) tools and methodologies to reduce product development time and cost are emerging all the time. Most of these tools are currently very loosely coupled.

However, increased chip densities and the design of higher performance systems have outgrown the current era of loose collections of EDA tools. Keeping pace with the advancement and the required designer productivity improvements will necessitate cooperative work across the industry toward creating a highly scalable and robust environment in order to meet EDA system needs in the area of designer productivity and design complexity management.

Distributed computing, Client/Server technology, and the tremendous growth of Internet bring us opportunities to solve this problem. Now the World Wide Web(WWW) has changed and is continuing changing many aspects of our life. The natural features of the web like scalability, platform independence, portability, timely distribution, robustness, and flexibility are perfect fits for the new needs of EDA. The great advantages of putting system level design tools on the web has already been realized by some CAD tools vendors and researchers. Web-Enabled Electronic Design (WELD) at the University of California at Berkeley aims to construct the first operational prototype of a national-scale CAD design environment enabling Internet-wide IC design for the U.S. electronics industry [1]. In the small, WELD will empower individual American electronics designers by affording them efficient desktop access to, and seamless interoperability of, the numerous, heterogeneous resources forming a national scale electronics design system built upon the National Information Infrastructure. In the large, WELD will reduce electronics industry

market entry barriers to new entrepreneurs by providing a streamlined pay-per-use design development environment and a robust software distribution infrastructure. In reducing the costs, and shortening the time-to-market of new intellectual capital, WELD is expected to stimulate the whole U.S. electronics industry to dramatic new growth.

The system of WELD consists of three parts: remote servers, network services, and clients. The clients communicate with communication server which provides network services. And the communication server dispatches the requests of clients to remote servers over the the distributed environment or on the same machine.

There are three great challenges for WELD to reach its goals. They are:

1. scalability, including both tools and user scalability
2. security, privacy and fairness
3. reliability, robustness and quality of services.

In our project, Web-Enabled Simulation(WEDS) [2], we aim to establish a scalable and secure client/server framework on the Internet that can offer interfaces to some microcontroller simulators, digital signal processor (DSP) simulators, and DSP board simulators. World Wide Web is the most popular part of the Internet right now. So it becomes our natural platform to put those simulators on line.

Basically the DSP/microcontroller Simulator is a software tool for developing programs and algorithms for DSP/microcontrollers. The simulator exactly duplicates the functions of supported DSP/microcontroller chips, usually including all on-chip peripheral operations, all memory and register updates associated with program code execution, and all exception processing activity. The device's pipelined bus activity is exactly simulated. This enables the Simulator to provide the user an accurate measurement of code execution time, which is so critical in DSP applications.

Web-Based DSP/microcontroller Simulator/Debuggers provides WEB access to DSP/microcontroller simulators/debuggers. You can profile and validate embedded software, learn DSP/microcontroller architectures, evaluate different DSP/microcontroller architectures, access latest technology immediately since it is on the WEB, simulate the whole system, and reduce the workstation expenses for dedicated DSP/microcontroller simulators. All you need is a desktop machine, a Java enabled browser and Internet connections.

Most of the challenges faced by WELD are also the greatest challenges we need to address. Our goal is to maximize the configurability, flexibility, scalability and security. Because our system is not so complicated as WELD, those issues faced by WELD are relatively easier to deal with. The following challenges are specific to our Web-Based DSP/microcontroller Simulator/Debuggers system:

1. portability limited by Java. Java itself is always changing. The current version is not even compatible with the previous versions.
2. efficiency limited by Java Virtual Machine and WEB browser.
3. security also limited by Java's security architecture. Java applets do not have privileges to access local file system. And Java applets can only establish one communicate with the server where it comes from.
4. privacy. We will provide protection between user files. Both authentication and encryption are necessary.
5. configurability and scalability depend on how well the embedded processors can be modeled by some simple configuration files and how common these processors are.

2 Objectives

Our long term goal is to establish a flexible, scalable and secure client/server framework on the Internet that can offer interfaces to as many as possible microcontroller simulators, digital signal processor (DSP) simulators, and DSP board simulators, or other embedded processors, and possibly to integrate into the WELD project. Our immediate objective is to make the current framework more flexible, scalable, configurable, portable and secure.

