

# Technology Integrated Learning Environment—A Web-based Distance Learning System

**Regina Gehne, Chris Jesshope and Jenny Zhang**

New Zealand Educational Software  
Massey University, New Zealand

Tel: +64 (0)6 350 5799 xtn 2467 fax: +64 (0)6 350 2259

Email: [r.gehne@massey.ac.nz](mailto:r.gehne@massey.ac.nz), [c.r.jesshope@massey.ac.nz](mailto:c.r.jesshope@massey.ac.nz), [j.zhang@massey.ac.nz](mailto:j.zhang@massey.ac.nz)

## Abstract

*This paper analyses the requirements of a third-generation technology integrated learning environment. The architecture proposed is analysed in terms of the generic constraints of a general distributed system, such as performance, scalability and security issues. We have also considered the pedagogical issues and the modes of use in which a student is likely to access the educational material. The result of this analysis is an architecture that will scale well with numbers of students and will be able to support a variety of emerging pedagogical models.*

**Keywords:** on-line education delivery, client server architecture, network protocols.

## 1 Introduction

There seems to be no doubt that we are facing a future, where globalisation and the application of technology will become major factors in our educational systems. This is not only going to affect our children, but also the current generation of adults, who have already been trained and are in full- or part-time employment. The world we live in is changing rapidly, companies are experiencing faster and faster product development cycles and must, in order to sustain a competitive advantage, embrace the continual training of their staff. There also seems to be a change in people's attitudes that is perhaps a result of this same globalisation. They have come to expect immediacy as the norm in whatever they are doing. These factors all contribute to a picture of a rapidly changing workforce in terms of the required knowledge and skills, and where our future employment will increasingly rely on just-in-time learning, delivered to the workplace or to the home by means of sophisticated technological solutions.

There are already a very large number of different systems available to deliver on-line education. Rather than list them all, the interested reader is referred to a recent and very comprehensive comparative analysis (see: <http://www.ctt.bc.ca/landonline/choices.html>), which compares 43 tools against a similar number of features. If we consider the web-browser as the first-generation of web-based educational delivery tools, then these tools might be considered as second-generation tools. Many have evolved from universities' local solutions to the campus-wide delivery of on-line education. However, these tools

have not really considered the implications of globalisation, which requires dependable and scalable distributed systems. Neither do these solutions offer a great deal of flexibility in delivery to the student.

If we were to define a third-generation learning environment, then it must match the future we have outlined above and provide a robust and scalable solution to education delivery. Moreover, it must also meet the flexibility requirements of this new generation of learners, who will not be confined to our university campuses. We have tried to embrace these goals in the Technology Integrated Learning Environment (TILE) project [1].

Let us consider the student's requirements first. The student may wish to study in a number of different situations:

- at home using their home computer - we assume that they have on-line access but that there is a finite cost for that access (it may be a time-based cost or, in the future, a packet-based cost where they are on-line continuously)
- while travelling, with a laptop or similar portable computer - in this situation the student may have no access whatsoever to an on-line connection, or if they do it may be very expensive.
- at a conference, an internet café or in a laboratory, using a computer, which they do not own and may not install software on.

These situations lead to conflicting requirements for browsing the educational material.

Ideally, in order to provide scalability, we must be able to distribute material to the student cheaply, whatever the number of student's studying a particular course. Using just on-line data delivery, scalability means providing faster and faster servers and eventually creating networks of servers, with their associated cost. The computer scientist E.E. Dykstra once said of computer networks "never underestimate the bandwidth of a truck-load of tapes". Well, today it would be CDs and tomorrow DVDs but certainly we should not underestimate conventional distribution channels, such as posting material on CD ROM. The characteristics of this approach are a high bandwidth with a relatively high delivery time (days instead of seconds) and static data. Provided that we can overcome the inflexibility of having static data, however, this is a very acceptable solution. Static data is a problem as it promotes cycles of publication (e.g. by academic year), which often constrains flexibility. The system must therefore allow producers to update errata and to get away from the normally hard deadlines that such cycles impose without having to re-post large numbers of CDs.

