DIVISION OF COMPUTER SCIENCE

Teaching Client Server Computing to Undergraduates and Postgraduates

James A Malcolm
Ping Hu

Technical Report No.239

January 1996

TEACHING CLIENT SERVER COMPUTING TO UNDERGRADUATES AND POSTGRADUATES

James A. Malcolm and Ping Hu

Abstract

An academic course on Client Server computing will bring together aspects of computer science which would normally be taught in separate courses, in subjects such as Databases, Computer Networks, and System Development. A course based around a series of practical exercises using commercial software tools can provide an intellectually challenging course while also delivering some commercially relevant training.

Introduction

The Division of Computer Science at the University of Hertfordshire has added a course on the practical uses of network technology to our degree in computer science. This was done to meet the needs of students interested in information systems rather than the more traditional computer science topics like software engineering and computer systems, but in fact the course has proved to be extremely popular with students of both the "hard" and "soft" aspects of computing.

The course covers the use of the Internet and client server computing. Some academics feel that these are not proper subjects for study. But it is evident that "client server" covers a broad range of topics which is just as intellectually challenging in its breadth as a course on a more traditional subject would be in depth. Many of the topics that need to be addressed in a course on client server computing would in fact be covered in traditional courses on operating systems, databases, computer networks, and systems development.

In industry meanwhile, perhaps because of the breadth of understanding required, staff with the skills needed to successfully design and construct client server systems continue to be in short supply. The course we have written therefore includes a substantial amount of practical work -- in fact the course is designed around the practicals. These are supported by lectures which discuss the theoretical principles needed to ensure that the experience gained from practicals done in the laboratory can be transferred to whatever system and development platform a future employer may have chosen.

What a client server course needs to cover

Graduates need to deal with a broad spectrum of systems. It is therefore best if undergraduates have encountered more than one kind of system in practice as well as having learnt about them in theory.

But in most courses there is not time to look at many different kinds of system. So students also need to be given a framework for understanding the variety of system types. One common way of categorising client server systems is to use the diagram originated by the Gartner group. This identifies a range of systems from distributed presentation at one end, to distributed database at the other. Of course the simplest systems are between these extremes -- namely those that use either remote presentation or remote data management. Another axis for categorisation is to look at how the data is structured. A database approach is suitable for many applications, but for others a less structured data model is appropriate.

In larger systems, there may be issues of server coordination (OLTP, concurrency control, database recovery, etc.) to be considered. The degree to which the system may need to be audited is also an important issue.

Finally, clients may run on a variety of operating systems and architectures (PC, Unix, Mac) and may be developed with a variety of software development tools. Similarly, there can be various server environments.

When we came to define the new course, the list of the main topics that should be addressed in the course was as follows:

- Understanding the client server concept
- Interfaces and portability
- Protocols and interworking
- Data design
- Inter process communication
- Concurrency control and transaction processing
- Meeting business requirements
- The rôle of the users

Perhaps we are being hopelessly overambitious in trying to squeeze the whole of a computer science degree into a single course!

In terms of the theory, client server computing doesn't represent anything new. Topics such as RPC, SQL, concurrency control, etc. have been in our courses for years. Formal methods appropriate for the design of concurrent systems have also been taught. The following three sections describe some of the approaches to relevant subject matter that have been used in previous courses, and the lessons learned from them which were relevant to the new course.

Computer networks course

In the computer networks course that has run for many years as part of the computer science degree scheme of study, we have a series of 5 practical exercises, taking about two weeks over each. These are:

- Network applications
- Data link protocol simulation
- Token passing MAC protocol simulation
- Distributed adaptive routing protocol simulation
- Network programming

The network applications practicals were good for motivating the rest of the course, especially for students who had not had first hand experience of using computer networks. The practicals included several applications that used a client server model. However it was noticed that although this practical covered the use of the Internet and the World Wide Web, students were in general unable to answer an examination question on this topic. It seemed that without teaching a topic in lectures, there was no recognition of its importance.

The simulations were used very much to back up the lectures, helping to clarify how the various protocols operate. They seem very effective. In particular a weaker, but industrious, student can gain a very good understanding of protocols at three of the most important layers in the ISO OSI reference model by carefully working through these exercises in his or her own time.

The network programming practicals were the most vocational, but perhaps the least effective for the average student. They included some very simple client server programming using TCP/IP, though the main purpose at the time was to help the students appreciate the differences between connection oriented and connectionless service. They went on to use an RPC package to construct a simple application. The problem was one of complexity. To get the most out of these exercises, students had to be adept in the use of Unix and reasonably competent in C programming. For part time students especially, who only have one hour per week on practical work, this was a big difficulty. Complexity was also a barrier in as much as students had to gain a reasonable understanding of the BSD socket interface. Finally the commercial RPC system used was quite complex, and in particular did not really meet the definition of RPC as a system that transparently allows a procedure call in one address space to cause a procedure invocation in another address space.