To fulfill this objective, we need to provide portability, authentication, privacy, encryption/decryption to both the client users and the server.

For the consideration of portability and user friendliness, we choose Java for most of the programming. As long as the users have Java-enabled WEB browser, the client side program can be executed. For the server side, it is also easy to port the server program to other platforms as long as they support Java Virtual Machine. So by choosing Java, the code itself will be quite portable.

For scalability, first, on the server side, it is easy to expand the set of DSP/microcontroller simulators/debuggers being supported since all the simulators/debuggers are standalone and separate from the communication server. And, also on the server side, the server can configure itself by checking resources available, security package installed, etc. Second, on the client side, since we developed a protocol to configure the services available on the server and communicate this information to the client side at the beginning when the client sends its query to the server, the client program doesn't need to make any modification. It will be automatically configured.

For security and privacy, we need authentication for client connection requests, encryption for all the data flowing between clients and server, protection for both user files on the server and server's own private files. We also set up quota/limitation for users' files.

3 Design, Implementation & Contribution

The Web-Based DSP/microcontroller Simulators/Debuggers system consists of GUI Java applets running on the client side, a multithreaded TCP/IP communication server(Java application) running on the server, a group of simulators and debuggers for DSPs, microcontrollers and/or other boards, and layers of security, as illustrated in Figure 1.

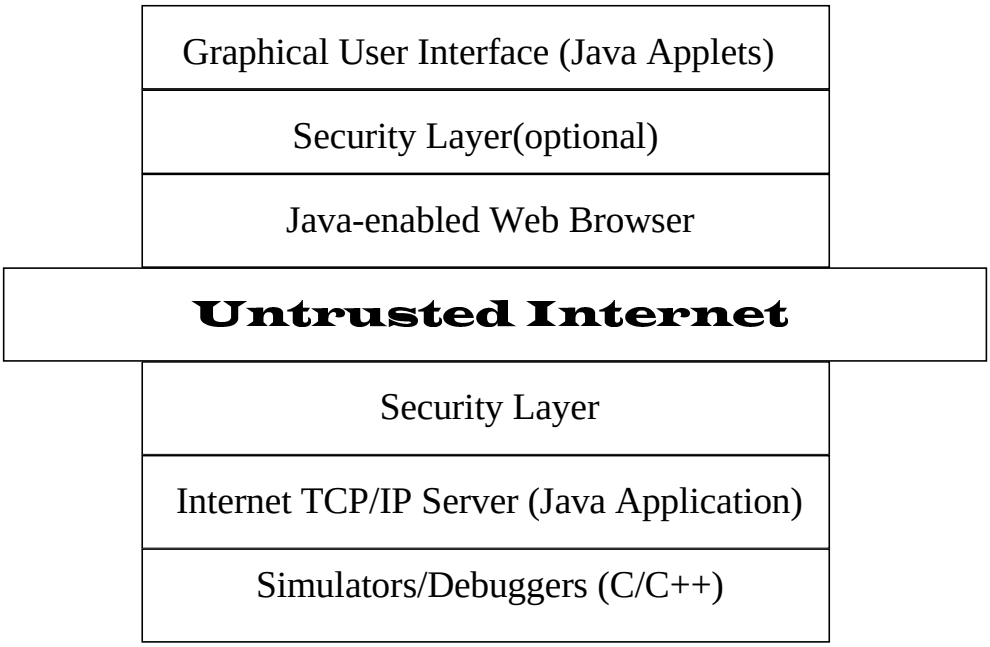

Figure 1: Architecture of WEDS

The clients(Java applets) and the server(Java application) basically follow the Client/Server model. Server program continuously listens for connections requests from clients. A typical track of one execution experienced by the client looks like this:

1. First a client will send a packet containing information about its public key, version information, etc. to the server.
2. After server gets this information, it will reply to the client by sending a message of the same structure, informing the client its public key and other information, and it starts to challenge the client, asking his/her user name, password. At this point, both the client and the server know about each other's public key. But the private keys are kept secret forever.