Simply providing educational material on CD has other limitations. For pedagogical reasons, we may wish to monitor the student's progress through that material, building up models of their knowledge and preferred methods of learning[11,12] and then use these models to adapt the delivery of material on a student-by-student basis.

Currently, educational material is either delivered in CD form[13], for use with first-generation tools (web browsers) or is distributed on-line within the second-generation tools described above. A solution is required to bring flexibility in both the publication and delivery of material, coupled with both on- and off-line monitoring of the student's patterns of access.

In the TILE delivery system we will provide for the following methods of students' access to the educational material:

- *On-line with local server* - in this mode, most of the educational material is provided from the student's local computer having been distributed on CD or DVD. Additional software (the local server) is required on the student's computer to manage and monitor its delivery. The student still has access to the on-line archive on the education provider's remote server(s) and local information can be updated from this source.
- *Off-line with local server* - in this mode the student has no access to on-line material, but the local server software continues to manage and monitor the student's progress.
- *On-line with no local server* - in this mode, the student is unable to install the local software and must access all material from the education provider's remote server(s).

In the latter two cases, inconsistency may arise between the information on the student's computer and that on the education provider's remote server(s). This may be due to the publication of new material on the remote server or updates in the student's logs or models. In either case, mechanisms must be implemented in order to synchronise any inconsistent information.

## **2 The TILE architecture**

### **A client-server approach**

The architecture of the TILE education delivery system has have both client and server components. The client presents the information to the student and the server keeps track of and structures the information to be delivered to the student. The structure may be defined by the producer of the material, may be modified by the students' models or indeed, may be defined by the students themselves. In order to provide such flexibility, the system must be database driven. The flexibility provided by the database also allows for the distribution, updating and synchronisation of material.

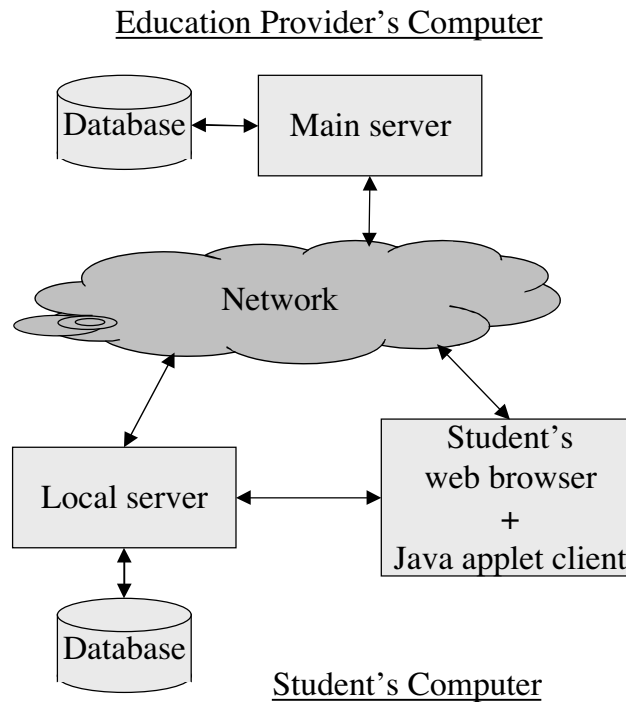


Figure 1. Overview of the TILE architecture

Because of our students' need for different modes of access, the client-server model we have adopted is slightly modified. Figure 1 shows this. The student's client software can talk to a local server application or to the education provider's main server. The local server is also a client to the main server. Thus we require a server application and a database on both the student's computer and that of the education provider. This gives us a potential problem, as we have no such control over the student's environment. Thus the local software must work with any operating system, web browser, etc. In short the local server must be completely cross-platform compatible. The minimum requirement that we can impose will be a Java-enabled web-browser, in which the client interface is implemented.

The number of layers or tiers in a given architecture will determine how we can group or distribute the functionality of the system. For a typical two-tier client-server architecture, the user interface and course-delivery logic would both be implemented on the client side and the database only would be located on the server side. The client application is charged with local processing, network processing and also connection to the database [2]. Development of such a *fat client* architecture is easy but maintenance is difficult.