The lectures do not follow a book, but it is important that the students have a source of information other than their possibly inadequate notes and the lecturer's handouts. They are advised to buy "Computer Networks" by Andrew Tanenbaum -- rather an old text now, but the best source to backup the lectures. Each lecture is linked to appropriate sections and exercises in the book.

Advanced computer systems course

This course is part of the combined studies scheme, so is studied by final year students who are taking computing in conjunction with another subject. In this case we follow a textbook quite closely: Jean Bacon's "Concurrent Systems". This provides an integrated view of a range of topics: Computer architecture, operating systems, computer networks, database management, and transaction processing. Our experience confirms her view that these topics can be effectively integrated. Integration makes especial sense for these students for whom this may be their last formal course in computing -- an in depth course in any one of these areas would not be nearly so appropriate.

This course also gives students a chance to investigate an area of their own interest. Such a student centred approach can sometimes be an excuse for inactivity on the part of the lecturer, and may not produce much activity from the students either. But we have found that if students are given regular feedback and advice, they can work independently and learn a lot and enjoy it.

Client server course

As part of our M.Sc. in Information Management, we ran a course on Client-Server computing. This was a two day short course, mostly based on lectures. This approach can cover the ground, but is not very exciting for the students or indeed for the lecturer. If nothing else, practicals are needed to break the monotony.

The lectures covered the definition of client server and why it is important from a technical and a business perspective. The need for data modelling was examined, with revision of basic techniques like entity-relationship modelling. The data model can either be built into both client and server (as with RPC or with most commercial, SQL based, systems) or can be passed from server to client (as in the web, or a general purpose database browser). Different methods of IPC were examined, particularly RPC. RPC was also used to illustrate (OSI) presentation layer issues of data representation. Finally we looked at concurrency control systems for single and multiple databases.

As part of this course, a case study was introduced to look at how client server techniques had been used to solve a specific commercial problem.

Design of the new course

In recent years in the UK, we have seen a dramatic increase in the number of students entering higher education. It must be the case that the typical student today is somewhat less able than those of the past. The best students are still just as good, but some of the weaker ones are not best served by maintaining the assumptions of 20 years ago.

We can (and do) test depth of understanding in our examination questions, and inevitably weaker students do particularly badly on such parts. We are aiming to produce students who are self starters, who can continue to develop and learn throughout their careers, and who do not incessantly need to go on training courses. But we should try not to lose the less academic students. Even students who get third class degrees can be well thought of in industry, but for this to continue it does mean that our course content should be informed by industrial requirements.

Part of that requirement is the combination of database and communications known as client server. But we cannot test the integration of knowledge across courses, because all final year courses are optional.

Because client server computing is such an important area, we wanted to create a course where the integration between the various topic areas of the subject is done explicitly by the course rather than implicitly by the students. Leaving students to provide the integration between the topics for themselves is not always successful. Because breadth of understanding is so important to their practical success in this area, our choice has been to provide the integration in a new course.

Plan of the practical use of network technology course

This new course follows a revised version of the first semester course on computer networks mentioned above. Particularly in the second semester, one cannot plan for more than ten weeks of teaching. The week by week plan for the practical work undertaken by students on the Practical Uses of Network Technology course is as follows:

In the first week, students set up their own World Wide Web home page. This is an easy task that can be done before any lectures -- indeed many students will have done this for themselves. The lectures will expand on the client server relationship between the Web client and server, the nature of the http protocol, and the two kinds of presentation issues. Having mastered simple html, students can be asked to produce a simple application using the web, of a sort that would be more usually implemented using a database package.

After we have explained the common gateway interface (cgi), in the second week students will continue to use the Web architecture, but now the effect of following a hypertext link will be to run a program rather than merely returning a piece of static data. By using a program to search a file, we can introduce simple database functionality as well as the idea of server side scripts. At this stage all the required URLs will be typed by hand.

The next stage is to use the forms capability of html to improve the user interface and of course to allow the user to select which record in the file is to be displayed. An important lesson here is the way in which the client now has to know something about the data that it is manipulating, though of course that knowledge is loaded into the Web browser only shortly before it requests the cgi script to search the "database".

The fourth week is about server configuration. Such mundane activities could easily be left out, but are an important part of client server computing. There are many important issues of authentication, access control and electronic commerce that could also be addressed at this point.

Most of the documentation for these tasks is available on the Internet, so students can be encouraged to take more responsibility for their own learning, and hence learn how to learn, by giving them pointers to on-line documentation rather than giving step by step instructions.