3. The client sends his/her user name/password pair to the server in an encrypted form, which is actually encrypted by the RSA [3] algorithm using the server's public key. So although maybe anybody can eavesdrop this message, only the server can decrypt it.
4. After the server gets the user name/password pair, it decrypted that with the its private key. So the original data is recovered.
5. The server checks to see if the user name/password pair has a match in its accepted user name/password table. If it does, the connection request is accepted. Otherwise the client is not a legal user.
6. The server responds to the client by accepting or denying entrance. Also, this message is encrypted by the client's public key, which it gets from the first packet from the client.
7. The client decrypted the message received using the its own private key.
8. If the client is granted entrance, then it will send a query to the server, querying what kind of services are available on the server. Also, this query message is encrypted the same way as above.
9. The server responds by sending the list of available services to the client, and of course in encrypted form.
10. The client configures itself(GUI setting up) by the information it gets from the server. And then it sends a request of a certain DSP Simulator/Debugger/Board to the server.
11. The server initiates an instance of the DSP Simulator/Debugger/Board requested by the client. And it is ready to accept commands issued by the client program.
12. The client user is free to upload files on the server(limited by quota). And it can retrieve his/her own files next time when he/she logs in.

The implementation is based on previous work. The client GUI doesn't change much. I inserted in a process of user name/password checking. Now all the data streams between the client and the

server are encrypted except the first message, which is plain text exchanging each other's public key information. I renovated the server, redesigned the communication protocol. On the server side, when a new connection request comes in, it can be of one of the following four:

1. a login connection: waiting for user/passwd checking
2. a query about what kind of services available on the server
3. a DSP Simulator/Debugger/Board request
4. a security information exchange

The format is like the following. For the first three, the message is: a integer value of CHECK_IN, GET_RESOURCES, or simulator ID, flollowed by a tab '\t', and a string end with '\n'. For security information exchange, the format is: a integer of value SECURITY_INFO, followed by a tab '\t', and a serializable object containing using/not-using encryption, public key, version, etc.

Dynamic configurations on both server and client sides are implemented. The server configure itself by reading configuration files(resources configuration, user name/password files). Applets configure themselves based on feedback from the server. Security(data encryption/decryption) on the client side is optional since the client may not have JCE 1.2 installed. The Java applet will detect whether JCE is available. So almost no hard coded local information is contained in the source code. Also, simulators can be added without changing any application code. Great flexibility, scalability, and extensibilty are achieved.

Authentication, user file protection, security are also implemented. Security features include authentication(password checking), file protection(both users and server), data encryption/decryption. RSA [3] of 1024-bit key length is used. And the key is dynamically generated for each communication session. So key hacking is almost fruitless.

We also renovated the communication protocols between the client and the server. Four types of messages are communicated in the system. Initialization, which basically exchanges public keys

between clients & server, happens at the very beginning when a client starts to communicate with the server. Authentication, which is actually user name/password pair matching, happens right after initialization. Also, immediately after authentication, the server sets up user programs directory and exerts protection on user files. Configuration, which includes client querying about available services and the server replying, is used for the client to configure itself. Last is data exchange, which are actually the data input/output streams, can be cipher input/output streams or just plain data based on the configuration.

Dynamic configurations are tested by appletviewer. Authentication is also tested by appletviewer. Encryption needs Java Cryptography Extension(JCE) 1.2, which needs Java Development Kit(JDK) 1.2. The condition is not matured right now. No testing on encryption is done. But the framework set up for RSA encryption is correct and quite flexible, which means it is easy to replace it with other security package.

4 Conclusion

WEDS consists of graphical user interfaces(Java applets), an Internet server(Java application), standalone command-line tools(C/C++), a security layer to do communication over the Internet. The framework is configurable, portable, secure, extensible, and freely distributable.

5 References

[1] Richard A. Newton, "Web-Based Electronic Design (WELD) Project," URL: "http://www-cad.eecs.berkeley.edu/Respep/Research/weld/", Dept. of Electrical Eng. and Comp. Sciences, The University of California, Berkeley, CA, 94720.

[2] B. L. Evans, "Web-based Simulators of Embedded Software for Programmable Digital Signal Processors," URL: "http://www.ece.utexas.edu/~bevans/talks/WebSimulatorTools/", March 14, 1997.

[3] Vijay Ahuja, *Network and Internet Security*, Academic Press, ISBN 0-12-045595-1, 1996.