A three-tier, client-server architecture adds a middle layer between the client and the database system. The user interface, course-delivery logic and data access are separated and the client is now only concerned with the user-interface and connection to the server. Most of the course-delivery logic is moved into the middle layer, where it communicates directly with the database. This *thin client* architecture provides increased security, flexible and significantly easier maintenance[2] and meets all of the demands of the TILE system. A web browser and Java applet provide the client software.

## Network Communication Protocols and firewalls

One of the major problems faced in the design of any client-server architecture is finding a solution that is compatible with the network security measures that may be in place. This may restrict the choice of network communication protocols. Special attention must be paid to the protocol between the local client and the remote server, as both client and server may sit behind a firewall, which will not allow some data packets to pass through. We can not assume control of the security measures implemented in a particular firewall. The particular issues we face in the TILE system are the loading of an applet from the server to the client and the network or socket connection between the client to server applications. It is this link over which we need to pass presentation data, course structure and synchronisation information.

For a client behind a firewall, the firewall will, in most cases, allow access to the web. This means that they can connect to server using the HTTP protocol. For the client-server connection, three firewall techniques may be used: IP filtering, proxy servers and SOCKS servers. Their impact on the client-server protocol choice has been analysed elsewhere[4] and the conclusion drawn is that for complete generality, the HTTP protocol is again the best choice of protocol.

### 3. Software required on the student's computer

#### Cross-platform issues

As already mentioned, the software on the student's computer must be platform independent. We have the choice therefore, to write the application for each platform supported or alternatively to use Java because of its "write once, run anywhere" capability. This provides a powerful reason for developing all software for the student's computer in Java.

For the client, web pages with embedded Java applets will provide the logic that is required for the adaptive presentation of the educational material. The applet will also support the TILE *online with no local server* access mode. We also require a local server and database on the student's computer, to support the other two TILE access modes: *online with local server* and *offline with local server*. Both server and database must run on any platform. Again, we may develop the server software in Java but although there are many database systems available, few meet the dual requirements of cross-platform support and low cost. A native Java database would meet our requirements provided that performance is not an issue; it could be run on any platform because of Java's characteristics and there are several open source Java database systems available[e.g.5, 6]. Performance it seems is not an issue, as the database will only be supporting a single user, the student.

Although it would seem that this strategy solves all of the cross-platform problems, this is not the case. There are several versions of Java development kits (called JDKs by Sun microsystems). Macintosh platforms will only support JDK1.1.8 under Mac OS 9 and below. JDK1.2, the current standard, is supported on most other systems, but will only be available from Mac OS X, yet to be released. It is imperative that we support the lowest common denominator in developing the TILE system. We are installing software

on the student's computer and must not force the student to upgrade either computer or operating system. Thus using JDK1.1.8 is the only answer for cross-platform portability but this introduces further problems. The package used for designing user interfaces, Swing, is only supported in JDK 1.2. Fortunately it is implemented natively (in Java) and we do have solutions to this.

### **Java applet security issues**

In general, an applet loaded over the network is prevented from reading and writing files on the client file system, from accessing a local database system, from making network connections, except to the originating host, or from starting other programs on the client computer. This is because an applet loaded from the network is not trusted[7]. There are two ways for an applet to be trusted by the client system. One way is that the applet is stored on the local disk, the second involves digital signatures. We have written programs to test a Java applet's security restrictions on a range of platforms and browsers. We found that an applet loaded from local disk is only treated as trusted by Appletviewer. Neither Netscape nor Internet Explorer trusts applets stored on the local disk to access the local file system or to open a connection to a local database.

Applet signing is another solution to make an applet trusted [8]. To be trusted, an applet must be signed with an unforgeable digital ID and then the user must state that s/he trusts applets signed with that ID. A digital ID is proof of identity of the person signing that applet. For a browser to understand a signed applet, there must be two digital IDs involved: one used to sign the applet and a second installed in the browser and used to verify the first one's authenticity.

Netscape and Internet Explorer have different approaches to granting trust to applets [9]. For Netscape, extra code, which uses Netscape's specific Java classes must be added into the applet. Signing methods are also different. For Netscape, the files must be signed with a Netscape Object Signing ID (creating a "manifest" of the files), and then the files and manifest must be wrapped into a .jar archive. For Explorer, the files must be wrapped into a .cab archive and then signed with a Microsoft Authenticode ID. Authenticode signing and Netscape Object Signing are not supported by all platforms. The ways that Netscape and Internet Explorer deal with expired digital IDs are also different. However, the most important and fatal issue concerning using signed applets to ensure security is that not all versions of Netscape and Internet Explorer understand signed applets.