We now move from Web-based client server applications to applications based on relational database technology. It would be good to use the same case study throughout the course, but the amount that can be achieved by an average student in one week is quite limited. The kind of case study that seems appropriate for the first four weeks looks rather trivial when implemented on a proper database system, so a larger example is introduced at this point.

The activity for week 5 is to build the database schema for the (larger) case study using a client server database design tool. Schemas can be produced for a PC database as well as for a mainframe RDBMS.

The practical task for week 6 is to set up the mainframe database, populating it with suitable data and testing it using (local) SQL commands.

In week 7 of the course, students use a PC based browsing tool to perform ad-hoc queries on the server database. We do not anticipate being able to let students perform configuration tasks for themselves on the mainframe database, or on the (networked) clients. This is one of the reasons for spending a week on configuration (and security) earlier (in week 4).

The next task is to produce the same database in a PC based DBMS. Given the design from week 5, the test data from week 6, and the experience of week 7; this should be a straightforward task -- especially when one considers the ease of use of modern PC based packages.

In week 9, we now introduce some client side processing of the data fetched from the server by introducing simple use of one of the so called "visual" programming languages that provides easy access to remote databases. This can be used both with the local PC based database and with the mainframe based RDBMS. This task should provide further help in understanding the range of choices about what is done at the client and what is done in the server, though the amount of actual programming that can be achieved will be quite limited.

In the final practical activity, the aim will be to partition the data being used between the PC database and the mainframe database. It is important that this is not seen as a proper distributed database. OLTP will be dealt with in the lectures this week, but distributed database is otherwise a topic mostly for the Database Management course. This activity does give a good illustration of what client server technology can do, and of the range of design choices that are therefore available.

If there were time, it would be good to introduce a practical exercise using a remote procedure call system. This would illustrate another, quite different, approach to client server computing.

Commercial software used

Apart from the Unix based web server and development environment, all of the software used is commercial:

- Client-server CASE tool
- PC database package
- Unix database package
- Commercial "middleware" standard
- Pascal based "visual" programming language
- Web browser (used both on Unix machines and on PCs)
- RPC package

Because we wanted to relate the course to the real world, the kind of functionality we needed was only available in commercial products. However, we have used products from four different vendors in the database part of the course, and from two other vendors for the web and RPC parts. We thus avoid presenting an over narrow view of how computing should be done.

Relation to the rest of the scheme of study

A degree student currently studies 10 modules per year. The only compulsory course in the final year of the B.Sc.(hons.) degree in Computer Science is the (4 module) project. A student who wanted to concentrate on the topics most relevant to client server computing, could choose to study courses in the following pattern:

Computer Networks	PUNT
Database Management	
User Centred Design	
Project	

Database Management: this 2 module course builds on the foundation of earlier years. It deals with relational theory, extended relational model and object oriented databases. It looks at databases from the points of view of using, designing, and managing a database.

User Centred Design: this is also a 2 module course. It deals with the design of interactive systems, focusing on issues of human computer interaction. Visual Basic is used as a vehicle for developing such skills in practical work.

Computer Networks: this course concentrates on how networks operate, rather than how they are used. Applications are introduced to motivate the rest of the material, to justify the need for different network characteristics, and also to show how technological developments have engendered new uses for networks. The course tries to emphasise principles rather than the details of particular network systems.

Practical Uses of Network Technology: as discussed above.

A practical project in a relevant area: of 4 modules. Students work individually on a substantial piece of work, which is assessed by considering the work done, the project management, and the quality of the final report.

This set of courses, we believe, would provide a well integrated preparation for a career in client server development. Of course, students have a completely free choice and some choose to study the practical uses of network technology in quite eclectic combinations with other subjects.

Conclusions

Designing a course around what the students will do, rather than what you want to tell them about, makes a lot of sense.

A practical course in client server technology should not be merely a training course designed to meet some current need of industry. Industry's needs will change soon enough, leaving graduates with out of date skills.

Instead, a course which uses the latest commercial software can provide an academically sound education, whilst including some industrially relevant training at the same time.

Acknowledgement

The protocol simulation practicals were the work of Ben Potter.

Footnote: an imperfect analogy

I think this is a little like the Pascal *vs.* COBOL arguments of days gone by.

If you teach COBOL, all students will be useful in industry, but will eventually need retraining. On the other hand, if you teach Pascal, the good students will learn COBOL (and Ada, and C, and C++, and VB...) for themselves, but the poor ones will need retraining immediately.

Of course, what we are trying to teach is *programming*, for which Pascal is the better vehicle. My approach is analogous to teaching programming, but using COBOL as a vehicle to code your programs.