### **Student-side application**

Since neither a Java applet loaded from local disk nor a signed applet can provide a satisfactory solution to the requirements of accessing local files and connecting to the database system, another solution must be found. Because there are no security restrictions on Java applications for JDK1.1, the Java local server application and database can provide all local functionality, providing that we can establish a connection between the Java server and the Java applet. When there is a need to access the local file system, the applet will send a request to the server application and it will access the database or file system. However, because both applet and application run on different Java Virtual Machines (JVM), there is no way for them to share memory. Again, a client-server approach is the only solution. The Java applet is the client, which opens network

connections to the local Java server application, which accepts network requests from that applet. The user interface logic can therefore be separated from the courseware delivery logic, with the user interface provided by the Java applet and all delivery logic by the local Java server application.

However, the network communication between the Java applet and the Java server application, even when running on the same machine, is still a potential problem.

### **Communication between the Java applet and the local server**

Java's security policy constrains an applet to be able to open a network connection only to its originating server and then only by naming the host in exactly the same way as the hostname of the HTML page in which the applet is embedded [7]. This means if the HTML page is loaded from URL: <http://www.nzedsoft.com/applet.html>, then the applet embedded in this page will only be able to open a network connection to the host [www.nzedsoft.com](http://www.nzedsoft.com). Neither the numeric IP address for [www.nzedsoft.com](http://www.nzedsoft.com) nor any shorthand notation for that name will work.

For *offline with local server* access mode, the applet which contains the user interfaces must be loaded from the local file system. But an applet loaded from <file://URL> is treated as a special case for security reasons. Such an applet is not allowed to open a network connection to "localhost", the local machine IP address or to <file://URL>.

There are two ways to solve this problem and one is the already discredited method of signing the applet. The second method must therefore be implemented, and that is to set up a simple web server on the student's computer and to load the applet from this local web server via a URL address. The function of this web server is only to provide the HTML page with the applet embedded in it, all other pages may be accessed directly from the local disc. Because both the web server and the local Java server application run on the same machine, they share the same host name. Therefore the applet loaded from a local web server may open a network connection to the local Java server application.

Thus, on the student's computer we must install a system, which comprises a database, a limited web server and the local Java server application. The applet that provides the user interface will be loaded from this local system, when the student is using their own computer, or from the education provider's server, when the student is using an anonymous computer. All of these programs will be implemented in Java JDK 1.18.

## **4. Software required on the education provider's computers**

On the education provider's computers, for firewall security reasons, all transactions with the student must use the HTTP protocol. The requirements of the education provider's server must therefore be met by adding functionality to a standard web server. In this situation however, performance is very much an issue. Although we have offloaded many of the transactions onto the student's local computer, any remaining transactions, to do with synchronisation and anonymous computer use, must be implemented as efficiently as possible to ensure good scalability. This logic may be implemented using a number of different techniques:

- the Common Gateway Interface (CGI);
- a web-server specific Application Programming Interface (API) ; or
- Java Servlets.

CGI is a very flexible technique and is used in many small-scale applications. It can be used to develop an application on any web server and on any platform, and it is programming language independent. Its disadvantages are bad performance and poor scalability, coupled with potential security holes. The scalability is poor, as each new user request requires a new process to be created on the server computer, and if the number of processes created is very large, then the performance of the system will be very poor.

A web-server specific API is much faster, as the application can be multi-threaded, which means that any number of different users will be managed by just a single process. This approach can also be optimized for the target platform. Its disadvantage however, is its poor portability, as each web server has a different API.

Java servlets are also an API approach but they are not web-server specific, provided that the web server supports a servlet API interface. Thus the advantages of a servlet approach is a combination of performance, portability, security and the full access to Java's functionality [10]. From our feasibility comparisons, Java servlets outperform the other two approaches.

Thus on the education provider's computers we must install a system that comprises a Java-enabled web server, a Java servlet API and a database server. We will have much more control over the choice of platform for the education provider; the computer may even be provided as a part of a total package. Therefore the problems that we have faced in cross-platform portability on the student's computers will not apply.

## **5. The complete TILE solution**

The overall architecture of the TILE education delivery system is shown in Figure 2. It is an extension to the classical three-tier, client-server architecture. Students interact with the system using a standard web browser. The intelligent user interfaces are provided by the Java applet. For *offline with local server* mode, the delivery logic is provided by the local TILE server application and a local database server provides the data services. For *online with no local server* mode, the delivery logic is provided in the remote TILE-enhanced Web Server and a remote Central Database Server provides the data services. For *online with local server* mode, the delivery logic and data services may be provided from either local or remote system, as appropriate.

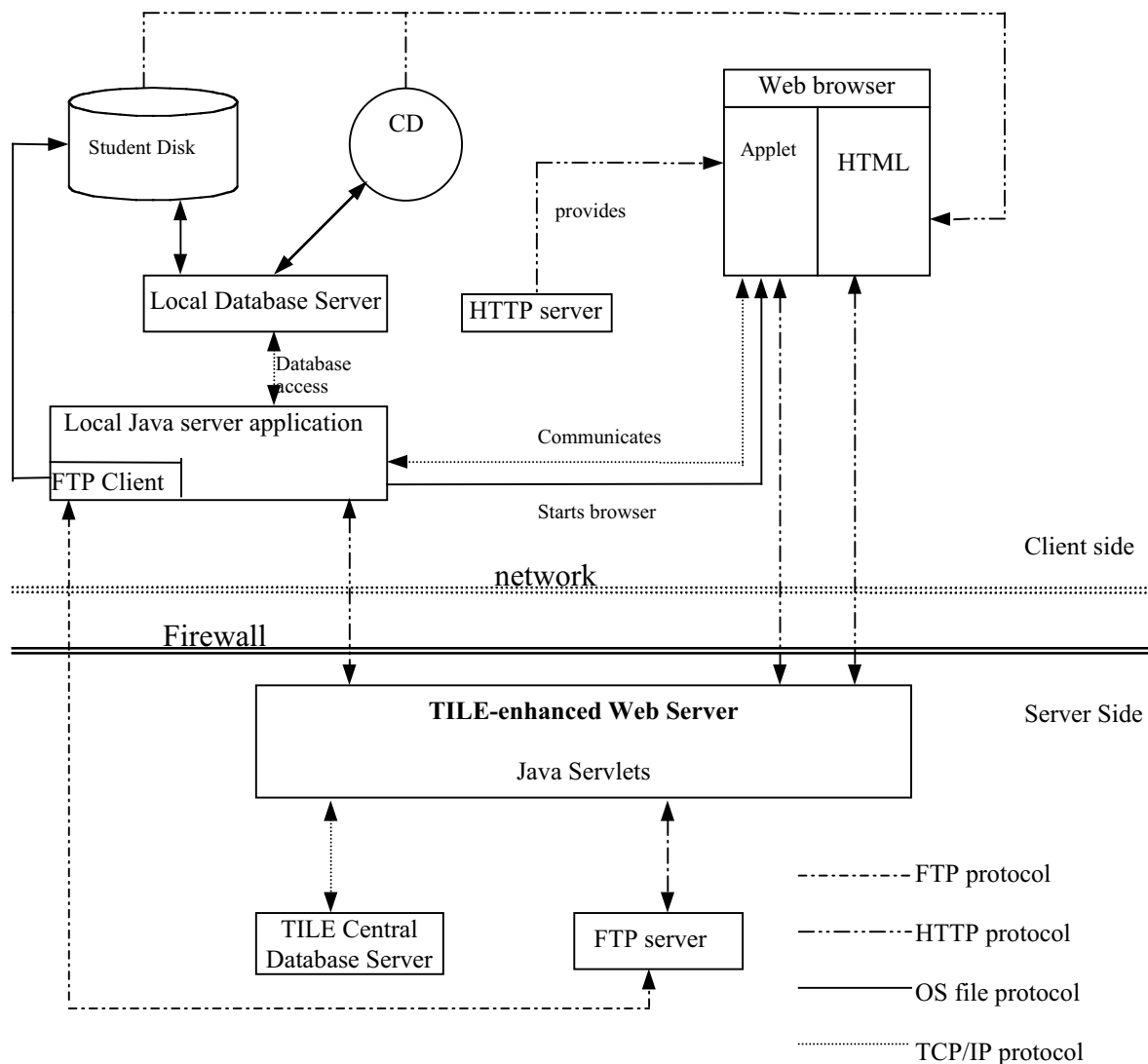


Figure 2. The TILE client-server architecture

## 6. Conclusion and future work

We have analysed the requirements for a third-generation, technology-integrated learning environment from the perspective of both performance and security as related to the implementation of a general distributed system. We have also analysed the student's requirements in terms of modes of access and flexibility of delivery, which will support a range of good pedagogical practices. Our final constraint was to produce a system that did not force the learner into updating his or her computer to a particular operating system or specification. The conclusions that we have arrived at are that systems design constraints as well as pedagogical requirements can be satisfied by the architecture that we have proposed in this paper. This architecture does seem complex but the implementation may make use of many existing components.

Data, such as the structure of a course, students models etc. will be abstracted using SQL databases. All of the delivery logic is contained in the two servers, one on the education provider's side and one on the student's side. The logic in the server on the student's side can be automatically updated by distributing the new application or applet from the education provider's side. This makes the overall system easy to maintain.

This design phase was the first phase of the TILE project and we are now proceeding with the implementation of the client/server system as well continuing with various authoring, querying and delivery tools. Further details of the project can be found at (<http://www-tile.massey.ac.nz/>).

## 6. Acknowledgements

We would like to acknowledge the support for this project from the New Zealand government's New Economy Research Fund (NERF) under contract MAUX9911.

## 7. References

- [1] C. R. Jesshope (2000) Integrated tools for on-line education, To be published, *Proc IWALT 2000*, Massey University, New Zealand, Dec 2000.
- [2] Joseph T. Sinclair and Mark Merkow, (2000) *Thin Clients Clearly explained*, Morgan Kaufman, Academic Press.
- [3] Alex Chaffee (accessed October 2000) One, two, three, or n tiers? Should you hold back the tiers of your application? [http://www.javaworld.com/javaworld/jw-01-2000/f\\_jw-01-ssj-tiers.html](http://www.javaworld.com/javaworld/jw-01-2000/f_jw-01-ssj-tiers.html)
- [4] Marco Pistoia et. al. (1999) *Java 2 Network Security*, second edition, Upper Saddle River, N.J. ; London : Prentice Hall.
- [5] InstantDB, (accessed October 2000) InstantDB home page, <http://instantdb.enhydra.org/>
- [6] Hypersonic (accessed October 2000) Hypersonic SQL, <http://hsq1.aron.ch/hSqlMain.html>
- [7] Sun Microsystems (accessed October 2000) Frequently Asked Questions - Java Security, <http://java.sun.com/sfaq/>
- [8] Daniel Griscom (accessed October 2000) Code Signing for Java Applets, <http://www.suitable.com/CodeSigningOverview.shtml#top>
- [9] Robin Green (accessed October 2000) Overcoming Java Security Problems on Netscape and IE, <http://redrival.com/greenrd/javasec.html>
- [10] Sun Microsystems (accessed October 2000) Overview of Servlets, <http://web2.java.sun.com/docs/books/tutorial/servlets/overview/index.html>
- [11] Nikov A. & Pohl W. (1999) Combining User and User Modelling for User-Adaptivity Systems. *Human Computer Interaction - Ergonomics and User Interfaces* (Eds. H.-J. Bullinger & J. Ziegler).
- [12] Hoschka P. (1996) *Computers as Assistants: A New Generation of Support Systems*. Lawrence Erlbaum Associates Publishers, Mahwah, NJ, New Jersey, pp336-340 (ISBN 0-8058-3391-9).
- [13] C. R. Jesshope (2000) The use of multi-media in internal and extramural teaching, *Proc Lifelong Learning Conference*, Central University of Queensland, Australia, ISBN 187 6674 06 7, pp257-262.